SECOND EDITION

THE BOOK OF INKSCAPE

THE DEFINITIVE GUIDE TO THE GRAPHICS EDITOR

DMITRY KIRSANOV

no starch press

REVIEWS FOR *THE BOOK OF INKSCAPE*

"Highly recommended, well written, and full of practical examples and insights, this excellent text can make you a pro at using Inkscape."

—IT World

"Inkscape is a really exciting, interesting, and powerful design program to learn, and *The Book of Inkscape* makes this learning process fun and rewarding."

—Dr. Dobb's CodeTalk

"Dmitry Kirsanov's book enables you to sit down with a freshly downloaded version of Inkscape and work in an ordered and logical way through the software features."

—Alan Berg, Free Software Magazine

"This book is a great guide to using Inkscape."

—On Computers

"Clear writing and numerous step-by-step tutorials make this manual the go-to reference for artists new to Inkscape and for experienced users looking for detailed information on the application's features."

—SciTech Book News

THE BOOK OF™ INKSCAPE

2nd Edition

The Definitive Guide to the Graphics Editor

by Dmitry Kirsanov

no starch
press

San Francisco

First printing

24 23 22 21 1 2 3 4 5 6 7 8 9

ISBN-13: 978-1-7185-0175-1 (print)
ISBN-13: 978-1-7185-0176-8 (ebook)

Publisher: William Pollock
Production Manager and Editor: Rachel Monaghan
Developmental Editor: Jill Franklin
Cover and Interior Design: Octopod Studios
Technical Reviewer: Joshua Andler
Compositor and Indexer: Alina Kirsanova
Proofreader: Scout Festa
Illustrator: Dmitry Kirsanov

For information on book distributors or translations, please contact No Starch Press, Inc. directly:
No Starch Press, Inc.
245 8th Street, San Francisco, CA 94103
phone: 1.415.863.9900; info@nostarch.com
www.nostarch.com

The Library of Congress issued the following Cataloging-in-Publication Data for the first edition:

```
Kirsanov, Dmitry.
  The book of Inkscape : the definitive guide to the free graphics editor / Dmitry Kirsanov.
       p. cm.
  Includes index.
  ISBN-13: 978-1-59327-181-7
  ISBN-10: 1-59327-181-6
 1.  Computer graphics. 2.  Inkscape (Electronic resource)  I. Title.
  T385.K491256 2009
  006.6'8--dc22
                                                                2009023973
```

BRIEF CONTENTS

CONTENTS IN DETAIL

ACKNOWLEDGMENTS

This book would not exist if not for the help and encouragement of the No Starch Press staff, in particular Jill Franklin, Rachel Monaghan, Barbara Yien, and Bill Pollock. I am also indebted to the technical reviewer, Joshua Andler, whose suggestions really improved the book.

I owe gratitude and respect to the Inkscape development team—dozens of dedicated volunteers who spent countless hours creating one of the best vector editors now in existence. It was a pleasure and honor to work with you.

Finally, special thanks go to my wife, Alina, for her unwavering support, constant prodding, and the huge work she did compositing, indexing, and proofreading the book.

INTRODUCTION

This book is about Inkscape, the powerful, open source SVG-based vector graphics editor. It describes 1.1, the latest version, released in 2021 after almost two decades of development, in great detail, covering all of its features, excluding only those that are experimental or incomplete.

The second, perhaps more ambitious goal of this book is to evangelize the vector way of creating graphics and share the joy of thinking in vector. Once the exclusive domain of professional designers, vector graphics software has become more common and accessible. A vector editor is not yet a standard app in the same way a bitmap editor (like Photoshop) is, but it's getting there—and Inkscape has been a major cause of this shift.

Much of Inkscape's appeal is due to the fact that it's the only fully open source and cross-platform professional-level vector editor. But that's not all. Simple and accessible at the basic level, Inkscape is also extremely powerful if you dig deeper. It is vast and inexhaustibly hackable—from customizable keyboard shortcuts (3.4) and a command line language (Appendix C), to powerful chainable path effects (Chapter 13), to arbitrarily complex vector filters that push the envelope of vector graphics (Chapter 17). This book swims the Inkscape ocean from shore to shore leaving no island unvisited.

What's in This Book?

Chapter 1 is a high-level description of Inkscape, its capabilities, and its place in the world of vector graphics (and, to some extent, computer graphics in general). Next, Chapter 2 gives an absolutely down-to-the-basics primer on the first steps in the program (installation, opening, creating objects, transforming them, and saving); you can safely skip this chapter if you have ever completed even a small Inkscape project.

The bulk of the book, Chapters 3 to 19, is written to be read more or less in sequence. These chapters present all aspects of Inkscape's functionality (objects, transformations, style, paths, text, effects, and more) with detailed explanations, illustrations, and practical tips.

The second part of the book (starting from Chapter 20) contains several complete step-by-step tutorials demonstrating real-world uses of Inkscape, including an animation, a business card, and a complex illustration of a rose in different styles. The book ends with several appendixes that provide reference information.

Inkscape is a work in progress—even if this progress may at times seem slow. The previous edition of this book described version 0.47 and was published way back in 2009. Since then, Inkscape has evolved and matured enormously, and this new edition brings the book up to date with that development. Here are the major features that are new in Inkscape 1.1 compared to 0.47:

- **Speed!** Inkscape's slowness has been the number one user complaint for many years. Version 1.1 brings great improvements in rendering speed and UI responsiveness, especially with filters (Chapter 17) as OpenMP multithreading is now supported for all filters.
- **New UI.** Inkscape has a modernized GUI toolkit based on GTK 3 and a more usable implementation of dockable dialogs.
- **No more SVG-incompatible text.** In previous versions, creating flowed (autowrapped) text made your file non-compliant SVG because Inkscape used an obsoleted draft SVG specification for this feature. Now, Inkscape-flowed text is valid SVG and displays correctly in all SVG-compliant renderers including web browsers (Chapter 15). Also, there's a new option for autowrapping text into a column of unlimited height (15.2.1.1).
- **New tools.** Eraser (14.4), Spray (4.7), Diagram (14.5), and Gradient mesh (10.7).
- **More powerful Path tool.** You can now edit multiple selected paths simultaneously.
- **Canvas conveniences.** You can now use keyboard shortcuts to rotate the canvas for easier editing or drawing as well as mirror the canvas vertically or horizontally (3.13).
- **Many new path effects and extensions.** These are described in Chapter 13.
- **Blend modes.** These tools are now supported as a powerful way to alter how the overlapping objects in your art blend together (17.2).
- **More and better filters.** Many new preset filters (17.4) are available, and you now can edit the filter area on the canvas (17.3.1).

- **Bitmap performance.** Inkscape can now import, display, and export much larger bitmaps (Chapter 18).
- **More export formats.** In addition to PNG, Inkscape can export bitmaps to JPG, WebP, and TIFF (18.6).

NOTE *Throughout the book, material that is new in Inkscape 1.1 is indicated by* ***[1.1]*** *in the margin.*

For the latest updates and news about the book, visit *https://nostarch.com/inkscape*.

Who Is This Book For?

This book can be your first introduction into the world of vector graphics. You don't need to be a graphics professional to find it useful. You will learn some new concepts and terminology, but that should come naturally if you practice what you study. Install Inkscape (if you haven't already) and immediately try the techniques as you read about them in this book.

Even those who are experienced in using Inkscape or another vector graphics program will find enough to chew on in this book. I am writing from two perspectives—that of a longtime Inkscape user and that of a developer who has contributed a lot of code to it. I hope this dual perspective allows me to see the logic of the program's interface and behavior more clearly, and to identify its unique strong and weak points (as well as suggest workarounds).

1

INKSCAPE AND THE WORLD

Maybe this is your first encounter with vector graphics. Maybe you have used vector graphics before and are now curious to see what else it can do for you. Or maybe you are considering Inkscape after you've tried other vector editing applications and want to know what sets it apart. Whatever the case, you may find some background information enlightening. What is SVG? What is Inkscape? Where does it come from and where is it headed? And what can you use it for? What is Inkscape's place in the world of computer graphics? This introductory chapter gives up-to-date answers to these questions.

1.1 What Vector Graphics Is and Why It Matters

Inkscape is a *vector graphics editor*. What does that mean?

Most of the images stored and processed on computers today are represented as *rasters*, also called *bitmaps*. A raster image is a primitive representation—just a lattice of small rectangular areas called *pixels*. For each pixel, the only information stored is its color and, sometimes, its transparency.

For example, if you have a bitmap image with a black circle on white background (Figure 1-1, left), there is in fact *no* black circle as such stored in the image. It's only when *you* view the image that you may (or may not) get the idea of a black circle. All the computer knows about the image is that some of its pixels are black and some are white (and a few are an in-between gray).

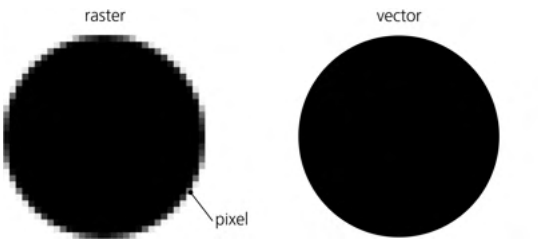


Figure 1-1: A circle as a bitmap or raster (left) and vector (right)

As a result, there is little the computer can do with such an image without human guidance. It can change all the white pixels to blue, but it cannot readily move the circle because it does not *see* it as a separate object. Such a task may be difficult even for humans, as anyone who has used GIMP or Photoshop would attest; you'll have to use some fickle and unreliable tools to "select" the circle—and this is especially hard to do if, for example, the edge of the circle is *anti-aliased* (that is, some pixels on the edge have intermediate values between black and white) as in Figure 1-1.

All of this is different with *vector graphics* (Figure 1-1, right). In a vector format, the actual circle is stored, along with its properties, as an *object*. This means it is separate from any other objects, so you can do whatever you please with it. With such an image, your computer can do many smart things automatically—for example, it can delete all circles, paint all red objects with green, or scale all black circles to twice their size.

No more frustrating pixel selections—just pick any object, any time, and edit it as needed. This is how Inkscape works—and this is its main point of difference from raster editors such as Photoshop.

Let's look at the most prominent advantages of the vector approach:

Vector images are scalable.

Scalability means you can view or export your drawing at any resolution, and you'll never see any jaggedness, pixelation, or unwanted blurring. Everything remains perfectly crisp regardless of size. This is often cited as the main advantage of vector graphics; in my view, other advantages are no less important.

Vector images are editable at any time.

No matter how complex your drawing, you can always pick any object in it and edit away. Think of it like a Photoshop file in which every single brush stroke is placed on a layer of its own—automatically. Furthermore, in a raster editor, you are supposed to eventually "flatten" your image, so that all separate layers are merged. By contrast, there's no need—indeed no possibility—to flatten a vector drawing (except by exporting it to a raster).

Vector images are easy to create and read.

Vector objects conform much closer to human visual perception. When we look at a scene, we tend to mentally separate it into objects—just as is done in a vector drawing. This makes vector graphics a very natural medium to work in. Also, since many vector formats (including SVG) are text-based, it is easy to *write* a simple vector drawing manually (even without a graphic editor of any kind) or to program a script to generate or modify such a drawing.

Vector images are laconic.

Since you don't have to store information about every pixel, a vector image usually takes much fewer bytes to store and transmit than the corresponding raster image. As a nice side effect of this, Inkscape has an unlimited undo history, simply because each undo step takes much less memory as a vector than it would as a raster.

Vector images are infinite.

A raster always has a fixed size in pixels—for example, 468 by 60 pixels. In an uncompressed bitmap, doubling the dimensions will quadruple the size of the file because the extra pixels need to be stored even if they are empty. Not so with vector images. A vector image is virtually *boundless*, and expanding it by moving an object an inch or a mile away costs nothing in terms of file size or computer memory. Similarly, a vector document is virtually infinite in terms of *depth*: you can zoom in as close as you want and create any number of microscopic-sized objects in any given space. (In reality, of course, both canvas size and zoom level are limited, but these limits are way beyond those of a raster editor.)

Vector images can be animated.

Since, in a vector drawing, objects are stored separately, it's easy to animate them by moving or transforming them, changing their colors, and so on. Naturally, some vector formats, such as SVG or Flash, have animation capabilities built in. Inkscape does not yet support animated SVG, but this may change in a future release.

Vector images can be interactive.

Not only can you animate objects, you can also make them interactive. A drawing can change its objects' properties in response to user actions—which enables complex user interfaces with buttons, links, drag-and-drop, and so on. Again, Inkscape's current capabilities in this area are limited, but you can manually edit an SVG file created in Inkscape, adding interactivity, which can then be played in an SVG viewer.

Vector objects are reusable.

It's easy to pick an object from one drawing, transform or restyle it without any loss of quality, and then insert it into another drawing.

If you are into digital music, you may better understand the vector/raster distinction if you think of vector graphics as similar to a MIDI sound file and raster graphics to a WAV sound recording. Programmers might think of vector as the "source code" of an image and raster as its "compiled binaries."

Of course, it can't be all roses and no thorns. Here are the two main disadvantages of vector graphics compared to rasters:

Formats are fundamentally limited.
A vector image format (of which there are many, just as there are many different raster formats) always limits you to a certain repertoire of objects and their properties. New versions of formats invent and introduce new capabilities, but that only highlights the fact that perfection is unattainable. Vector formats are forever "under construction."

Many images are difficult or even impossible to reproduce exactly in vector form. For example, images that require complex textures, such as human skin, hair, or wood, are inherently vector-unfriendly. The world of vectors has traditionally been made up of precise shapes, flat colors, and smooth gradients; if what you want is naturalistic texture of noise, scratches, or dirt, you will have a hard time rendering that in a typical vector editor.

Still, with SVG's transparency, gradients, and filters, you can create amazingly photorealistic vector images in Inkscape (see Figure 1 in the color insert). Also, being a higher-level abstraction, any vector drawing may include raster images as a special kind of object. That is, you can always insert a photo into an Inkscape drawing and combine it with any vector objects.

Vector format conversions are unreliable.
For the same reason, it is never trivial to convert an image from one vector format into another. The repertoires of object types and capabilities of the formats always differ, sometimes in nonobvious ways; different versions of formats muddy the picture even more. For example, SVG supports blurring objects (17.1) while PDF does not; on the other hand, PDF supports gradient meshes (1.5.5), a feature presently missing in SVG (even though Inkscape supports it on an experimental basis, 10.7). This makes any kind of vector conversion an iffy business. In the best scenario, objects that the target format cannot represent directly will be approximated—for example, for exporting SVG to PDF, blurred objects may be replaced with embedded rasters. In the worst scenario, format conversion will simply produce a more or less broken image.

This is one of the reasons Inkscape's vector format, SVG, is so important. It is an acknowledged international standard with rich and well-defined capabilities, which therefore can serve as the lingua franca of vector graphics (although Adobe's PDF is still much more common in this role). We will discuss SVG in more detail later in this chapter.

So what is actually stored in a vector drawing? Apart from the embedded raster objects just mentioned, the most common vector object type is a path. A *path* is just a sequence of commands like "draw a straight line to such-and-such point" and "draw a smooth curve through such-and-such points." There may be any number of such commands in a sequence, which means a path can approximate any geometric or real-world shape with any desired accuracy. A path can have *fill* (paint in the area enclosed by the path) and *stroke* (paint along the path itself), as well as many other properties that define how the path looks (Figure 1-2).

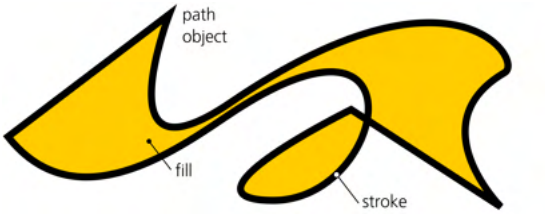


Figure 1-2: A path object can represent any shape.

There are several other object types (such as text objects, clones, and groups) and many other object properties (such as font size, visibility, and blur). Many properties can apply to all kinds of objects, while others are specific to particular object types. A drawing is just a collection of objects of various types; you can place these objects wherever you need them, including on top of each other, and you can make them partially or fully transparent so that whatever is underneath shows through. Figure 1 in the color insert shows an example of a complex Inkscape drawing that makes heavy use of transparency, gradients, and blur to achieve realism (load it from Inkscape's *examples* folder if you want to examine it in its full glory).

1.2 What Can You Do with Inkscape?

A lot.

Schemes, charts, diagrams. Plans and drafts. Scientific illustrations and data graphs. Icons, symbols, logos, and emblems. Heraldry, flags, road signs. Comics, cartoons, anime characters, page layouts. Maps of lands—real or imaginary. Typography of all kinds. Banners, leaflets, posters. Web graphics. (Ads, too.) Book covers, holiday cards, headings, and vignettes. Kids' scribbles and stunning photorealistic art. Fantasy art, fan art, games art, and simply art of all flavors and varieties.

One of the goals of this book is to demonstrate that vector editing tools are applicable to a much wider range of tasks than is usually acknowledged. In fact, instead of trying to list all of the uses for which Inkscape is suitable, it's easier to describe situations where it is *not* a good choice. Let's try to lay out the borders of the Inkscape universe.

- Many tasks with photos or any other *preexisting raster graphics*, such as color correction, retouching, and format or size conversions, are better done in raster editors such as GIMP or Photoshop. Of all the limitations of vector images, this one is the most obvious because it separates the two commonly contrasted kinds of graphic tools: vector and raster. Note that some raster-related tasks—such as adding callouts or marks, drawing shapes over a bitmap background, masking (18.3), or even simple retouching (18.4)—can still be done quite naturally in Inkscape.
- Drawing with *natural-media tools*—those emulating oils, pastels, watercolors, and so on—is best done in specialized raster tools, such as Krita, Corel Painter, MyPaint, or ArtRage. More generally, this applies to any art where the *texture* of the colored surface is what matters most. Still, if you care more

about shape and color than texture, or if your art looks good with just flat color, gradients, and blurs, Inkscape is one of the best tools for "simply drawing" (Chapter 14).

- Producing *text-rich multipage documents*, especially with complex features such as footnotes, indexes, or a table of contents, is naturally the domain of page layout software (such as Scribus or Adobe InDesign) or batch formatters (such as TEX or Apache FOP). Inkscape, however, works very well for graphics-rich single-page documents, such as posters or leaflets. You can even use it for some multipage designs by storing each page in a separate document or layer (it does not yet support multiple pages within one document). Also, Inkscape lacks support for CMYK color separations or spot colors, which limits its usefulness for print work.
- While Inkscape's 3D Box tool (11.3) can be used for simple *three-dimensional drafts and scenes*, it is a more of a tool for a traditional 2D artist whose drawings depict 3D objects, not a 3D artist creating 3D worlds. In other words, if you need a one-off drawing of a simple 3D scene, you can use Inkscape to get a nice-looking and geometrically correct result. If, however, you want several renditions of the same scene from different angles or a 3D animation, use some real 3D software instead (Blender, Maya, or SketchUp, to name just a few).
- You can do some simple *CAD* (Computer-Aided Design; this term usually applies to engineering drawings) work in Inkscape. Inkscape provides ways to draw and transform objects precisely, as well as a plethora of snapping, alignment, and distribution features. However, Inkscape does not support features like parametric modeling, nor does it yet have any libraries of CAD elements (such as screws or tubes) that are essential to professional CAD work. While you may try to borrow such elements elsewhere, in most cases it is still better to use a specialized tool, such as QCAD or AutoCAD.
- Inkscape has a dedicated Connector tool that you can use to draw pretty complex *diagrams and flowcharts* with automatically routed connectors. This tool is rather limited, however; if you need to create many standardized diagrams, look into specialized tools, such as Visio or Dia.
- Some people have successfully used Inkscape for *presentations*. With its ease of manipulating objects and many eye-candy effects, Inkscape is indeed an attractive choice when you need to concoct a presentation. You can create and reuse a page template with headers and placeholder text (3.2). Inkscape even includes a stand-alone SVG viewer (Inkview) with fullscreen mode and spacebar key for the "show next" command, which is typically all you need to show a presentation whose pages are saved as separate SVG documents. Office presentation applications, such as PowerPoint or OpenOffice.org Impress, still have their advantages, although these programs tend to feel clumsy once you get used to Inkscape's graphic power. Some people use Inkscape for drawing the graphics and a presentation application for adding text and creating the actual presentation. There are also helpful add-ons, such as InkSlides (converts a multilayer SVG into multipage PDF) or JessyInk (converts a multilayer SVG into a scripted presentation that you can view in Firefox or another SVG-capable browser).

- Inkscape's path effects (13.3) and extensions (Chapter 19) can render a lot of interesting graphic artifacts, such as Lindenmeyer systems, random trees, spirograph curves, or barcodes. New extensions are easy to program, too. However, if what you need is some complex *algorithmic art*, such as fractals, use specialized software and import the result into Inkscape.
- Inkscape does not yet support SVG *animation* and won't run *scripts* in an SVG document (though you can still view and edit a document's scripts in Inkscape). So while you can use it for drawing animation frames, characters, and UI mockups, you will need a different application to combine these elements into a working animation or interactive application.

Admittedly fuzzy (and still changing), these are the current frontiers of the vector graphics land. Everything *within* these boundaries is the rightful domain of a modern vector editor such as Inkscape. Note that only some of these limits are inherent in vector graphics as such; others are just current limitations of Inkscape that may be overcome in future versions.

Curiously, a lot of people roam the outskirts of the graphics land but are oblivious to the vector heartland. As a result, they often frustrate themselves trying to use their favorite specialized tool for the wrong generic task. Examples include Microsoft Office users struggling with PowerPoint whenever they need to make any kind of picture or layout, or novices complaining in forums about how difficult it is to draw simple geometric shapes in GIMP. These are typical cases of vector blindness; do not fall prey to this disease!

The vector land is also the place where a lot of project roadmaps begin. No matter what kind of project I am starting and what software I will end up using, my first step is usually to open Inkscape and start making quick drafts. Only when I run into some of the limitations listed above might I move on to more specialized applications to complete my project. More and more often, I find that I don't really need to leave Inkscape to finish what I started. Inkscape's universe keeps expanding.

1.3 Sources of Inkscape Art

A tool is dead without a community of users, and a community of users is nonexistent without a body of work they can study and reuse. Inkscape wouldn't be quite as fun to use if you always had to start a project from an empty page—or if you had no one to share your work with.

The two main reasons to seek SVG art are *learning* and *reusing*. Reusing is simple; this is what the whole idea of "clipart" is about. Instead of drawing everything from scratch, you take elements created by someone else and combine them with your own work (if the license for those elements permits that, of course).

When reusing others' art, source format is not too important so long as Inkscape can read it (Appendix B). You can use not only SVG, but PDF and AI files as the source of vector images for your designs. PostScript and EPS (Encapsulated PostScript) are supported to a lesser extent; you can also import CDR (CorelDRAW) files.

Also, nothing prevents you from importing a bitmap file into Inkscape and either using it as is in your design, or taking advantage of Inkscape's versatile bitmap tracer (18.5.2) to convert it to vector paths.

Even if you don't need clipart, learning is always a good reason to download the SVG source of an image you find interesting. Unlike a bitmap, a vector image contains a lot of information about the way it was created, and in Inkscape you can examine that information in detail. The correspondence between visible areas and objects, the types of these objects, their properties, grouping and layer structures—all of these aspects are very instructive when you're studying Inkscape or SVG techniques.

Inkscape's files are easy to find online, compared to most other vector tools. It shouldn't be surprising given that its native format, SVG, is human-readable and naturally web-friendly (it is directly supported by most recent browsers). Moreover, as open source software, Inkscape promotes a culture of sharing that extends to content as well. There is plenty of SVG on the web already (for example, search *https://images.google.com/* for *filetype:svg face*)—much more than, say, AI or CDR (but still a far cry from PDF).

Apart from searching the web for SVG content, you might try the following resources:

- *FreeSVG.org* and *http://www.openclipart.org/* are community sites with a lot of public domain clipart in SVG format.
- Wikimedia Commons (*http://commons.wikimedia.org/*) contains thousands of SVG images of all sorts, most of them created in Inkscape.
- The official Inkscape Forum (*https://inkscape.org/forums/*) is where Inkscape users help each other as well as share and discuss their creations. There are also Inkscape groups on DeviantArt, Flickr, Reddit, and other social media sites.
- Finally, you're not limited to SVG—Inkscape is a general-purpose vector editor. Just search for "vector clipart" online and you will find tons of it, both free and commercial; most clipart will come in EPS, AI, or PDF formats, which Inkscape can use just fine.

1.4 A Brief History of SVG

Inkscape uses SVG as the format for saving its vector files. What is SVG?

The SVG (Scalable Vector Graphics) standard was born at the height of the XML revolution in the late 1990s. In those days, when the lure of simple yet infinitely expressive XML was fresh, people wanted to create XML vocabularies for *everything*; vector graphics was a natural candidate. In 1998, a working group was formed at W3C, the international consortium that is behind the most commonly used web standards such as HTML, CSS, and XML. The first fruit of their labors, SVG 1.0, appeared in 2001; the most recent official version is 1.1, published in 2003.

From that point on, however, the progress of SVG slowed down. It took many years for proper SVG support to appear in all the major browsers and graphics

applications. Version 1.2, long under development, was eventually dropped in favor of 2.0; as of this writing, SVG 2.0 still has the status of a draft standard. On the other hand, SVG has outlived most of its proprietary competitors (such as Adobe's Flash or Microsoft's XAML and Silverlight), has been firmly established as the default vector format on the internet, and has made inroads into the traditional design and publishing. At the end of the day, being open does count for something!

Stemming from the long and often convoluted history of vector formats, SVG tried hard to do things the right way from the beginning. Inspired by PostScript and PDF (1.5.1.1), it was designed to be free from their limitations. SVG natively supports transparency, gradients, Unicode for text, and many other conveniences that are taken for granted in the 21st century. It also adds unique filter effects (Chapter 17), which are basically raster operations (such as blur) that can be applied to objects without losing vector editability and resolution independence.

Like the once-ubiquitous Flash, SVG also includes quite comprehensive animation features. Version 2.0, still under development, adds more goodies, such as autowrapped (flowed) text, vector effects, WOFF fonts, and more.

All modern browsers support SVG to some (usually sufficient) extent. This means that you can load almost any of your Inkscape SVG files into a browser the same way you would load a JPG or HTML file, and the browser will display it exactly as it looked in Inkscape.

SVG is a large and complex standard, and most existing software has only limited support of it. Inkscape is no exception. Most notably, it cannot do SVG animation or scripting. Some smaller SVG features are also missing; for example, references to resources (such as gradients or symbols) work only within the same document. It is a stated goal of the Inkscape project to eventually support all of SVG.

When you save an Inkscape document, you have a choice of two SVG formats: *Plain SVG* and *Inkscape SVG*. Plain SVG is just that: pure SVG 1.1 code and nothing else. Inkscape SVG, however, adds quite a number of elements and attributes in Inkscape's private namespace.

Don't be afraid of Inkscape SVG! It is perfectly valid and standard-compliant SVG; the goal of these additional elements and attributes is just to provide Inkscape-specific metadata *about* the SVG objects, not to add some incompatible objects of Inkscape's own. Inkscape extensions may affect how objects *behave* when you edit them in Inkscape, but they never affect how the document is *rendered*.[1] Therefore, Inkscape SVG and Plain SVG versions of the same file will look exactly the same in any compliant SVG renderer. The only reason to use Plain SVG may be to reduce the file size or produce a document more suitable for manual editing by an SVG expert.

Inkscape can also save files as *compressed SVG* (both Plain and Inkscape varieties). Compressed files have the extension *.svgz*; unlike SVG, they are not human-readable but take much less space on disk. Most programs will read SVGZ files just as easily as they read SVG.

---

[1] See 15.2.2 for a legacy exception, however.

1.5 Inkscape and Its Competition

Of course, Inkscape is far from being the only game in town. There have existed and still exist dozens of vector editors: commercial and open source, for different platforms, generic and specialized, alive and dead. A few of them deserve to be mentioned here.

At the time of the first edition of this book (2009), all of Inkscape's serious competitors were commercial (and often quite costly) desktop applications for Windows and Mac. Now, there's significant new competition from online editors, partially or completely free (1.5.4); like almost everything else in our lives recently, vector editing has gone online and into the cloud. (Moreover, newer arrivals in this space tend to not emphasize being *vector* editors, even though that's what they are; they are "just" graphics and design tools.) Still, Inkscape's raw power, extensive ecosystem, zero cost, and cross-platform availability continue to give it a significant competitive advantage.

1.5.1 Adobe Illustrator

Adobe Illustrator takes the indisputable first place in this list. Old but continuously developed, it is an immensely powerful and feature-packed application that is usually considered the leader in the field and a de facto standard in vector graphics. Even if you don't use Illustrator, you are likely to run across mentions of its features and versions, comparisons of other programs to it, and of course AI-created vector files in various formats (including SVG).

> **FOR ADOBE ILLUSTRATOR USERS**
>
> I'm not trying to write a comprehensive migration guide for AI users. Still, throughout the book, I will provide the AI equivalents and comparisons for some of Inkscape's features.

No doubt much of Illustrator's clout is channeled from its more famous cousin, Adobe Photoshop. Positioned as parts of the same creative suite, Photoshop and Illustrator share many UI traits and are optimized for working together. However, compared to Photoshop, Illustrator's position in its field is fortunately much less of a monopoly. Even without Inkscape, it still has very serious competition, although AI's prominence has been steadily growing.

Dating from the late 1980s, Illustrator has had a long and winding history. It wasn't always the dominant player in vector graphics. Many of its features were pioneered in competing packages and, sometimes only after many years, reimplemented in AI. By now, however, it's so big—and growing bigger with every version, especially if you consider the rest of Adobe's Creative Suite as well as a whole industry of third-party AI plug-ins—that any generalizations are risky. Illustrator is a lot of things to a lot of people.

Still, I think I can risk one such generalization: whatever its capabilities, few people will claim Illustrator's UI as a paragon of usability. Critics (admittedly mostly users of competing packages) cite a cluttered interface dominated by a

swarm of floating dialogs, too many tools with too narrow functions, limited on-canvas editability of objects' properties, and scarcity of context information. Competing editors also often claim a speed edge over AI.

1.5.1.1 Adobe's Vector Formats

Vector formats associated with Illustrator, and with Adobe in general, play a crucial role in the modern digital world. Even if you're not planning to use anything but SVG with Inkscape, it is useful to have an idea of what PostScript and PDF are, how they are related, and what they are capable of.

Adobe's first claim to fame, back in 1984, was creating the grandfather of all vector graphic formats: *PostScript*. Designed as a standard for sending data to printers, it was ready just in time to play a major role in the "desktop publishing revolution" of the 1980s, driven by accessible personal computers and laser printers.

PostScript was (and is) quite unusual in that it is a complete *programming language* and not just a data format. A PostScript file is actually a program that a printer or computer must run in order to get an image. For example, it may contain an instruction to print a text line, such as "I must not disrupt the class," and a loop that will repeat this line a hundred times. Unfortunately, this also means that, due to an error in the program or someone's malicious intent, a PostScript program might run indefinitely, tying up the system's resources.

On the positive side, PostScript's interpreter used little memory and could therefore be embedded in the hardware of the day. As a result, it became popular with printer makers and soon was the de facto standard for sending files to print.

It was also used as the base for the native file format of the first versions of Adobe Illustrator, which appeared about that time. Even though the AI file format changed in many ways with every version of the application, for a long time its foundation remained the same: an AI file was simply PostScript that followed certain conventions and used PS function libraries from Adobe.

Unfortunately for the users of Illustrator, PostScript's priorities as a highly optimized printer language were not easy to balance with its goal of becoming a general vector graphics medium. For example, Level 2 of PostScript (1991) added device-independent CMYK color, but it wasn't until Level 3 (1997) that such a basic thing as gradients became directly possible. (Until PS 3, applications wishing to create a gradient in PostScript had to "fake" it by overlaying many narrow strips of gradually changing color.) Even the latest version of PostScript hasn't evolved enough to support transparency natively.

This is undoubtedly one of the reasons why early versions of Illustrator were so slow to gain the new features that users demanded (and that competing vector editors were already providing). To this day, Illustrator's UI bears marks of being built upon the PostScript feature set, with everything else treated as an afterthought.

However, the biggest issue with PostScript is not its feature limitations. Over time, its being a programming language proved to be much more of a burden. What was once (in the late 1980s) a clever hack now felt increasingly cumbersome and dangerous. Since any PostScript file is a program, you simply cannot tell what exactly this file will display except by *running* that program. This means you need a complete PostScript interpreter in order to do even the simplest

processing of a PostScript file; you can't even directly combine two PostScript files into a single document with predictable results.

Adobe tried to rectify this by imposing various limitations on PostScript files. One such limitation was the *Encapsulated PostScript (EPS)* format. An EPS file is simply a one-page PostScript document that can be reliably inserted into other documents. However, this was obviously not enough.

So in 1993, Adobe took a more drastic step. It introduced *Portable Document Format (PDF)* which, although based on PostScript, drops the idea of being a programming language. At first, PDF was just simplified PS rewritten in a declarative fashion with added compression and some top-level document management features. Later, Adobe went on to develop PDF well past what was ever available in PostScript; for example, transparency was added in PDF version 1.4 (2001).

Although PDF's stated goal was internet document exchange, it grew in popularity and eventually gained a foothold in print and design as well. The fact that PDF is an open format, standardized by ISO and free for implementation by anyone, also helped. By now, PDF has largely replaced PostScript in most commercial applications, including print.

More important for our discussion, with version 9.0 (2000), Adobe Illustrator switched its native AI file format to one based on PDF instead of on PostScript. This means that any AI file saved in a modern version of Adobe Illustrator is in fact PDF and can be viewed and imported by any software that supports PDF. Inkscape's AI importer is actually the same as its PDF importer.

In summary, at this point in time it really makes no sense to use PostScript or EPS if you can use PDF instead. Inkscape can import PS and EPS, but only by converting them into PDF first, which requires the free cross-platform Ghostscript package (B.4) to be installed on your system. Exporting from Inkscape to PDF gives better results than exporting to PostScript.

1.5.2 CorelDRAW

One of Illustrator's biggest rivals is CorelDRAW. Like Illustrator, it is a large, full-featured application and is part of a suite of graphic applications. Here, however, the similarities end.

CorelDRAW has always positioned itself as a vector editor "for the rest of us." Priced lower than Illustrator or Freehand (the main contender of Illustrator back in early 1990s, now bought out by Adobe and discontinued), CorelDRAW has emphasized ease of use and, during the 1990s, greatly expanded the audience of vector editing tools. In some countries and communities, it was, and still is, more popular than Illustrator.

CorelDRAW pioneered some valuable UI concepts that were passed down to a number of other applications, including Inkscape. For example, the single Selector tool that can do all kinds of selections and transformations (click to select, move and scale, then click again so you can rotate and skew) first appeared in CorelDRAW; Illustrator and Freehand have separate "select," "scale," and "rotate" tools instead.

It has also introduced the notion of shapes (such as rectangles or ellipses) as separate object types, with the path editing tool acting differently on the shapes than on plain paths. For a long time, Illustrator had no concept of shapes; it had a tool to *draw* rectangles, but any new rectangle became just a rectangular path, with no rectangle-specific properties to edit.

1.5.3 Xara

Xara (its current iteration is called Xara Designer Pro) has always been a minor contender in the race of vector editors, but it is historically important. It was the first vector editor to have complete onscreen anti-aliasing, on-canvas editing of gradients with convenient handles, convenient transparency support, and a context-sensitive panel with controls relevant to the current tool. On top of that, Xara has always been impressively fast, which was especially important with the hardware of the 1990s and 2000s.

In general, Xara had followed the CorelDRAW UI paradigm but greatly improved on it. For a long time, Xara maintained a sizeable and very loyal user base. It's no wonder Inkscape has borrowed a lot of ideas and approaches from Xara. Gradually, however, Xara's novelty faded while its competitors caught up; by the mid-2000s, Xara was perceived by most people as a "nice little app"—very solid and usable, but somewhat passé.

In 2005, Xara Ltd. released as open source—and ported to Linux—a version of its vector graphic editor, called Xara LX. One of the reasons for the move was the rapid progress of Inkscape—although much younger than Xara, Inkscape already had some unique features. On the other hand, Inkscape developers had always acknowledged Xara, with its consistent interface design and excellent usability, as one if its role models. Apparently, Xara wanted to tap into the open source talent pool to revitalize its product.

However, after the initial spurt of activity, Xara LX failed to attract significant attention from open source developers, largely because Xara Ltd. refused to open-source one crucial part of the code—the renderer. Now at version 0.7, Xara LX remains usable, but its development has stalled.

[1.1] 1.5.4 Online Editors

One distinct class of vector editors that has matured since the first edition of this book is online vector editors—those that work in your browser and require no software installation on your computer. Such online tools are naturally limited in performance and features, compared to desktop applications. At the same time, many users don't even need that much power: all they need a quick draft, sketch, or mock-up with, say, a photo plus some text lines and simple shapes. For such users, the convenience of an online editor is attractive.

This is the case especially when such an online editor belongs to a company providing print or other design services. Whenever you want your customers to submit their designs, giving them even a primitive online editor is much better than forcing them to work in a complex piece of software that most would have little need for otherwise (even if that software is free, as is Inkscape). Users who

feel limited by the online editor will still be able to use their complex offline designs by uploading them as bitmaps.

One of the most capable online editors is Canva. It does provide print services, but you can also download your Canva-made design as a bitmap or PDF (but not SVG) file and do with it what you please. Canva's attraction is not so much in its editor (though it's adequate) as in its online community and the wealth of fonts, graphics, and reusable templates that it provides. Another good online editor (with SVG export) is Gravit; both it and Canva provide a basic version of their online editors for free but charge for advanced versions with additional features.

On the open source side of things, a number of projects provide an embeddable vector editor library that you can add to your site. Two notable projects are SVG-Edit and Fabric.js, both of which support SVG natively. Using such a library requires serious JavaScript programming skills, but the payoff of having a decent editor for simple graphics right on your own website may be well worth the trouble.

1.5.5 . . . and Inkscape

Compared to its commercial competition, Inkscape still scores low in several respects, such as rendering performance and support of advanced print technologies. On the other hand, Inkscape has some innovative editing features that are pretty much unique in its field, such as the clone tiler (16.6), 3D Box tool (11.3), path effects (Chapter 12), and parts of the Tweak tool (8.9).

The most obvious reason why some features are not yet implemented in Inkscape is that they are missing in SVG. For example, Inkscape implements SVG's linear and elliptic gradients (10.1) and even gradient meshes (10.7)—but it lacks other gradient types, such as conical supported by Xara. The same applies to multiple-page documents: this capability is simply not provided for in the SVG standard at this time (though you can, to some extent, emulate pages via layers).

On the other hand, SVG's limitations only matter for what kinds of objects or properties Inkscape can have. In its ways of manipulating these objects and combining them into higher-level objects, Inkscape is entirely free to innovate—and it has made good use of this freedom.

In interface and usability, Inkscape is often acknowledged to be easy to learn and work with. Inkscape's UI is comparable, and in fact quite similar, to that of Xara. Some things are done more conveniently in one of these programs than the other, but both feature an unobtrusive, streamlined interface with a lot of easily accessible power.

As a true open source application, Inkscape is coded mostly by geeks, and geeks' interests and priorities tend to differ from those of commercial developers. For one thing, geeks hate dumbed-down interfaces and enjoy powerful controls and unlimited tweaking. They especially appreciate rich keyboard control, and Inkscape has an unprecedented number of keyboard shortcuts covering most of its modes, tools, commands, and features (3.4).

Not any less important than features or usability are an application's stability and speed. Inkscape's released versions may crash or freeze occasionally (save

often!), but generally its stability is not a concern for most users. Speed used to be a major complaint in older versions of the program—but recently, with a multithreaded renderer, Inkscape's display speed has improved drastically.

1.6 The Life of an Open Source Application

As you can see from this discussion of Inkscape's competition, it is rather unusual for a serious vector editor to be an open source application. What helped Inkscape succeed where other open source attempts (and there were many) failed? Let's take a brief look at Inkscape's history.

The story of Inkscape begins almost simultaneously with that of SVG, though that first chunk of code wasn't yet called Inkscape. It was called *Gill* and was the creation of a single person, Raph Levien, who in early 1999 hacked away at a simple viewer and editor for the new vector format, then under discussion by the W3C. It was a typical one-man project, and it suffered the fate of many one-man projects—its author soon lost interest in it.

But the Gill code was open, and another person picked up the development. Lauris Kaplinski renamed the project to *Sodipodi* and set a much more ambitious goal for it: to develop a real vector editor with most of the bells and whistles of this genre, with the UI largely modeled on CorelDRAW but also influenced by GIMP, a well-known open source bitmap editor.

The first releases of Sodipodi were done by Lauris alone and attracted little attention. By 2002, however, more and more people were discovering Sodipodi, and patches from other developers started flowing in. Yet Lauris remained the sole maintainer of the application; he decided which patches would go in and when the releases would be made. Few people other than him had commit rights to the project's code repository.

Lauris did a great amount of work on Sodipodi; to this day, a sizeable portion of Inkscape's code bears his copyright. However, with time, his autocratic way of managing the project was becoming more and more of an obstacle. Patches being ignored for a long time and disagreements erupting over the development direction weren't healthy.

In October 2003, things came to a head. A group of dissatisfied developers, after failing to reach an agreement with Lauris, declared a fork. They took the latest Sodipodi codebase, added their patches, and coined a cool name for the new project: Inkscape.[2] For a short while, Inkscape and Sodipodi were being developed in parallel, but the parent project soon entered hibernation, unable to keep up with its energetic offspring.

In contrast with Sodipodi, the Inkscape project is very open. No single person has the supreme say on how things are to be done. To be granted full developer rights, including the right to commit directly to the code repository, you need to submit just two successful patches. Especially in Inkscape's early days, very little discussion was taking place at all; if someone took active interest in some

[2] If you're wondering why it's legal to take someone's copyrighted code and start changing and renaming it, read the copy of the General Public License (the file *COPYING*) that comes with your copy of Inkscape.

aspect of the program, others just assumed that that person knew what he or she was doing and didn't interfere.

Over time, a number of guiding principles have emerged, and now it is not uncommon to see a contribution rejected if it contradicts the developers' idea of how things should work. However, there's still no top authority figure; everything is being decided by discussion and consensus. This means that anyone who's passionate enough to persuade others *and* can code efficiently still has a good chance of influencing the overall direction of the project.

Among the guiding principles of the project there's a simple rule: *listen to the users.* This sounds commonplace, but it is surprising how few software projects, be they commercial or open source, actually follow this. In Inkscape, it really helps that many of the developers are themselves active users of the program. Also, in its early days, Inkscape had to win over Sodipodi's users in order to survive at all—but the tradition of taking user feedback seriously has continued after Sodipodi stopped being a competitor (it's not like Inkscape suffers a shortage of other competitors, though).

Another rule of the project is *patch first, discuss later.* That is, if you have an idea and can code, don't try to talk others into implementing it; do it yourself and let others test it live in the program. If it's any good, your fellow developers will help you make it perfect. (If it's not, you'll just have to back it out.)

One consequence of the "patch first" rule is the policy that development builds must always be workable; any regressions are noted and fixed as early as possible. A community of enthusiastic users downloads the most up-to-date code daily and tests it, long before an official release is even on the horizon. Developers encourage this constant testing and facilitate it by providing new compiled binaries for all major platforms every day. The result has been steady growth of the project—which is now widely acknowledged as one of the most important open source projects and a strong competitor in its software domain.

2

AN INKSCAPE PRIMER

If you have no previous Inkscape experience and little to no experience with other vector editors, this introductory chapter is for you. These are your first baby steps in the new world. If you have already tried working in Inkscape and feel like you can re-create this chapter's final example (Figure 2-26 on page 35) on your own, feel free to skip to the next chapter.

2.1 Installing Inkscape

The first question to ask yourself before installing Inkscape is whether you want a *stable version* or a *development build*. Stable versions have been officially released and have version numbers associated with them. Calling a version stable does not mean it's perfect and never crashes; it just means this version has received a decent amount of testing, is more completely documented, and is what most people use. In fact, apart from its stability, the biggest advantage of running a stable version is that this is what other users run, so fellow Inkscape artists are more likely to be able to help you.

Overall, I recommend that you start with a stable version, but consider upgrading to a development build if you find that you like Inkscape and want to support its development, or if you need the new features added since the release (visit the next version's Release Notes on *http://wiki.inkscape.org/* to see what's being worked on).

Using a *development build* has its own advantages. First of all, such builds have all the latest and best features, which may be significant compared to the stable version (especially if the stable release is more than a few months old). These new features often include fixes for the important bugs of the stable version. Also, by running the development version, you help the development community to discover and iron out new bugs. Naturally, active developers are more interested in the development version, so you're more likely to receive prompt help directly from the developers.

But what about stability? Granted, development builds are, on average, buggier and crash more often. However, if you run into a new bug that's driving you crazy—and if that bug is not fixed quickly enough—you can always go back to the latest stable version without this bug. (Remember to save often, no matter which version you use!)

NOTE *For many years, Inkscape's stable releases have been numbered below 1.0, and each release incremented the version number by just 0.01 (for example, 0.40 was followed by 0.41, then 0.42, and so on). Often, a 0.xx stable release is followed by 0.xx.x bugfix releases (for example, 0.92 was followed by 0.92.1, then 0.92.2, 0.92.3, 0.92.4), which add no new features but fix some bugs. In addition, major releases are preceded by alpha and beta versions, for example 1.0beta1. You can always find out the exact version number, revision, and build date of your installation of Inkscape by opening* Help ▸ About *(look at the top-right corner of the* About *window), or by running Inkscape from the command line with the* `--version` *parameter (not on Windows, Appendix C).*

On the Inkscape website (*https://inkscape.org/*), choose **Download** from the menu and make your choice of the current (stable) version or the latest development version. Then, choose a version for your operating system.

Windows

You have a choice of an installer in *.exe* or *.msi* formats. For the *.exe*, just run it; for the *.msi*, right-click and choose **Install**. Then follow the prompts to choose the language, folder, and other options. At the end you will get a clickable icon of Inkscape on your desktop and in the Start menu. You can also get Inkscape on Windows from the Windows App Store.

NOTE *Development or alpha/beta versions for Windows are archive files with the extension .7z. Download the free unarchiver at* http://www.7-zip.org. *With such a build, all you need to do is unarchive the file into some folder and it will run nicely from there.*

Linux

There are several options. If all you want is a stable version, most Linux distributions already include one; just select it in your software installation application (for example, on Ubuntu, launch Ubuntu Software Center or simply run `apt-get install inkscape` from the command line).

This version, however, may be quite old. If you want a newer stable version or the development bleeding edge version, Inkscape.org offers a number of downloadable packages that you can use with Linux package managers.

macOS

For the Mac, you will download a disk image (*.dmg*) file. You can also install Inkscape via MacPorts.

New development versions are made available fairly regularly, normally every day. If, however, you want absolutely up-to-the-minute Inkscape, you can get the very latest Inkscape code from GitLab and compile it (it's generally easier to set up development on Linux than on Windows or Mac). This requires an above average level of computer savviness, so we won't discuss it here.

2.2 Inkscape's "Hello, World!"

Learning a new programming language traditionally starts with a "Hello, World!" example. This is a minimal, yet real and working, program that simply outputs the string "Hello, World!" somewhere and then quits. I'll now show you a minimal yet real Inkscape editing session that includes starting the program, creating some objects, editing them, and saving the result.

Starting Inkscape is no different from starting any other program. Depending on your OS and personal preferences, you can click the icon, select it from a menu, or type `inkscape` on the command line.

When first run, Inkscape starts by showing a helpful welcome dialog (Figure 2-1). Its first tab, Quick Setup, lets you tweak the program's appearance and behavior: the theme for the UI and icons (all themes include a Dark variant), the active set of keyboard shortcuts, and the look of the canvas. On the Time to Draw tab of this dialog, you can select one of the recently edited documents (if any) or one of the new document templates, categorized into Print, Screen, Video, Social, and Others (3.2). Also on this tab, you can uncheck Show this every time to avoid seeing the dialog in the future.

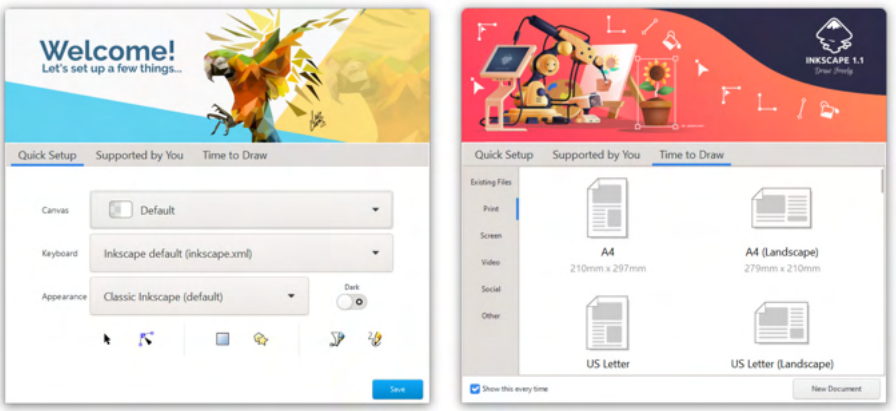


Figure 2-1: Inkscape welcomes you.

After you dismiss the welcome dialog (by choosing a document or pressing Esc), Inkscape opens its main editing window. Figure 2-2 shows the default Inkscape interface.

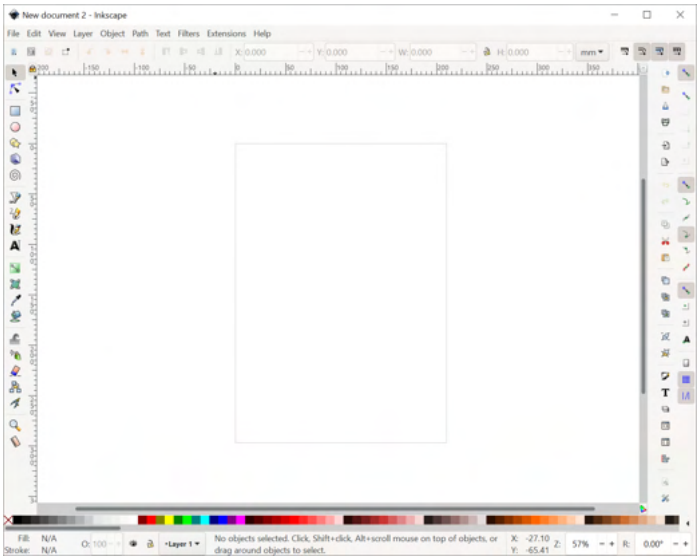


Figure 2-2: The first look at Inkscape

Inkscape's window displays a white work area, called the *canvas*, in the middle and a number of tools and controls at the edges. The canvas contains a new, empty, document Inkscape has created for you, so you don't need to select File ▸ New. You can start working on it right away.

In the column of icons on the left, click the icon with the blue square. This is the Rectangle tool that lets you create and edit rectangles. Now, click the mouse button anywhere on the canvas and drag (or tap and drag if you're using a tablet). A blue rectangle will appear; when you release the mouse, the rectangle will be created as a new *object* in your document (Figure 2-3).

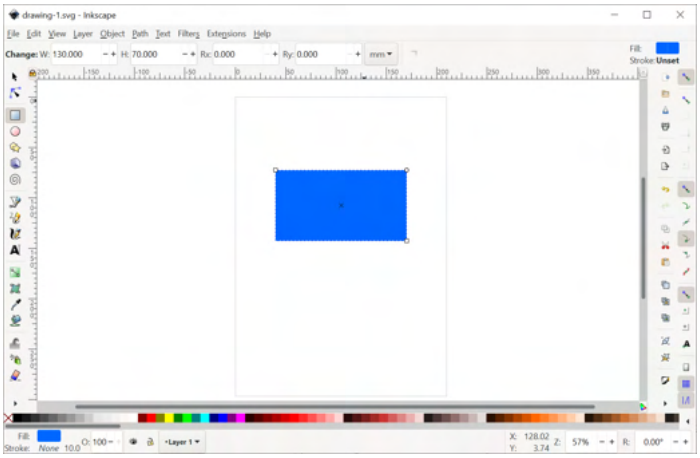


Figure 2-3: We have created a rectangle.

Now, click the **A** button on the left, which will switch you to the Text tool. Click (but do not drag) inside the rectangle. You will see a text cursor blinking where you clicked. Type `Hello, World!`. You have just created a text object, the second object in your document (Figure 2-4).

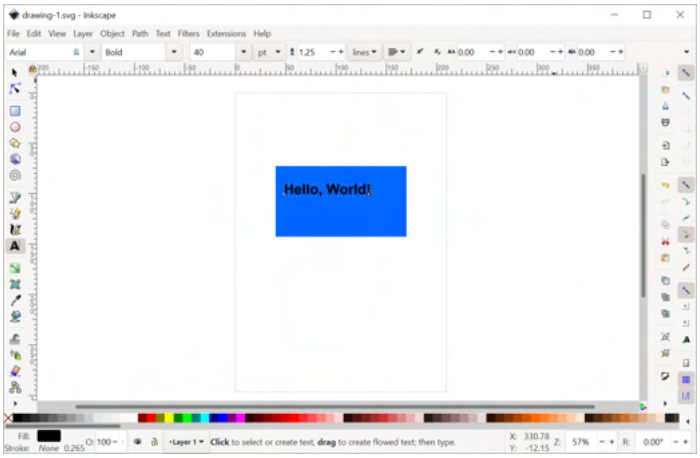


Figure 2-4: We have added some text.

Now, you may have a better idea for the size, position, or color of your objects. Easy fix! In the toolbar on the left, click the topmost button (an arrow). This is the Selector tool. Now you can move any of the objects anywhere on the canvas by dragging with the mouse. To change the color of an object, simply drag that color from the color palette at the bottom of the window and drop it on the object.

Enough tweaking. The document looks perfect. The only problem with it is that it's not saved to a file yet. Just select **File ▸ Save**, navigate to the folder of your choice, and type a filename. That's all. You have just created a new SVG document with graphics and text. Congratulations!

2.3 Interface Overview

Let's take a closer look at Inkscape's interface (Figure 2-5).

It's usually best to maximize Inkscape's window. If you need even more space for work, press F11 to toggle fullscreen mode.

Most of the window is occupied by the *canvas*, surrounded by rulers (top and left) and scroll bars (bottom and right). On the canvas, you can see a rectangular page (A4 paper size by default) with a shadow. This *page frame* defines the edges of your document. However, Inkscape allows you to draw freely anywhere, both on and off the page—the page frame is not a limit, just a hint. This frame matters only when you export into some vector formats and when you view your SVG file in an external viewer; in both cases, you will only see the objects that overlap the page rectangle. You can hide the page frame if you don't care about it.

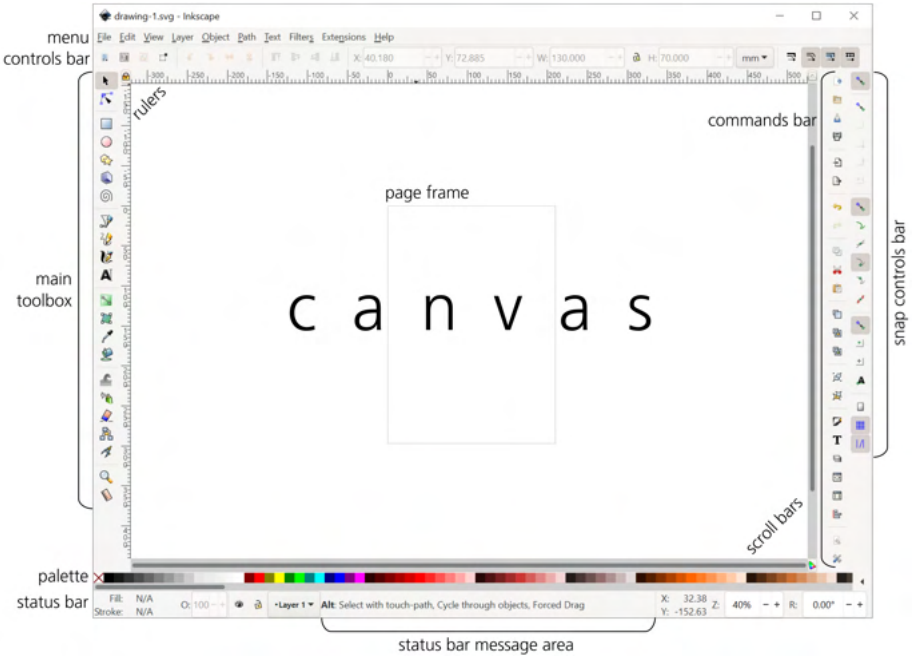


Figure 2-5: Elements of the Inkscape interface

The part of the canvas you see is far from all the canvas there is. In fact, the canvas is so huge, it is almost infinite. You can view more of it by zooming out or by scrolling in any direction, as you will see in the next section. You'll never run out of room in Inkscape! (In fact, you can even get lost in the vastness of canvas; to return to the page frame, just press the 5 key at any time.)

To the left of the canvas, there's the *main toolbox* (vertical). The buttons on this toolbox activate various Inkscape tools. Each tool has its own purpose, features, controls, keyboard shortcuts, mouse cursors, and other stuff. Some tools create new objects, others edit them in various ways, and still others help you navigate in your document. We'll briefly look at some of the tools in this chapter and explore them all in detail in the rest of the book.

Above the canvas is a horizontal toolbar called the *tool controls bar* (or simply the *controls bar*). As its name implies, it contains various options and controls for the currently selected tool. Click some tool buttons on the left and watch the contents of the controls bar change.

At the top of the Inkscape window is a standard menu bar: File, Edit, View, and so on. Throughout the book, we will refer many times to commands and submenus in these menus (for example, Edit ▸ Clone ▸ Unlink Clone). To search through the menus, press the ? key to open the *command palette*. Start typing to filter the list of all currently available commands or press the up arrow in the search field to access your command history (Figure 2-6).

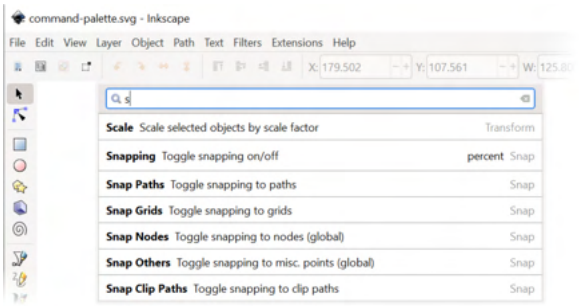


Figure 2-6: The menu and the command palette (after you press ?)

On the right-hand side, there are two more vertical toolbars. The *commands bar* is a regular toolbar of the type you'll find in many programs. It has buttons for actions such as Save, Open, Undo, and so on. The other toolbar, still further right, is the snap controls bar where you can view and toggle any of the many snapping options (7.3).

Below the canvas, there's the *color palette*; it can be scrolled horizontally by a scroll bar of its own. Further below, there's the *status bar*, a bag of stuff whose components we will explore in detail later. For now, take note of the *message area* of the status bar, which always displays some message relevant to what you're doing at the moment. Pay attention to what the messages are saying; it may save you a lot of time and frustration!

Finally, Inkscape has a collection of *dialogs*, each devoted to a specific task. You can place a dialog in a floating window or in one of the two static *docks* (initially empty) on the left and right of the canvas (Figure 2-7).

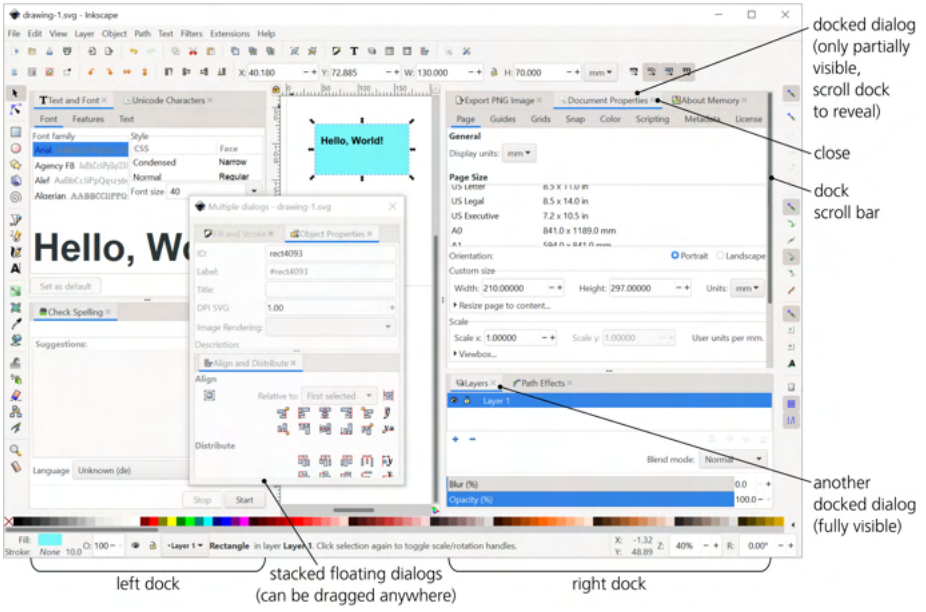


Figure 2-7: Several dialogs in the static docks and in a floating window

Within each window or dock, dialogs can be overlaid as tabs or stacked vertically. Simply drag each dialog's title tab to place it where you need it (a green frame appears where you can drop a dialog). To move a dialog from a dock into a static window of its own, right-click its title and select **Move Tab to New Window** or drag it away from the Inkscape window. Inkscape remembers and restores the location and size of each dialog from the last time it was visible.

To show or hide any of the elements of the interface (except the menu, the canvas, and the docks), use commands in the **View ▸ Show/Hide** submenu. For the dialogs, whether docked or floating, use the convenient F12 key to hide or re-show all of the currently active dialogs.

2.4 Panning and Zooming

With an infinite vector canvas, the tools to move around that canvas as well as zoom in and out for a more convenient view are very important. Inkscape offers plenty of ways to pan (scroll) and zoom—enough to satisfy every taste.

Of course, the canvas scroll bars work just fine to scroll the canvas—but they aren't all that convenient, so I usually hide them by pressing Ctrl-B. Instead, what I use most often is *middle-drag*. Press the middle mouse button anywhere on the canvas and drag it in any direction. Middle-drag works in any tool or mode.

NOTE *Most likely your mouse does not have a middle button as such; instead, try to press (not roll) the wheel that most mice do have (between the left and right buttons). Usually, you can click this wheel just as if it were a button. Of course, rotating the wheel also works to scroll the canvas vertically; rotating it while pressing Shift scrolls horizontally.*

Pretty often, however, my right hand is on the keyboard, not the mouse. In that case, the easiest way to scroll the canvas is by pressing Ctrl-arrows. If you press and hold Ctrl-↑, for example, the canvas starts scrolling downward (that is, your view starts going up), at first slowly but gradually accelerating. When you get the hang of it, it feels very natural.

Zooming in and out is also easy. With your keyboard, just press the plus + or minus – keys to zoom in or out (unlike most other programs, use just plain plus and minus keys, either on the main keyboard or on the keypad; you don't need to press Ctrl or Shift with either). With your mouse, middle-click to zoom in and Shift-middle-click to zoom out. Or, switch to the Zoom tool where you can use regular left click to zoom in, Shift-left-click to zoom out, or drag with the left mouse button to zoom into a rectangular area.

For lots more zooming and panning information and tips, jump to 3.10.

2.5 Creating Objects

Now that you can navigate your way around the canvas, let's have a closer look at the tools available for creating objects.

Looking down the main toolbox, the first object-creating tool you see is the Rectangle tool. This and the next four tools—Ellipse, Star, 3D Box, and Spiral—are collectively called *shape tools*, and the objects they create are called *shapes* (Figure 2-8).

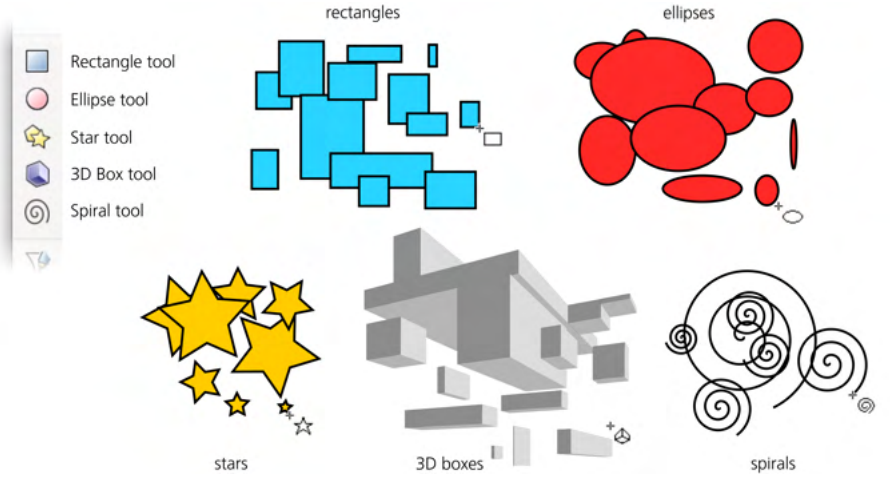


Figure 2-8: Playing with shape tools

A *shape* is a geometric object that you can not only move, scale, or rotate (as you can any other object) but also edit in some ways specific to its type. For example, with a rectangle, you can have its corners rounded; with an ellipse, you can turn it into a segment (pie-slice shape) or elliptic arc; with a 3D box, you can move and resize it in its own 3D space. We will talk about shapes in Chapter 11; for now, choose any of the shape tools and start drawing on the canvas. Each mouse drag, from press to release of the left mouse button, creates a new shape.

Next down in the toolbox are the *Pen* and *Pencil* tools (Figure 2-9). Both create arbitrary paths, but they do it differently. The Pencil tool works just like a real pencil—you draw on the canvas, and it leaves behind a trace. (Note that the trace is not exact; it's a bit smoothed out compared to the actual mouse trajectory.)

Figure 2-9: Using the Pencil and Pen tools

The Pen tool is more complex; it assumes that you understand how paths are composed of nodes connected by linear or curved segments. For now, just click several times with this tool at different points (this creates linear segments between click points), then click and drag a few times, and finally press Enter to finish the path. Both Pencil and Pen by default create paths with a thin black stroke but no fill. We'll discuss these tools in Chapter 14.

The next two tools, *Calligraphic pen* and *Paint bucket*, also create paths, but unlike Pencil and Pen, they by default produce filled paths without stroke (Figure 2-10).

Figure 2-10: Using the Calligraphic pen and Paint bucket tools

The Calligraphic pen is one of the most versatile tools in Inkscape; its many options and parameters allow it to imitate not only a calligraphic pen (which was its original purpose), but also various brushes and many other drawing implements and behaviors (some of which have no counterpart in the real world). This is the primary tool of those who use Inkscape to *simply draw* (Chapter 14).

The Paint bucket tool is also a great help for a cartoonist or illustrator. Just as you'd expect, it fills in any bounded area, creating a filled vector path. For practice, draw around a closed area with several brush strokes of the Calligraphic pen, then click inside with the Paint bucket. (If you click in an unbounded area, the tool will fail to fill it and say so in the status bar.)

One last tool we'll look at is the *Text* tool that creates and edits text objects (Figure 2-11). Creating a new text object is as simple as clicking anywhere and starting to type. Or, you can click and drag, in which case you're creating a rectangular *flowed text* object that will automatically wrap when you type up to the right edge.

Figure 2-11: Using the Text tool

Editing an existing text object is just as easy—click it with the Text tool to position the cursor and use the familiar text editing keys (the arrow keys, Home, End, Delete, Enter for a new line, and so on). The Text tool is the subject of Chapter 15.

Now, what if you need a raster object, such as a photo, in your SVG document? Instead of a tool, use the **Import** command in the **File** menu. Just select any raster image file (JPG, PNG, GIF, and TIFF all work), and it will be inserted into your document. It will, however, remain a raster; for ways to convert it to vector objects, see 18.7.

2.6 Selecting

As you may have noticed, right after you've created a new object (for example, with the Rectangle tool or Calligraphic pen), that object shows a dashed frame around it. This frame indicates that the object is *selected*.

Selection is a fundamental concept in Inkscape. Almost all tools, commands, dialogs, and shortcuts work on the object or objects that are currently selected. Various informational displays, such as the status bar, always describe the selected objects. The ability to select what you need quickly and precisely, as well as to figure out what is currently selected, is vital for effective work in Inkscape.

The main tool that allows you to select objects is called, appropriately enough, *Selector* (Figure 2-12). It is also the first (topmost) tool in the toolbox, because selecting is so important. You can switch to this tool not only by clicking its button but also by pressing either F1, S, or spacebar. You really can't miss this tool!

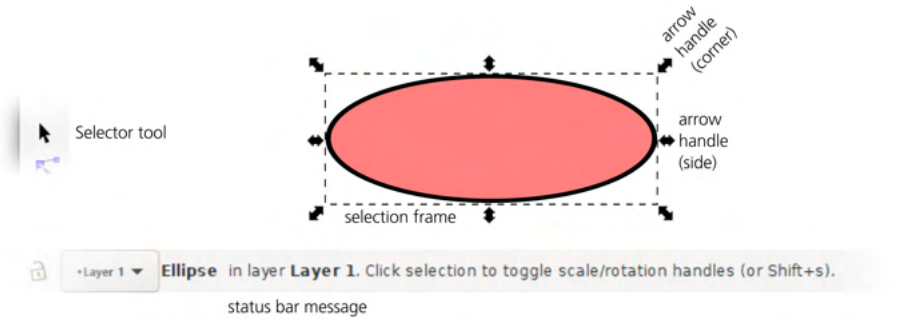


Figure 2-12: The Selector tool and a selected object

Basic selection with Selector is very easy (as is any other basic thing in Inkscape). You just click any object and voilà, it's selected. Try it now. Watch how the status bar message immediately updates to tell you what exactly you have selected: a rectangle, a path, an ellipse, a text object, and so on.

More than one object can be selected at any time, as shown in Figure 2-13. Usually, when you already have something selected, clicking another object deselects the first. However, if you Shift-click an unselected object, you *add* it to the list of selected objects. (If you Shift-click an already selected object, you will *remove* it from selection.) Note what the status bar says about your new selection. Note also how each of the selected objects has its own dashed frame.

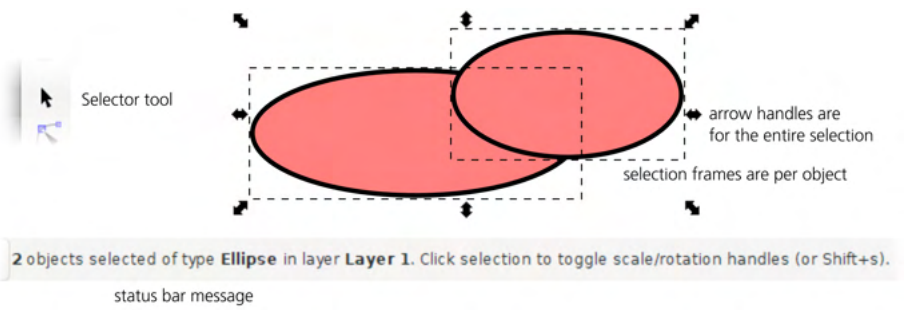


Figure 2-13: Multiple objects selected

Want to select 1,000 objects at once? No need to Shift-click 1,000 times. With Selector, just *drag* (press the mouse button and move the mouse without releasing the button) across all the objects you want to select. You will see a rectangular *rubber band* that follows your mouse. When you release the mouse, all objects that are completely inside the rubber band will become selected, as shown in Figure 2-14.

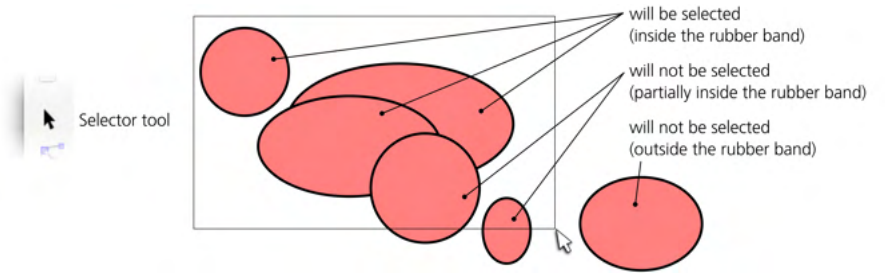


Figure 2-14: Using a rubber band to select multiple objects

You can select by using keyboard shortcuts, too. The Tab key has the generic meaning of "going to the next one"—so in Inkscape, pressing Tab selects the next object (in z-order, 4.4, which, unless you change it, is the same as order of creation), whereas Shift-Tab selects the previous object. The familiar Ctrl-A, which in many programs means *select all*, also works as expected.

Need a different selection? Deselect anything you have selected by pressing Escape or by clicking anywhere on the empty canvas (not on an object).

That's just a fraction of all the selection methods and techniques Inkscape offers, but it should be enough to get you started. Let's move on to actually changing what we have selected.

2.7 Transforming

The other function of the Selector tool, apart from selecting objects, is *transforming* them.

Transforming is simply an umbrella term for *moving*, *scaling* (resizing), *rotating*, and *skewing*. The first two of these—and especially moving—are very common in vector work because it's nearly impossible to create objects that are at once in the exact place and with the exact size you need. You will find yourself moving things around all the time, so learn how to do this quickly and precisely.

NOTE *All four transformation modes treat an object as a whole. By transforming, you can move, squeeze, rotate, or skew the entire object, but you cannot move one part of the object relative to the other parts; that's a task for other tools. For example, if the nose in your drawing of a face is a separate object, you can make it larger on the face using nothing but the Selector tool. But if the nose is part of the same object as the rest of the face, you will need to use other tools (12.5.7.2) to make the nose larger without also changing the face.*

Moving the selection using the Selector tool is simple—just drag any of the selected objects with your mouse. If you have more than one object selected, dragging any of them moves them all together.

One very common operation is *duplicating* objects, or making an exact copy of the selection. To do this, press Ctrl-D; a copy is created and placed exactly over the original selection (which, therefore, apparently does not change). Just drag it away with the Selector tool to where you want to place the copy. You can also use the traditional copy-and-paste method of duplicating objects (Ctrl-C, Ctrl-V).

Other types of transforming are a little more difficult. Whenever you have anything selected, the selection is surrounded by eight arrow marks (Figure 2-15). Dragging any of these arrow marks performs *scaling*. This is how you make your selection larger or smaller, taller or wider, and so on. The four corner arrows can move in any direction to resize the selected objects freely; the four center arrows scale in one direction only (the top and bottom center marks move vertically; the left and right center ones move horizontally).

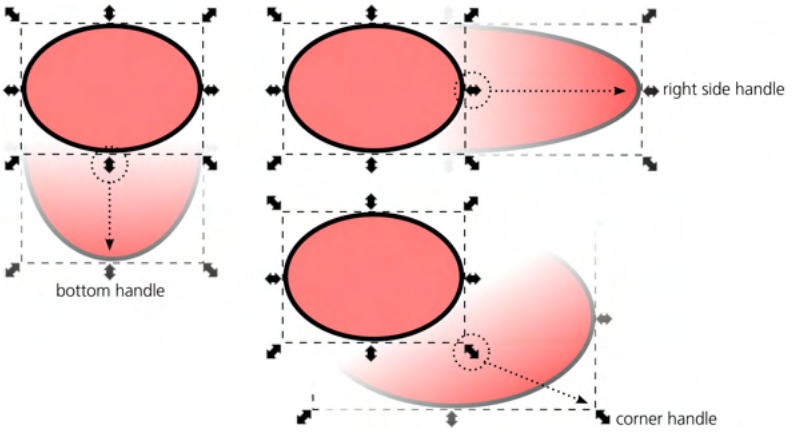


Figure 2-15: Scaling objects

For the remaining two transformation modes, *rotate* and *skew*, click anywhere on an already selected object (or, alternatively, press Shift-S). The arrow marks change—now your selection is in the *rotate mode*, not the *scale mode* we used before (Figure 2-16).

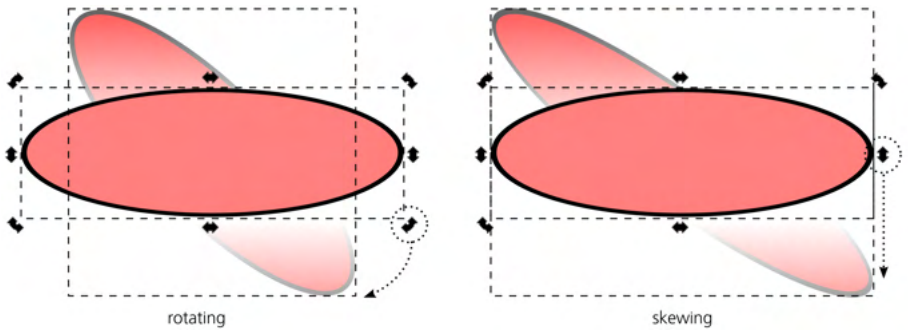


Figure 2-16: Rotating and skewing objects

Now, dragging any of the four corner arrows rotates the selection around its center, while the four center arrows skew the selection. Also, in rotate mode, you can still move your selection around by dragging. To go back to the scale mode, click the selection or press Shift-S again. Next time you select these or other objects, they will again be in the scale mode first.

If you prefer to control Inkscape from the keyboard, you won't be surprised to learn that the four arrow keys each move the selection in the corresponding direction. Less obviously, the left and right angle brackets (< and >) scale the selection down and up, correspondingly, while the square brackets ([and]) rotate it counterclockwise and clockwise, correspondingly. There are no shortcuts for skewing objects.

This should get you started with basic transformations of objects with the Selector tool. Again, there are many more tricks, shortcuts, and rules of thumb that we'll discuss later, but for now, this should provide a general feel for how Inkscape works.

2.8 Styling

Any object in Inkscape has style. *Style* is a complex concept that includes many separate properties, from the color of the fill to the width of the stroke to complex SVG filters that totally alter the way an object looks. Here are just a few of the simplest and most common techniques for changing the style of objects.

First of all, you should have guessed by now that the style-changing actions, as any other actions in Inkscape, apply to the *current selection*. That is, before you change the style of something, you must first select that something.

Once you select the object or objects you want to style, the simplest thing to do is click the color palette at the bottom of the canvas. The color you click becomes the new fill color of all selected objects (Figure 2-17). Even if an object had no fill at all (like a path created with the Pencil tool), it becomes filled with that color too. (Note that you can scroll the palette to the right to see more colors than fit the screen.)

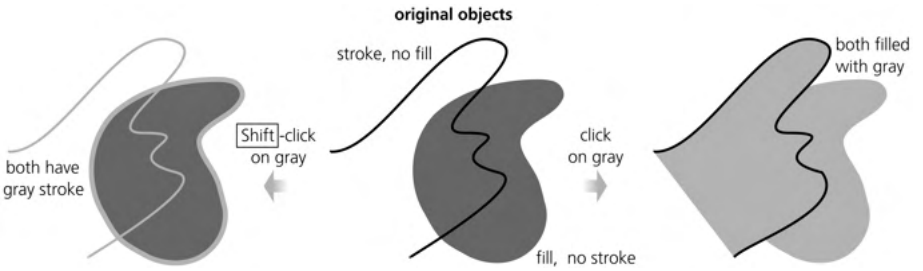


Figure 2-17: Setting fill or stroke from the color palette

You can just as easily change the stroke color of the selected objects—simply Shift-click instead of clicking a color. Again, if some objects had no stroke, they will get it now.

Next, look at the complex control in the lower-left corner of your Inkscape window, on the left end of the status bar. This control (shown in Figure 2-18) is called the *selected style indicator*; in accordance with its name, it indicates the style

(or rather, the most important style properties; there's much more to an object's style than shown in this indicator) of the selection. That's where you look to check the fill and stroke colors of selected objects.

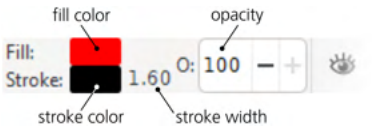


Figure 2-18: Selected style indicator

This indicator also allows you to change the style properties it displays. One very common operation is *removing* the fill or stroke from selected objects. Just middle-click the **Fill** (top) or **Stroke** (bottom) swatch to do this, or right-click the corresponding swatch and choose **Remove** from the pop-up menu. Note that an object with no fill and no stroke is completely invisible—you can't see it and you can't click it to select (although you can drag the rubber band *around* it to select it).

Apart from the fill and stroke colors, one important style property is *opacity*. Think of stained glass that has its own color but still shows through whatever is behind it; this is how objects with less-than-100% opacity behave in Inkscape. (Obviously, an opacity of 0% makes an object invisible.) To change opacity, use the **O:** control in the selected style indicator. Try selecting an object that is on top of other objects and type in a different opacity value; notice how the object becomes semitransparent (Figure 2-19).

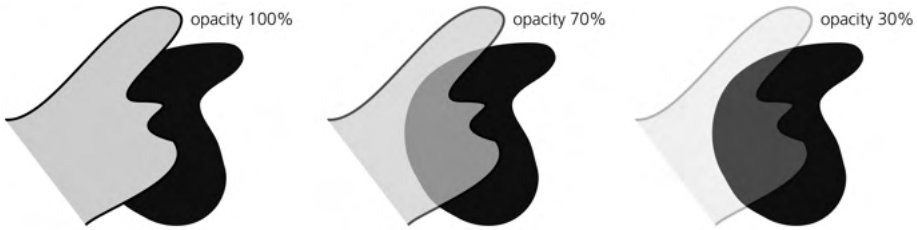


Figure 2-19: Changing opacity

If you just want two objects to have the same style, it's easy to do. Select the first object and press Ctrl-C; this places a copy of the object on the clipboard. Then select the other object (or objects) and paste the *style* of the clipboard object onto them by pressing Shift-Ctrl-V. This is the easiest way to make multiple objects look the same.

2.9 Saving and Exporting

Suppose you've created something in Inkscape that you would like to share with others. What is the best way to go about it?

Saving a document as SVG works as expected. Just press Ctrl-S or choose **File ▸ Save** from the menu, navigate to the folder, type the filename, and click **Save**.

As far as sharing goes, SVG isn't a bad format at all; for example, as already mentioned, you can view it, in all its vector glory, in any modern browser. However, for any number of reasons, SVG may not be a viable choice in your situation. In that case, you need to *export* your document to some other format. Go to the **Save** dialog again. Note the drop-down list displaying *Inkscape SVG (*.svg)*; check out what else is available. Quite a lot, in fact! More than 20 different vector formats are listed.

Don't get overexcited, though. Several of these formats are just variations upon the primary Inkscape SVG format (such as plain SVG or compressed SVG). Others are limited in various ways, either because the target format is itself limited compared to SVG, or because Inkscape's exporter for that format is less than fully developed, or both. These export formats are usable, but only if you know what the limitations are and design your documents with these limitations in mind.

Perhaps the safest export format, besides SVG, is PDF (Appendix B). It still cannot fully preserve some SVG features (such as filters, which are by default rasterized), but for most Inkscape documents, it should be perfectly adequate. The biggest advantage of PDF, apart from its being a vector format, is that it is supported by a much wider range of software than just the canonical Adobe Acrobat.

As you may have noticed, the Save dialog only lists vector formats for export. What about exporting to raster? This is done in a different dialog called Export PNG Image (18.6.1). As its name suggests, the only raster format Inkscape can export is PNG. In most cases, though, it's not a problem; PNG is universally supported (viewable in all browsers), supports full RGB color and gradual (alpha) transparency, and is just a nice all-around all-purpose format perfectly suitable for renditions of any vector artwork. If you really need JPG, GIF, or TIFF, you can use any number of specialized tools for converting your image from PNG.

Open the dialog by pressing Shift-Ctrl-E or choosing **File ▸ Export PNG Image** (Figure 18-22 on page 393). At the top of the dialog, there are four buttons for choosing *what* to export. If you drew your image paying respect to the page frame that now nicely enframes your art, select **Page**. Otherwise, select **Drawing** to make the export area cover all the objects of your drawing no matter where they are. You can also export only the current selection or specify a custom export area.

Then, set either the desired pixel size of your raster image or its resolution (these two values are obviously related: increasing one increases the other, and vice versa). Finally, type the export filename (or use **Export As** to select location and filename) and click **Export**. (After exporting, the dialog remains open so you can use it over and over; this kind of workflow is explored in 18.6.1.)

You're done! Take your saved or exported file and send it out into the world.

2.10 A Final Example

Let's apply what we have learned so far to create a slightly more complex illustration: a silly, *South Park*–style boy's face. This isn't much of a challenge, of course, but it should be useful for digesting all the new knowledge you've acquired.

To start, run a new Inkscape instance, or press Ctrl-N to create a new empty document if you're already running Inkscape.

Choose the Ellipse tool and draw an ellipse (Figure 2-20). That will be the boy's head. Scroll the palette at the bottom, choose a suitable body color, and click it to assign it to the ellipse. (You can try several colors until you find the best one.) If the ellipse came up with a stroke, middle-click the **Stroke** swatch in the lower-left corner of the status bar to remove it.

Figure 2-20: One ellipse

Draw a smaller ellipse on top of the head ellipse. What's going on—is it invisible?! No. Ellipse (like most other tools) simply remembered the color you assigned to the head and now used the same color for the new ellipse, so it appears merged into the background of the larger head ellipse (although you can still see its selection frame and handles). Scroll the palette back to the left and click the white color swatch to see the smaller ellipse (Figure 2-21).

Figure 2-21: Two ellipses

This smaller ellipse is supposed to be the white of the boy's eye. It's probably not exactly the size you want the eye to be and is in a funny place for an eye. Switch to the Selector tool and drag the small ellipse to approximately where you want it. Then drag one of the arrow marks at the corners to make it approximately the right size (Figure 2-22).

Figure 2-22: Positioning the second ellipse

Now you need yet another ellipse for the pupil of the eye. Instead of going back to the Ellipse tool, however, let's use duplication. With the white ellipse still selected, press Ctrl-D. Nothing visibly changed, but you now have two ellipses there, the new one being selected. Click the black color swatch, then drag the arrow marks to make the pupil smaller and drag it into place, as shown in Figure 2-23.

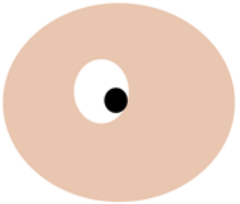


Figure 2-23: Head and one eye

The eye is ready—but you need two identical eyes (Figure 2-24). Let's duplicate the entire eye you just created. The eye consists of two separate objects, so before duplicating you need to select them both. To do this, we either drag a rubber band around the eye (make sure to start dragging from an empty space and not from the head ellipse; otherwise, the head will be dragged, which is not what you want!), or just click the pupil and then Shift-click the white. Press Ctrl-D and drag the second eye sideways (both objects move when any of them is dragged).

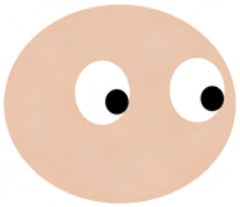


Figure 2-24: Head and two eyes

Now, to finish the drawing, choose the Calligraphic pen tool and draw the nose, mouth, ear, and some hair (Figure 2-25). If the paths appear in some wrong color, click black on the palette; after that, all new calligraphic paths will be black too. If the pen draws too wide or too thin, adjust the **Width** value on the tool's controls bar above the canvas. If something goes wrong, press Ctrl-Z to undo.

Figure 2-25: Welcome to the world!

The image is ready, but we're not finished yet with all the possibilities it offers. What you have done so far would be just as easy to do in a raster editor. But we're in vector—so let's use the specific vector advantages to play with the result. Using Selector, select the whites of both eyes and scale them up or down. Or move the pupils around. Or scale, rotate, and move the mouth. This creates a wide range of facial expressions. Let's select, duplicate, and drag away the entire drawing to present several examples side by side, as shown in Figure 2-26.

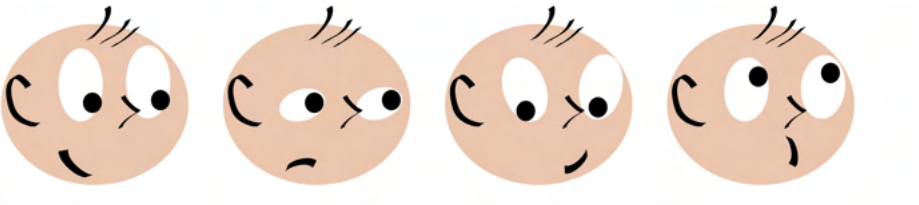


Figure 2-26: Life and its many faces

Save or export the result. That's all!

3

SETTING UP AND MOVING AROUND

After the first introductory chapters, this one will finally get you started in Inkscape in a serious way. From here on, I will explore practical topics at a maximum level of detail, leaving no stone unturned. I cannot know what features or techniques you will find most useful—that depends on the nature of your work and your personal tastes. That's why this book tries to cover *everything* Inkscape has to offer.

In this short chapter, we won't be creating or editing any objects yet. Instead, we'll prepare our workspace, discuss some useful customizations, and learn techniques to navigate documents by zooming and panning.

3.1 Preferences

Inkscape is an extremely configurable piece of software. Throughout this book, I will mention all sorts of Inkscape *preferences* you can change to tweak the program's capabilities and better adapt it to the way you prefer to work. Before you begin using Inkscape for anything of consequence, let's see how and where all the various preferences are set and stored in Inkscape.

All user preferences in Inkscape belong to one of two main classes.

3.1.1 Inkscape Preferences

Inkscape preferences (often called *global preferences* or just *preferences*) control those aspects of the program that do not depend on a particular document you're editing. These include the default behavior of various tools and shortcuts, details of treatment of various object types, display and color management options, and so on.

Most of these options are edited in the Preferences dialog (press Shift-Ctrl-P or select **Edit ▸ Preferences**). Some preferences, however, are set by commands or shortcuts; for example, when you hide or show scroll bars by pressing Ctrl-B (or choosing the equivalent command from the View ▸ Show/Hide submenu), that setting is also remembered as a global preference.

Throughout the book, I will discuss specific global options where they are relevant, referring to the corresponding pages of the Preferences dialog. The tree of pages is in the left pane of the dialog; above it, there is a search field where you can type to search within the entire contents of the dialog. Each option in the dialog has a description; if you need help, just hover your mouse over an option and read the floating tool tip (Figure 3-1).

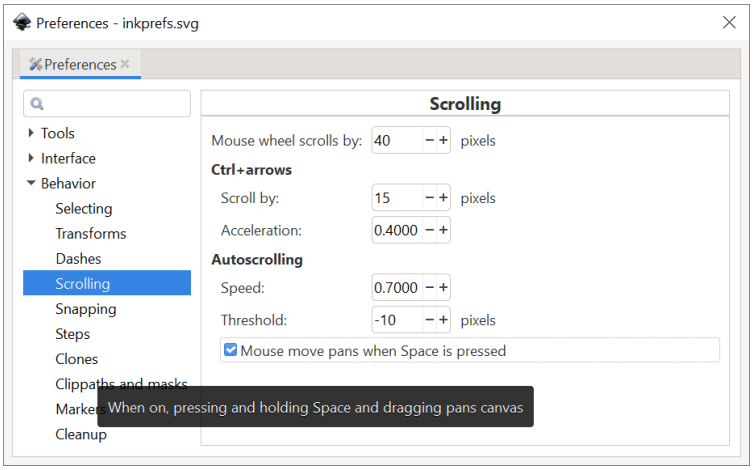


Figure 3-1: A floating tool tip on the Scrolling page of the Preferences dialog explains one of the options.

Note that the dialog has no Save or Apply button. Most changes take effect immediately. A few require you to restart Inkscape, which will be mentioned in the option's description. All global preferences are automatically saved when you exit the program.

The file where the global preferences are stored is called *preferences.xml*; you can view (and change) its location from the Preferences dialog itself, System page.

Some little-used options have no UI; the only way to set them is by manually editing the *preferences.xml* file. This is an XML file with a simple and largely self-explanatory format. You may need to change a specifically named attribute of an element with a certain ID, for example, `printing`. You would then load that

file in a text editor, search for the string `id="printing"`, and edit the value of the specified attribute in that element, for example, by replacing `attribute="old"` with `attribute="new"` (note that attribute values are always in quotes).

3.1.2 Document Properties

Document properties, unlike global preferences, apply to an individual document. This category includes the size and orientation of the canvas, the default measurement unit, various snapping options, status of the snapping grid and guidelines, and so on. Most of these options are set in the Document Properties dialog (Figure 3-2).

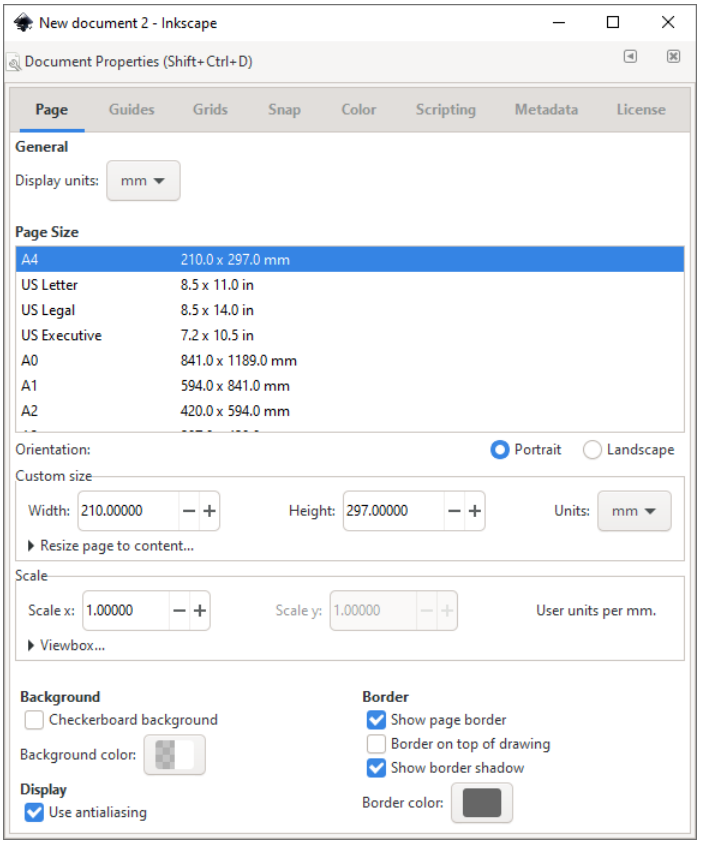


Figure 3-2: The Page tab of the Document Properties dialog

Document properties are automatically saved as part of the Inkscape SVG document for which they are set. For example, if you change snapping modes in one document, Inkscape will remember it the next time you load that document, but other documents will not be affected. To change the document properties for newly created documents, edit the document template used to create those documents, as you'll see in the next section.

3.2 Document Templates

When you run Inkscape, it automatically creates a new document for you. Inkscape's window cannot exist without some document loaded into it, so if you don't give it an existing document to load, it will create a new one for you.

Whenever you need to create an empty document from a window where another document is being edited, just press Ctrl-N or use **File ▸ New**. This will create a document based on the *default template*. However, Inkscape also has a number of other templates that you can use by going to File ▸ New From Template. Just select a template from the list on the left, set its parameters (such as size) if necessary, and click **Create from template** (Figure 3-3).

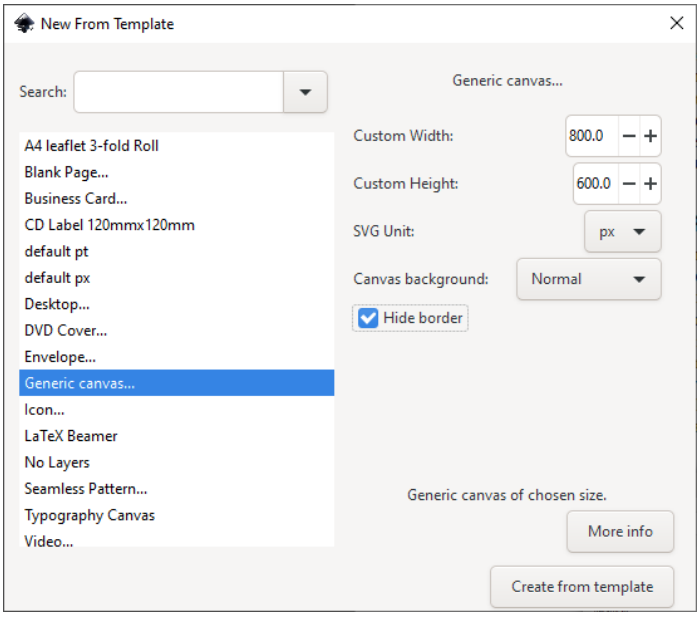


Figure 3-3: The New From Template dialog

Inkscape ships with a collection of templates for standard paper sizes, standard icon and web banner sizes, desktop sizes (for wallpapers), CD/DVD covers, business cards, and so on. A template does not have to be empty—it may contain any objects or document settings that can be useful. For example, look at the Seamless pattern template that has extensive instructions (top left) and a test tiling area (right) for the seamless pattern square in the middle, as shown in Figure 3-4.

In the simplest case (templates without parameters), a template is just a regular Inkscape SVG document; when creating a new document, Inkscape simply makes a copy of such a template. You can add your own templates to the list; any document that is a better starting point than the default empty canvas can be turned into a template. For this, just copy or save the document to the user templates folder you can open from Preferences, System page, by clicking Open next to User templates. You can even override the standard templates if you use the same name for your custom template; in particular, if you save your

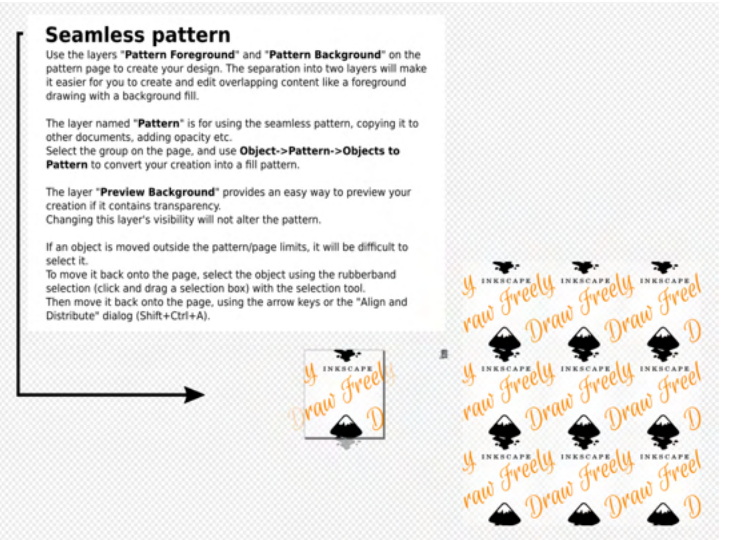


Figure 3-4: Seamless pattern document template. Create your pattern (replacing the sample content) in the middle and watch it being tiled on the right.

template as *default.svg*, it will be used for all new documents created when you run Inkscape or press Ctrl-N.

3.3 Input Device Setup

You can't really use a graphic editor such as Inkscape without some kind of pointing device. What probably *won't* work is any device where you point with your fingers, such as a touchscreen or a laptop touchpad—human fingers are too blunt for precise graphics work. A regular computer mouse works just fine, and most people do all kinds of Inkscape work with nothing but a mouse.

However, if you plan to use Inkscape for artistic drawing (Chapter 14), you may want to get a pressure-sensitive graphics tablet such as those made by Wacom, HUION, or XP-Pen. With it, you get more natural pointing and drawing with a pen and the ability to create variable-width strokes that depend on pressure—just like the darkness of a regular pen or pencil trace depends on how hard you press it on paper.

Inkscape has a dialog for setting up a pressure-sensitive device (Edit ▸ Input Devices). The first edition of this book recommended, for press sensitivity to work, to switch it to the Screen mode for each of the available Devices in this dialog. With modern Inkscape, however, I find that I never need to change anything in that dialog at all: a pressure-sensitive tablet just works.

3.4 Keyboard Setup

Inkscape is unusually rich in keyboard shortcuts. A complete list of all its keyboard and mouse shortcuts contains almost seven hundred entries.

Normally, keyboard shortcuts don't require any setup—they just work. Sometimes, however, you may want to change some shortcuts or add new ones, perhaps because you're used to a different keyboard layout from another program, or simply because your work habits are not adequately covered by the existing shortcuts.

Most (although not all) of Inkscape's keyboard shortcuts can be changed. You can also assign a shortcut to any of the commands that you see in the menus, including any extensions in the Extensions menu (Chapter 19).

To view or edit Inkscape's keyboard shortcuts, go to the **Preferences** dialog (3.1), **Interface ▸ Keyboard** (Figure 3-5). Here, you can redefine shortcuts for any menu command or action as well as (on the Modifiers tab) the keyboard modifiers for the most common mouse actions (canvas panning, selection, movement, and transformation). The shortcuts you cannot (yet) configure include those limited to a single tool or mode.

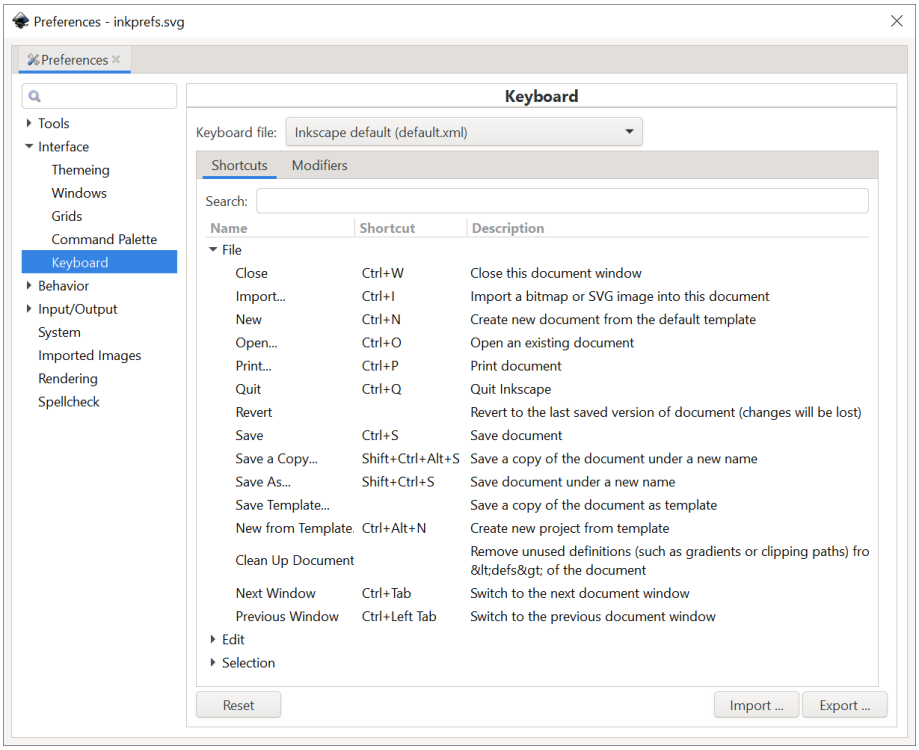


Figure 3-5: Viewing and editing Inkscape's keyboard shortcuts

Before you do any manual tweaking, check the list of available shortcut files at the top. Inkscape offers keyboard layout emulations of all of its major competitors (Adobe Illustrator, CorelDRAW, Xara Xtreme, Macromedia Freehand) as well as for some minor vector editors (Zoner Draw, ACD Systems Canvas). There's also a variant of the standard layout that places the most often used commands

on the left half of the keyboard so you can avoid lifting your right hand from the mouse or tablet. It may be that one of these emulations suits you better than Inkscape's default, so you don't have to edit shortcuts one by one.

Alternative layouts may not be as complete or up to date as the main one. Developers welcome any help in improving these alternative layouts or adding new ones.

The shortcuts in the list are grouped by the menus that have the corresponding commands; you can filter for a command you're interested in using the Search field at the top. To edit or assign a shortcut to a command, click in the middle column; once it changes to New accelerator . . . (accelerator being a fancy word for shortcut), just press the key combination you want and it becomes assigned to this command. Inkscape will warn you about any conflicts (shortcuts already assigned to other commands). If you messed up, use the **Reset** button.

3.5 Page Setup

The first tab of the Document Properties dialog, called Page (see Figure 3-2 on page 39), contains some general settings that you will be changing quite often, so let's look at them now. (Other tabs control grids, guides, and snapping modes; I will describe them in Chapter 7.)

3.5.1 Default Unit

The default unit of measurement for your document—such as px (SVG pixel; see A.6) or mm (millimeters)—will affect all places where you see or can specify lengths or coordinates. For example, the X, Y, W, and H controls of the Selector tool display their values measured with the default unit of the document, although you can change that with unit drop-down to the right of them (see Figure 6-10 on page 100). Similarly, status bar hints (for example, when drawing a path with the Pen tool, 14.1.1) indicate distances in the default unit.

3.5.2 Page Size

The list of standard document sizes is pretty self-explanatory; in the same area, you can specify a custom size and the orientation (portrait or landscape). Inkscape's canvas is virtually limitless (1.1), so you can draw anywhere on it, be it inside or outside the page; however, most standard SVG viewers will display only what is on the page, ignoring anything outside the page boundaries.

Therefore, if you're going to publish or share your artwork in SVG format, you can use the outside-the-page canvas as a clipboard or work area—but then place the finalized artwork onto the page. Printing or exporting to most vector formats also ignores everything outside the page. Bitmap export (18.6), however, is more flexible; you can export the page rectangle, or the actual drawing (all objects), or just the selected objects no matter where they are located.

If you've been drawing without much regard to page borders but now want your drawing to fit precisely within the page boundaries, a convenient way to do this is the Resize page to drawing or selection button in the Document Properties dialog (click **Resize page to content . . .** to expand that area). Clicking this button makes the page border frame the selected objects or (if there's no selection) the entire drawing exactly; you can also specify any of the four margins.

As you resize the page, the objects in your drawing remain fixed relative to the top and left sides, whereas the right and bottom sides move.

At other times, however, you may not care about the size of the page at all. For example, you may be working on a collection of icons or logos, all in the same SVG file, not intending to share the SVG but exporting each icon to a separate bitmap file. In that case, the rectangle of the page is just a distraction. The page border controls at the bottom right of the Document Properties dialog allow you to change the color and shadow of this rectangle or just hide it altogether. You can also place the border on top of the drawing so it remains visible over any objects that might otherwise obscure the edges of the page.

3.5.3 Background

You can also change the background color and background opacity of a document (the default is fully transparent white). Just click the color swatch (lower left) and use the pop-up color selector menu, as shown in Figure 3-6.

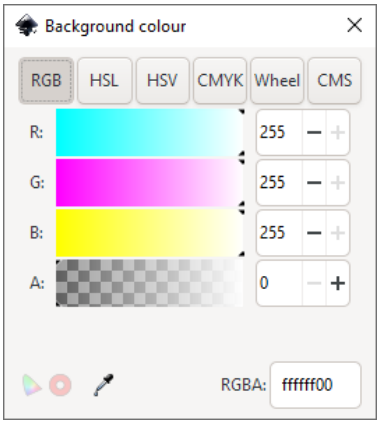


Figure 3-6: Choosing page background color and opacity

Changing the opacity of a white background has no visible effect on your screen, but it will affect the exported PNG files—they will have a transparent background only if you have a transparent background in your SVG document. You can also enable the Checkerboard background option in Document Properties to make it easier to manage transparent objects.

NO DOCUMENT BACKGROUND IN SVG

The background color and opacity will not work in other SVG viewers—they will always show a transparent background. The Inkscape background color setting is limited to display and bitmap export in Inkscape itself. Similarly, the checkered background is purely an Inkscape visual preference and not part of the SVG document. If you want a colored background that will show everywhere, create a colored rectangle underlying the entire drawing.

3.6 Instances, Documents, and Views

An Inkscape window you see on your screen (such as the one on Figure 2-2 on page 20) is not the same as Inkscape itself. A running copy of the program—an *instance*—may have many documents open, each document being in its own editing window. Moreover, you can load the same document into more than one window to have different views of the same drawing. Any changes made to a document in one window are immediately reflected in all windows (within the same instance) where that document is loaded.

If you start with any existing document's window and use the File ▸ Open command (Ctrl-O) from it, you will open a new window with a different document (but within the same instance). If, however, you start from a window with a new and unchanged document (that is, just an empty canvas), it will be replaced by the opened document without opening a new window.

Alternatively, you can start Inkscape from the command line listing the filenames of all documents you want to open (C.3):

```
$ inkscape portrait.svg ../path/to/document.svg another.svg
```

If you want to open a new window with another view of a document you already have open, use the **View ▸ Duplicate Window** command. After that, you will have two windows with two views of the same document, but each with its own independent zoom level, active tool, and selection. Any changes made to the document in one window are immediately visible in the other.

To cycle through all the windows belonging to the same instance of Inkscape (whether they have the same or different documents), press Ctrl-Tab (forward) or Shift-Ctrl-Tab (backward).

For example, in one window you can work in close zoom on small details of your drawing, at the same time viewing the entire document zoomed out in another window, as shown in Figure 3-7.

NOTE *If you just launch the Inkscape program twice, you will get two Inkscape windows, but these windows will belong to different Inkscape instances. Between such windows, automatic synchronization of the views of the same document does not work.*

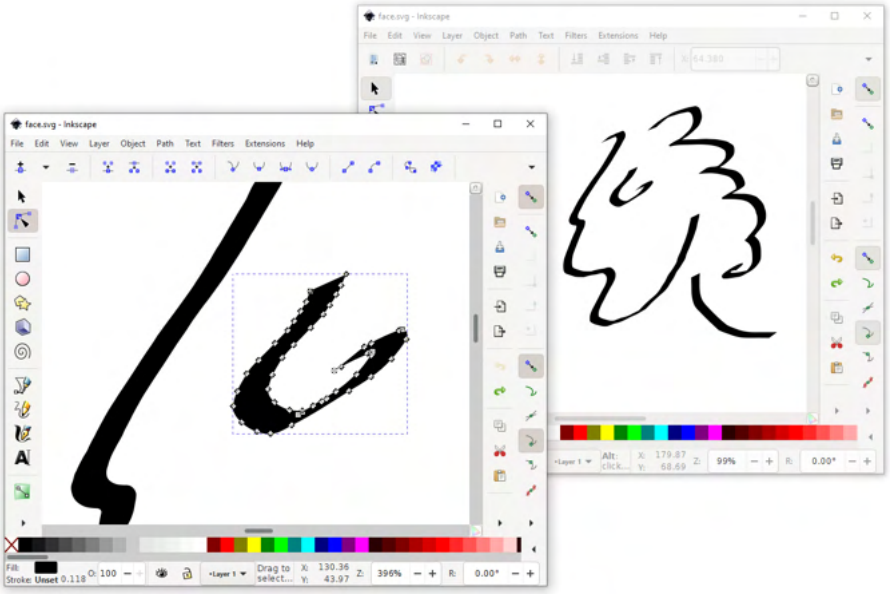


Figure 3-7: Using multiple windows with the same document

3.7 The Document Window

The main components of Inkscape's window (Figure 2-5 on page 22) are, from top to bottom: the menu, the controls bar (whose contents change when you switch tools), the main toolbox on the left and the dialog dock on the right with the document canvas between them, and, at the bottom, the palette and the status bar. By default, the canvas area has rulers (top and left sides) and scroll bars (bottom and right sides).

NOTE *Press Alt-X to access the first editable control in the controls bar; from it, you can navigate to the other controls using Tab or Shift-Tab.*

Using commands in the Show/Hide submenu of the View menu, you can hide all parts of an Inkscape window with the exception of the menu and the canvas itself, as shown in Figure 3-8.

Leave only what you need; the more you hide, the more room is left for your drawing. I usually hide the commands bar, rulers, and scroll bars. Most of the time I also hide the palette, unless I plan to use Inkscape for color-intensive tasks (such as coloring cartoons). All these options are remembered and will be restored when you next start Inkscape.

To switch Inkscape's window to fullscreen mode, press F11 or use **View ▸ Fullscreen**. In that mode, the set of visible interface elements can be entirely different. For example, you may show the toolbar and the tool controls in the regular mode, where you do all the editing, but hide them in the fullscreen mode to view your artwork with minimum distraction.

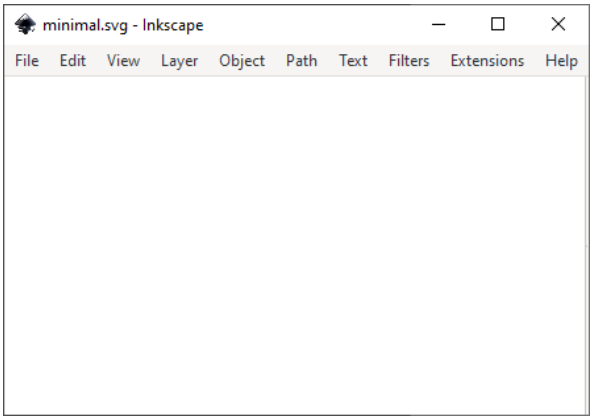


Figure 3-8: A minimal Inkscape window

3.7.1 Window Geometry

The phrase *window geometry* refers to something very simple: the size of an Inkscape window and its position on the screen. Trying to be as helpful as possible, Inkscape remembers the size and position for each document's window and saves those values in the document. So when you next load the document, its window will open with the same size and position it had the last time the document was saved.

In most cases, this is helpful. Sometimes, however, this is more of an annoyance—for example, if you get Inkscape SVG files from someone who has a different screen resolution, the window on your screen may end up in a weird place. To disable this behavior, in the **Preferences** dialog select **Interface ▸ Windows** page and change the **Saving window geometry** option, shown in Figure 3-9.

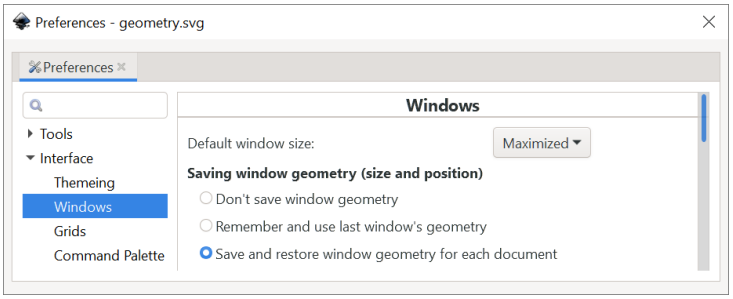


Figure 3-9: The Saving window geometry option

The first choice disables both saving and reading window geometry. The second choice makes window geometry a global rather than per-document preference, so that whenever you change the size or position of any Inkscape window, this geometry is then applied to all subsequently opened windows. The last choice—per-document window geometry—is the default.

Apart from window geometry, the current zoom level and the view area are always saved with the document, so you will be looking at the same place at the same zoom when you next load it into Inkscape.

3.8 Dialogs

Inkscape's user interface contains a number of movable panels, called *dialogs*, each with a specific function. While Inkscape strives to make most editing operations accessible right on the canvas via tools and shortcuts, dialogs still exist and are used quite frequently (Figure 3-10).

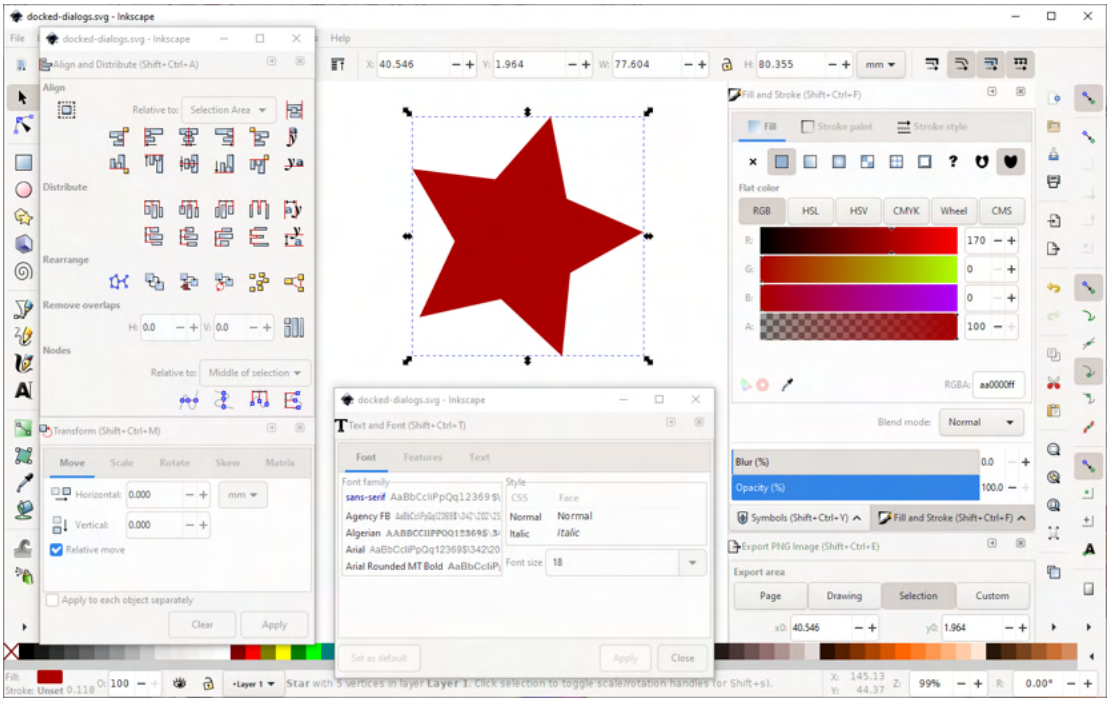


Figure 3-10: Floating and docked dialogs

For each dialog, usually there's a menu command and a keyboard shortcut that begins with Ctrl and Shift (for example, Shift-Ctrl-F opens the Fill and Stroke dialog). A newly opened dialog gets keyboard focus, so you can not only work with it using your mouse, but also navigate it using the Tab key, switch tabs with Ctrl-Page Up and Ctrl-Page Down, type values into fields, and so on.

To close a dialog, click the "X" in the top-right corner, or if the dialog has keyboard focus, press Ctrl-W. To move keyboard focus back to the canvas without closing the dialog, click the canvas or press Escape. If a dialog is already opened, choosing its command or invoking the shortcut again does not close it but transfers keyboard focus to it.

A dockable dialog can be *docked*—placed into the docking area on the right-hand side of the editing window—or it can be *floating* in a small window of its own. (Nondockable dialogs, such as Preferences, are always in separate windows.) Simply drag any dialog to or from the dock; two or more dialogs can even be docked together into a floating dock of their own. Closed, each dialog remembers its status (docked or floating) and position the next time you open it.

In Figure 3-10, one floating window contains two dialogs, another contains one, and the main dock in the editing window contains three more dialogs.

If you don't like the dock, go to the **Windows** page of **Preferences** and change the **Dialog behavior** option from **Dockable** to **Floating**. This will disable the dock and make all dialogs open in a floating state.

3.9 Themes and Icons

Not everyone is pleased by Inkscape's default appearance—which is not surprising given how sensitive its users tend to be to aesthetics and how much time they spend in the program. Fortunately, Inkscape itself is pretty flexible in this regard.

On the Interface ▸ Themeing page of Preferences, you can choose from a number of preinstalled *themes* that affect how all elements of the Inkscape UI look. Perhaps the most practically useful of these are the HighContrast (better contrast of labels for legibility) and HighContrastInverse (same but white-on-black instead of black-on-white) themes (Figure 3-11).

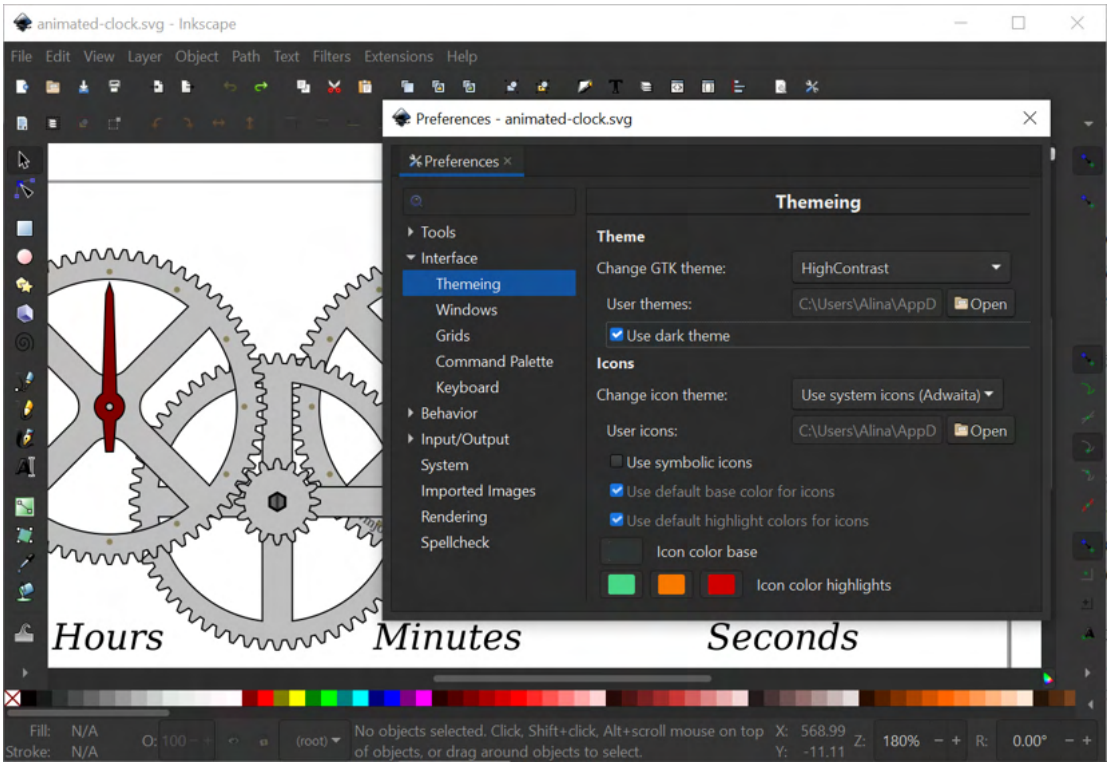


Figure 3-11: Inkscape using the HighContrastInverse theme with multicolor icons

On the same page, you can select one of the alternative icon sets, as well as control the size of the icons in various toolbars. Unfortunately, there's no way to similarly change the font size of the UI labels. You can add new themes and icon sets to the program; search the internet to see if there are any you would like to try.

3.10 Basic Zooming

Zooming (in or out) temporarily magnifies your view on a drawing so you can examine the small details (when zoomed in) or look at the whole picture (when zoomed out). Zooming does not change the drawing itself, only your view of it.

The (almost) infinite scalability of vector graphics makes zooming one of the most common operations in vector work. It's not surprising that Inkscape offers *a lot* of different ways to zoom.

First of all, the current zoom level is always displayed in the Z: editable field on the right end of the status bar (below the canvas), as shown in Figure 3-12. You can always click it, type the desired level of zoom, and press Enter.

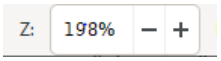


Figure 3-12: Zoom indicator in the status bar

The simplest way to zoom in or out using the keyboard is by pressing the plus sign and minus sign keys (+ and –), correspondingly. The keys on the keypad (on the rightmost end of the keyboard on most desktop computers) work just as well as those on the main keyboard (above the letter keys). What's more, the equal sign (=), usually located on the same physical key as the plus sign, also works for zooming in, and the underscore (_), usually on the same key as the minus sign, works for zooming out; in other words, you never have to press Shift to access the zooming keys.

Zooming with the plus and minus keys works in all tools and modes, except when you are editing text. In text, the plus and minus keys on the main keyboard insert the corresponding characters, but the keypad + and – keys still work for zooming.

Often your hand is on your mouse or tablet pen, so reaching for the keyboard is inconvenient. The simplest way to zoom in by mouse is with the middle button (on most mice, this is not actually a button but a scroll wheel that you can click as well as rotate). The point you click will keep its relative position in the window after the zoom, so by clicking on some small object, you zoom into that object—unlike the plus/minus keys that zoom in/out of the center of the visible area. To zoom out, middle-click with the Shift key pressed.

Additionally, with a scroll wheel mouse, you can rotate the wheel up to zoom in or down to zoom out. By default, you need to press Ctrl for that, because the scroll wheel without modifiers scrolls the document, as in most other programs. There's an option, however, to make it zoom without modifiers and scroll with Ctrl; you can set it in the Scrolling page of the Preferences dialog.

You can adjust the amount of zooming in or out that the plus/minus keys, the middle click, and the single scroll wheel click perform. This value is set in the Behavior ▸ Steps page of the Preferences dialog. The default is 141 percent, which is 100 multiplied by the square root of two, so that two subsequent zoom-in keystrokes zoom you in exactly 200 percent.

3.11 The Zoom Tool

More zooming methods are available in the Zoom tool (the one with the magnifying glass in the toolbar on the left). In my work, I actually never switch to this tool, because all of its commands and modes are also available in *any* tool via various shortcuts. Still, let's look at the Zoom tool for an overview of Inkscape's zooming capabilities.

As you would expect, simple mouse-clicking in the Zoom tool zooms in, and Shift-click zooms out. Sometimes, however, you already know the area that you want to zoom into, and you don't want to go through all the intermediate zoom levels to reach your target zoom. In that case, simply drag with the tool to create a rectangular rubber band around that area—Inkscape will zoom directly into the area when you release the mouse, as Figure 3-13 demonstrates.

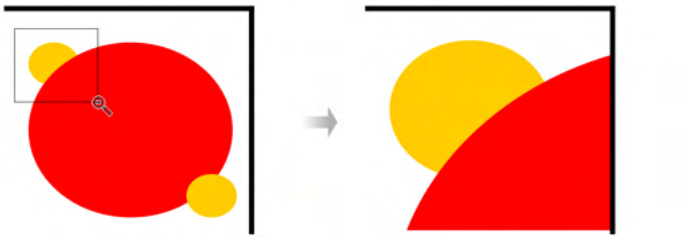


Figure 3-13: Zooming into an area with the Zoom tool

You can also use the zoom-into-an-area trick without switching to the Zoom tool. In any tool or mode, dragging the middle mouse button with Shift starts the same zoom area rubber band and zooms into it when you release the button.

Let's look at the controls bar of the Zoom tool, shown in Figure 3-14.

Figure 3-14: Command buttons on the Zoom tool controls bar

The first two buttons do the same as the plus/minus keys. Next are the three predefined zoom buttons: 100 percent (also via the 1 key), 200 percent (the 2 key), and 50 percent. Then, there is the very useful Zoom to fit selection button (also via the 3 key); it zooms in or out and scrolls the view so that your current selection entirely fits into the window. The next three buttons do the same for the entire drawing (the 4 key), the page (the 5 key), and the page width (the 6 key). You can also zoom to selection temporarily by pressing and holding the Q key; releasing it will return you to previous zoom (if there's no selection, Q temporarily zooms to the page).

Perhaps the most interesting are the last two buttons on the Zoom controls bar. As mentioned above, zooming does not change the document—so you can't undo it with the Undo command. However, there's another way to "undo zooming."

Every time you change your zoom level, Inkscape remembers the previous zoom and the area of the canvas you were looking at. The Previous Zoom and Next Zoom buttons on the Zoom controls bar allow you to travel back and forth in this history of zooms. For example, if after viewing the entire drawing you zoomed into a small area, edited something, and then want to see the entire drawing again, you don't need to press the minus sign key repeatedly. Instead, just press the backquote key (`) and you'll be taken to your previous zoom and view instantly. The same ` key with Shift will then re-zoom you back into the same small area. The zoom history is unlimited—it stores all your zooms from the beginning of your editing session.

Along with the Zoom toolbar and the keyboard shortcuts, all the same commands are also available in the Zoom submenu of the View menu.

3.12 Panning

When you *pan* a document in Inkscape, you shift the visible area inside the window without changing the zoom. It's basically the same as *scrolling*, except that scrolling assumes a single axis (such as up and down), whereas panning can go in any direction. After zooming, panning is the most common action when working in a vector editor.

An Inkscape document window has the traditional scroll bars that will scroll both horizontally and vertically to reach any point on the canvas. Scroll bars, however, are pretty clumsy, so I usually prefer other methods for panning and turn the scroll bars off (Ctrl-B) to free up some extra room for the artwork.

You can pan using the keyboard as well as the mouse. With the keyboard, press Ctrl with the arrow keys to scroll in any of the four directions. This is handier than it sounds, because if you press and hold Ctrl and an arrow key, the canvas movement *accelerates*. This means you can budge the canvas position a little with a single keystroke or swoosh it quickly aside if you press and hold. The speed of keyboard scrolling (the distance for a single keystroke) as well as the acceleration coefficient are set in the Behavior ▸ Scrolling page of the Preferences dialog.

With the mouse, the easiest way to pan the canvas is by dragging it around with the middle button. Again, this works in any tool or mode. There's no dedicated "hand" tool for panning in Inkscape. You can also scroll vertically with the scroll wheel on your mouse and horizontally with Shift-wheel.

Inkscape tries to make panning automatic whenever possible. For example, when you drag selected objects with the Selector tool and push them against the edge of the screen, the canvas will automatically scroll to give way. Also, when you select a new object or path node with the Tab key, Inkscape scrolls to make this object or node visible.

CANVAS PANNING IN AI VS. INKSCAPE

Canvas panning is one of the areas where Inkscape's user interface is noticeably different from that of its biggest competitor, Adobe Illustrator. Many Illustrator users are used to panning by pressing the spacebar and dragging with the left mouse button. In Inkscape, however, pressing the spacebar in any tool temporarily switches you to the Selector tool; hitting the spacebar again switches back to the tool you were using before. Specially for Illustrator refugees, Inkscape offers a compatibility option: with the spacebar pressed, simply moving the mouse (no need to click anything) will drag the canvas. Enabled by default, this option can be disabled in the Behavior ▸ Scrolling page of the Preferences dialog.

3.13 Canvas Orientation

If you've ever watched professional artists (drawers, cartoonists, illustrators) at work, you may have noticed how freely they handle their media. They slant, rotate, even overturn their drawings all the time for better view of their work as well as for a more natural pen swipe. You can easily do this in Inkscape, too.

Simply use your mouse wheel with Ctrl-Shift pressed to rotate the canvas around the cursor point, 15 degrees per notch (to change this value, go to **Behavior ▸ Steps** in **Preferences**). For more precise control, use the **R:** numeric widget at the right end of the status bar, as shown in Figure 3-15.

Figure 3-15: Rotate indicator in the status bar

In View ▸ Canvas Orientation, you can also flip (mirror) the canvas horizontally or vertically; to remove any rotations (but not flips) and return to the default view, use **Reset Rotation** from the same submenu. Rotation does not affect the SVG code of the document and is not saved at all; it is purely for workspace convenience.

3.14 Rendering Modes

By default, Inkscape strives to render each document as closely as possible to how it should be rendered per the SVG standard. However, sometimes you need to *work* with the document, not just look at it, and such complete rendering may become an obstacle. This is when you switch to the *Outline* view mode (View ▸ Display Mode ▸ Outline from the menu). In Outline mode, there are no fills, no transparency, no color, no gradients, no blurring or any other filters. Any object is shown as a thin outline whose width (1 screen pixel) does not depend on zoom, as shown in Figure 3-16.

Figure 3-16: A few objects in the Normal view mode (left) and the Outline view mode (right)

The obvious reason to switch to Outline mode is to make working with complex slow-rendering documents faster. Also, in Outline mode it's much easier to find and access invisible objects (those that are completely transparent or hidden underneath other objects). In this mode, everything is visible, and you can select any object by clicking its outline.

Different colors of outlines in the Outline mode are used to differentiate object types. Regular paths and shapes use black outlines, bitmap objects (Chapter 18) are shown as red-outline rectangles with two diagonals, clipping paths (18.3) are green, and masks (18.3) are blue. Text objects are the only kind of objects that are not outlined; instead, they are shown with black fill. You can also force bitmap images to render even in Outline mode; the corresponding option is on the Imported Images page of the Preferences dialog.

NOTE *You can force Inkscape to start in Outline mode. For that, edit your* preferences.xml *file (see 3.1) by finding the element with* `id="startmode"` *and changing its* `outline` *attribute to* `1`.

The Outline mode is so useful that Inkscape offers two more ways to use it. Instead of switching the entire document into Outline, you can enable the *X-Ray* mode by choosing View ▸ Split Mode ▸ X-Ray from the menu or pressing Alt-6. In this mode, only a circular area around the mouse cursor is rendered in Outline while the rest of the document is shown as usual (Figure 3-17). This way you can examine the structure of a complex drawing piece by piece without losing the context of the finished work. Or you can enable the *Split View* mode (View ▸ Split Mode ▸ Split or Ctrl-6) where the drawing is split into two halves, with a draggable vertical divider, showing normal rendering on the left and outline on the right.

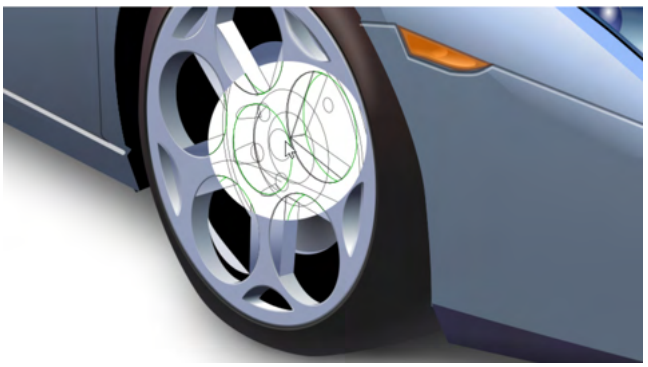


Figure 3-17: X-Ray mode: looking inside a complex illustration

The *Outline Overlay* display mode (View ▸ Display Mode ▸ Outline Overlay) is the same as Outline mode, but it adds a faded-out nonclickable rendering of the document in the background. You can still use the outlines to select objects, but this mode provides a better idea of what those outlines correspond to in the normal view.

Yet another way to speed up rendering is the *No Filters* mode accessible via View ▸ Display Mode ▸ No Filters. By far the slowest-rendering element in most drawing is the filters, such as blur (Chapter 17). By disabling them, this mode may speed up rendering considerably without the drawing becoming unrecognizable. Another rendering mode, *Visible Hairlines* (View ▸ Display Mode ▸ Visible Hairlines), differs from normal rendering only in that it never allows strokes (2.8) to become invisibly thin, no matter how far you zoom out; this may be useful in technical drawings.

Use Ctrl-5 to switch between all the rendering modes (Normal, No Filters, Outline, Visible Hairlines, Outline Overlay) cyclically.

3.14.1 Color Rendering

Just in case you need it, there's a way to render a document in grayscale; use **View ▸ Color Display Mode** from the menu or Alt-5 to toggle.

Also, Inkscape can perform onscreen emulation of the colors of some output device, typically a printer. This is called Color-Managed View (from the View menu). The details of this mode are set up on the Input/Output ▸ Color management page of the Preferences dialog (section Proofing). Basically, what you need is an *ICC profile* of the device you want to emulate, which you can then direct Inkscape to use; for details, see 18.8.

4

OBJECTS

Objects are the bread and butter of Inkscape graphics. Much of the rest of this book is devoted to various types of objects, their properties, and techniques for working with them. However, regardless of type, all Inkscape objects have a lot of things in common.

This and the next two chapters cover the fundamentals of objects and generic object operations in painstaking detail.

4.1 Object Properties

An object is just a thing somewhere on the canvas—a part of your drawing. Of course, it's not always that simple; what looks like a separate object may be either a part of some other object or a combination of several objects. Identifying individual objects takes experience and effort.

Inkscape has a number of dialogs for manipulating properties of objects, but only one of them is generic enough to be called simply Object Properties (accessible by pressing Shift-Ctrl-O, from the right-click menu on the object, or from the Object menu). Select (Chapter 5) a single object to view its properties, as shown in Figure 4-1.

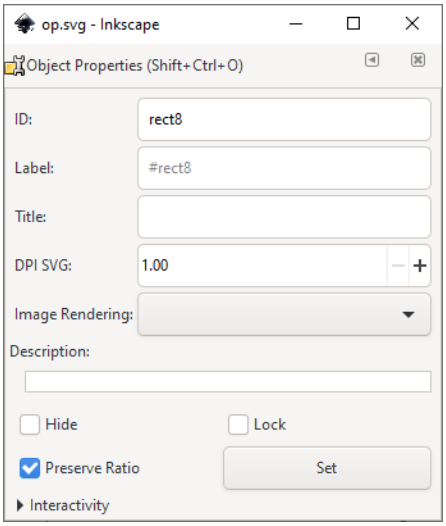


Figure 4-1: The Object Properties dialog

The *identifier* (ID field) of an object is the `id` attribute of the corresponding element in the SVG source of the document (Appendix A). Its value is always unique within the document; Inkscape provides unique IDs for all objects automatically, but you can change this value so long as it remains unique.

Assigning meaningful names to objects is a useful approach to organizing complex artwork. The ID is not the best way to do this, however; per XML rules, you can use only a very limited set of characters in element IDs—no spaces, only Latin letters, digits, hyphens, underscores, and dots. For a more human-friendly alternative, use the object's Label (it corresponds to the `inkscape:label` extension attribute in SVG). It can be of any length, use any characters whatsoever, and need not be unique.

TITLE, DESCRIPTION, INTERACTIVITY ATTRIBUTES

The Title and Description fields may contain additional freeform metadata about the object (these are the standard `title` and `desc` elements of SVG). Since these values are stored in elements, not attributes, they can even contain child elements such as text markup (even though you cannot do this via the Object Properties dialog).

Other fields of the dialog correspond to SVG attributes that mostly make sense outside of Inkscape. The Interactivity section (click to unfold it) allows you to edit the interactivity attributes (`onclick`, `onmouseover`, and so on) that are used in SVG with embedded scripts (usually in JavaScript). Inkscape does not run the scripts, but this can be useful if you want to view your SVG in a JavaScript-enabled viewer.

The Hide and Lock checkboxes control the locked and hidden status of the selected object. A *locked* object is visible, but most selection methods cannot select it, so it cannot be changed. A *hidden* object is both invisible *and* unselectable.

I do not recommend hiding or locking individual objects. You may find yourself in a trap where you need to select an object to unlock or unhide it—but

you cannot because it is hidden or locked. Specifically for such situations, the Object menu contains two commands, Unhide All and Unlock All, that will reveal and make selectable all objects in the current layer.

It's much more convenient to act on the layer containing the objects you want to hide or lock. Hiding and locking via layers is not only faster (affects multiple objects at a time), it's also easier—even if it's locked or hidden, you can always access any layer (via the Layers dialog, see 4.9.4) to change its status.

4.2 Coordinates and Units

Whenever an object is visible on the canvas, you can measure its position and size in the familiar rectangular coordinate system with a horizontal X axis and a vertical Y axis.

In Inkscape, the coordinate origin is always in one of the corners of the page frame (2.3). Older versions of Inkscape placed the coordinate origin in the bottom left corner of the page, with Y values increasing upward. However, the SVG standard places the origin in the top-left corner, with Y values increasing downward (X values always increase to the right), and more recently Inkscape made this the default in its UI as well. You can still switch it to the old (and frankly more natural) bottom-left origin on the Interface page of Preferences (3.1); just remember that this affects only the UI—when editing your document's SVG code (for example, in Inkscape's XML Editor, 4.10), all coordinates assume the origin in the top-left corner.

The rulers at the edge of your canvas area are one way to measure the coordinates (press Ctrl-R to reveal them if they are hidden). As you move your mouse over the canvas, small triangular markers on the rulers reflect its current position. Also, the X and Y coordinates of the mouse pointer are always displayed on the right end of the status bar, just before the zoom field marked Z (Figure 4-2).

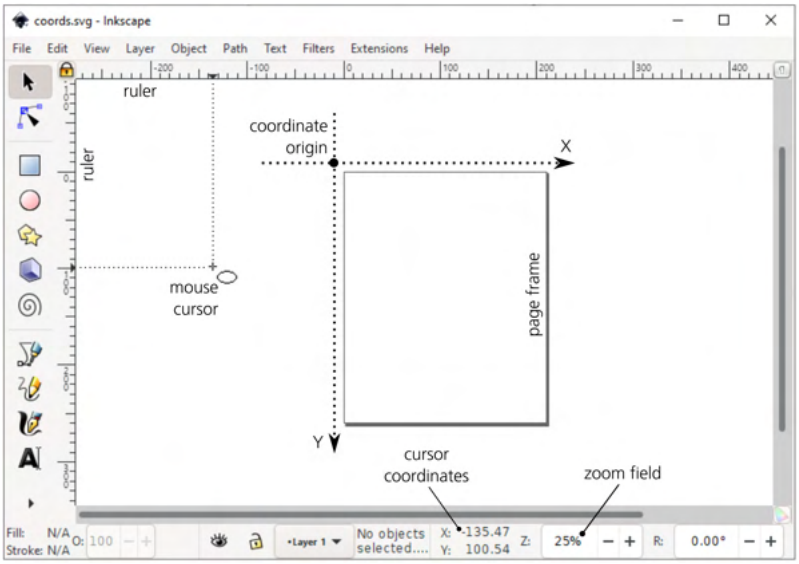


Figure 4-2: Inkscape's coordinate system

The measurement unit used by the rulers is the same as the document unit that you can set in the Document Properties dialog (3.5). To check, hover your mouse cursor over a ruler until a floating tool tip appears.

THE GREAT DPI CHANGE

There's one related issue that you may run into even if you don't care about units at all. In version 0.92, Inkscape changed the conversion factor between the SVG pixel unit (px) and inch (in). This factor has a name of its own: DPI (for Dots Per Inch). To match the changing SVG standard, Inkscape switched from the DPI value of 90 to 96. Whenever you open a file created in an older version of Inkscape, it pops up a dialog offering to "fix" this file by scaling all contents so that the sizes in inches (or any other physical units, such as centimeters, which are interlocked with inches) stay the same. If you don't care about physical units, simply choose the first option in that dialog to leave your document unchanged; otherwise, read the detailed explanation at *https://inkscape.org/learn/faq/#dpi_change*.

4.3 Bounding Box

Let's create an object—for example, draw an ellipse with the Ellipse tool. You will see that the object, initially selected, is framed by a dashed rectangle (Figure 4-3). This rectangle is the visual representation of what is called the *bounding box* of the object—the smallest possible rectangle that completely encloses the object. The bounding box is always upright—that is, its sides are parallel to the coordinate axes; if you rotate an object, its bounding box may or may not change its size, but it won't rotate with the object. (Switch to Selector and rotate the ellipse by pressing the square bracket keys, [and].)

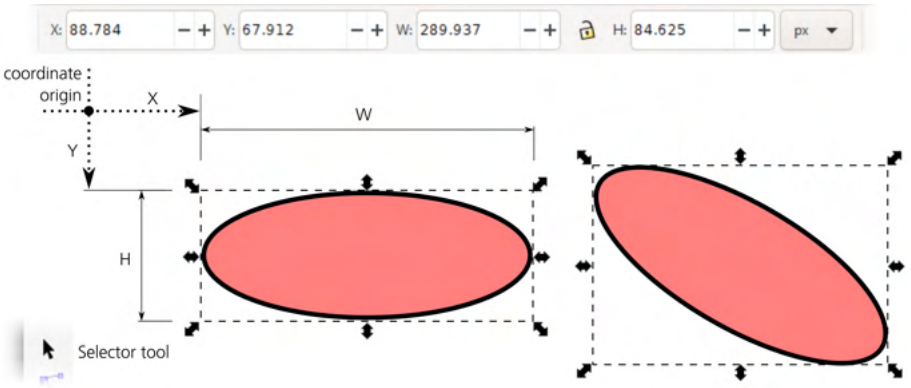


Figure 4-3: The bounding box of an object is always upright.

The coordinates and dimensions (width, height) of the selected object's bounding box are shown by the four editable fields in the controls bar of the Selector tool. The X and Y give the coordinates of the top-left corner of the box (or bottom-left, depending on the coordinate origin preference, 4.2); the W

and H give the width and height of the bounding box. If multiple objects are selected, the values reflect the bounding box of the entire selection. You can change the unit of measurement of these fields in the drop-down menu on the right; initially, it's set to the document unit (3.5.1).

Of course, you can click inside any of these fields and type a new value, which will make the selected object(s) move (for X and Y) or scale (for W and H). If you want the change in width to cause a proportional change in height or vice versa (preserving the aspect ratio), click the lock button between W and H to lock them together.

In Inkscape, bounding boxes of objects can be either *visual* (this is the default) or *geometric*. To switch from one type to the other, go to the **Tools** page of the **Preferences** dialog. The visual bounding box includes *everything that's visible* about the object—most notably its entire stroke (9.1) but also markers (9.5) and filter margins (17.5.4). The geometric box enframes only the *geometric outline* of the object's path.

The geometric bounding box closely corresponds to the object as it is appears in the Outline mode (3.14); for example, the outer fringe of a stroked object falls *outside* of the geometric bounding box. Usually, the geometric bounding box is preferred by those who use Inkscape for technical drawing, while the visual option makes more sense for almost everyone else. Figure 4-3 shows the visual bounding box: note how it fully encloses the black stroke of the ellipses.

4.4 Z-Order

Z-order refers to the order in which objects are drawn on top of each other. An object on top in the z-order may obscure those below it, if they overlap and if the top object is not transparent. The term *z-order* implies a third coordinate axis, Z, imagined to extend perpendicular to the X/Y plane of the drawing toward the viewer. Objects higher in the z-order are thus "closer" to the viewer.

A new object you create is always placed at the very top of the z-order of the current layer (and if you didn't create any other layers, this will be the top of the z-order of the entire document). For example, if you draw several ellipses, each new one will be drawn on top of the previous ones, as shown in Figure 4-4.

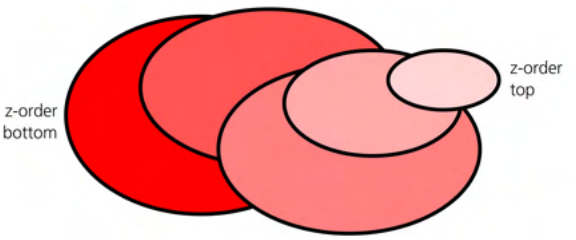


Figure 4-4: Z-order

Selecting, transforming, or style-changing operations never change the z-order of objects. This means you can move, scale, or paint an object while it remains upon its own "floor"—provided you've managed to select that object.

(In 5.9, you'll learn methods for selecting objects that are at the bottom of the z-order and cannot be simply clicked to select.)

Quite often, however, you need to rearrange the stack of objects in your drawing. Inkscape has four basic commands for moving objects up or down in the z-order: two absolute ones and two relative ones. They are used quite often, so you can access them as buttons on the Selector controls bar, via the Object menu, or via simple keyboard shortcuts.

The absolute commands are:

Raise to Top (Home)
: Raises selected objects to the very top of the objects' layer.

Lower to Bottom (End)
: Lowers selected objects to the very bottom of the objects' layer.

The relative commands are:

Raise (Page Up)
: Raises selected objects up one step (past one other object).

Lower (Page Down)
: Lowers selected objects down one step (past one other object).

Figure 4-5 shows an example of how these commands work on the selected ellipse that is in the middle of a stack of rectangles.

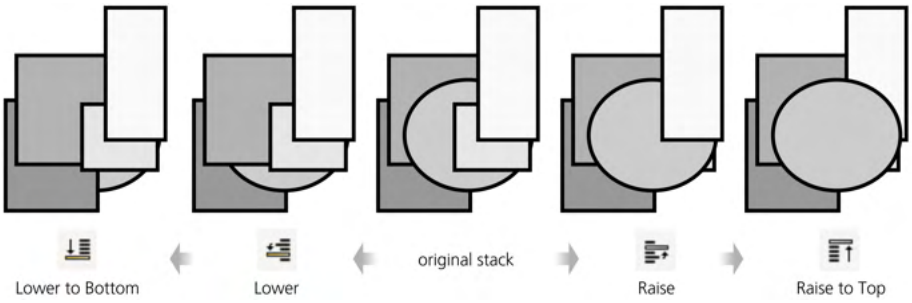


Figure 4-5: Changing the z-order

Note that the relative z-order commands take into account only those objects that *overlap* the selection (more precisely, those objects whose bounding boxes overlap the bounding box of the selection). If your selected object or objects do not overlap any others, the Raise and Lower commands will just move the selection all the way to the top or bottom of the layer, correspondingly.

DOCUMENT ORDER VS. Z-ORDER

In the SVG code of the document, objects that are *higher* in the z-order are those that are closer to the *end* of the document. Thus, new objects are normally appended to the end of the list of existing elements.

Another thing to remember is that all z-order commands work only *within the layer*. If you have several layers in your document, these layers form their own z-order, and an object in a lower layer can never be on top of an object in a layer above. For rearranging the z-order of layers, use the layer commands described in 4.9.2.

It is possible to have objects in different layers or groups selected at the same time. In this case, the z-order commands work on each selected object within *its own* layer or group. For example, it may be that a Raise to Top command will change the z-order of one selected object but will leave the other untouched if it was already the topmost object in its layer.

If you need to change the z-order for a lot of objects, try the Extensions ▸ Arrange ▸ Restack extension. It can rearrange any number of selected objects based on their position or current z-order, as shown in Figure 4-6.

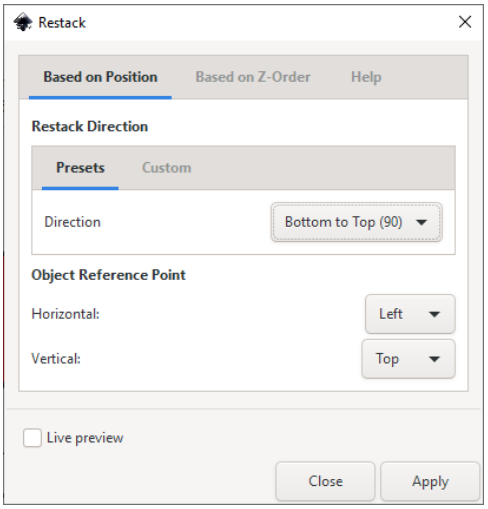


Figure 4-6: Managing z-order with the Restack extension

For example, you can sort objects into a z-order stepladder that goes left to right—so that, of any two selected objects, the one further to the right will be placed higher in the z-order (Figure 4-7).

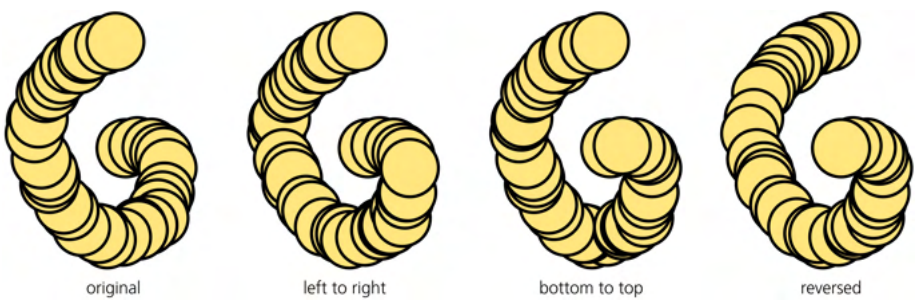


Figure 4-7: Restacking in action; the original scattering of disks (left) was made with the Spray tool (4.7).

The extension command's dialog lets you select the direction of the step-ladder (four cardinal directions, radial inward or outward from the center of selection, or even at a custom angle) as well as indicate the reference point—the point of each selected object that it will use when comparing positions (it can be any of the corners, side mid-points, or the center). Finally, you can reverse the current z-order of selected objects or randomize it.

4.5 Copying, Cutting, Pasting

Inkscape supports the traditional *clipboard operations*: copying, cutting, and pasting.

Copying (Ctrl-C) remembers a copy of the selected object or objects in the program's clipboard; *cutting* (Ctrl-X) does the same as copying but the object is then deleted. Finally, *pasting* (Ctrl-V) places the content of the clipboard back into the document (but retains it in the clipboard as well, so you can paste the same object multiple times).

The Paste command has two variants. The regular Paste (Ctrl-V) places the object at the point on the canvas where your mouse cursor currently is. This is handy for moving an object to an entirely different place in the document; just cut it from where it was, scroll and/or zoom to where you want it to be, hover your mouse over the desired location, and press Ctrl-V. You can even "paint" with copies of an object by moving your mouse around and pressing Ctrl-V repeatedly.

NOTE *Another command that places an object under your mouse cursor is* Import (*Ctrl-I*); *see 18.1.*

Sometimes, however, you want to paste an object exactly where it was copied from. That's what the Paste in Place (Ctrl-Alt-V) command does. For example, you can use it for moving an object from one layer to another without changing its position on the canvas: copy it, switch to the target layer, and paste in place.

With nothing selected, a pasted object is always placed on top of the current layer in z-order. However, if you have anything selected during paste, the newly pasted object gets placed just above the topmost selected object in z-order. You can disable this behavior in Preferences ▸ Behavior ▸ Selecting: Paste on top of selection instead of layer-top.

4.6 Duplicating and Stamping

Sometimes, people use Copy and Paste as a method to get a duplicate of one or several objects. In Inkscape, you don't need that workaround; there's a convenient Duplicate command (Ctrl-D) that creates a copy of your selection and places it in the same position on the canvas. This is equivalent to copying and then pasting in place, except that Duplicate does not change the contents of your clipboard.

All pasting and duplication commands, just as any other methods of creating new objects, place the new object on top of the z-order in the current layer.

NOTE *What if you have selected an object somewhere in the middle of the z-order and want to duplicate it but keep the copy at the same level instead of jumping to the top? This may be a symptom of your document not having enough layers. By placing such an object*

into its own layer, you will ensure that duplicating it will place the copy on top of that object's layer instead of on top of the document.

Yet another method for creating copies of objects is *stamping*. Whenever you transform a selection (move, scale, rotate, or skew) by dragging the mouse in the Selector tool (see 2.7), you can press spacebar to leave behind a copy of the selection without interrupting the interactive transformation. For example, if you grab and drag an ellipse and then press and hold the spacebar while dragging, the object being moved will leave a trail of its copies on the canvas. Stamping also works in the Node tool when you drag a selection of nodes (12.5.7).

All object copying methods discussed so far created new independent objects that are not linked in any way to their originals. If you want a *linked copy* that inherits some properties of the original and updates itself automatically, read about clones in Chapter 16.

4.7 Spray Tool

[1.1] As if all that (copy/pasting, duplicating, stamping) were not enough, Inkscape has an entire tool dedicated to flooding your canvas with objects: the Spray tool (Figure 4-8).

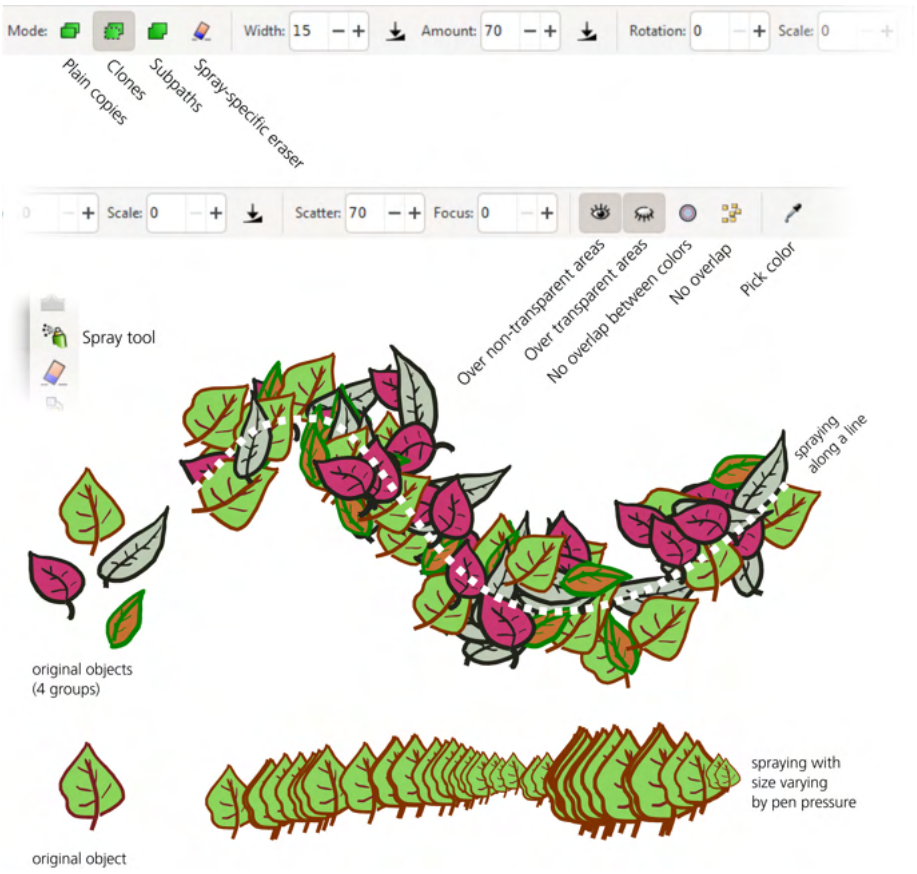


Figure 4-8: The Spray tool will airbrush your drawing with objects.

Its basic operation is simple: select one or more existing objects and drag on canvas to spray around their copies. This upends the traditional vector editing paradigm where you tend to work with individual objects; the Spray tool, as well as the Tweak tool (6.10) with which it is often teamed up, allow you to think in terms of *masses* of objects—clouds of particles, forests of trees, galaxies of stars, schools of fish.

Some of the Spray tool's controls are similar to those of the Calligraphic pen (14.2). To start, set the Width of the circular area you will be painting, using relative zoom-independent units; the maximum width of 100 gives an area approximately half as tall as the height of your screen. Then, you can adjust the Amount of the copies spewed out as you drag (think of it as the *rate* of painting).

Next, you can change how much random Rotation and Scale are applied to the copies—from zero (same-size copies in the same orientation) to 100 (free rotation and scaling from zero to twice as big as the original). Within the painting area, you can vary how much Scatter and Focus to use, from zero (concentrated in the center of the area) to 100 (evenly spread over the entire area). Inkscape supports pressure-sensitive input devices, such as Wacom tablets (3.3), and this tool allows you to apply pressure to vary the painting area size, rate of painting, or scaling of the copies—or any combination of these.

On the left end of the controls bar, the Mode switch controls what the Spray tool creates: plain copies (the default), clones (Chapter 16), or subpaths of a single path (12.1.1). Here, copies are objects that you can change independently from the original, whereas clones are automatically updated to reflect the style and shape of the original and cannot be edited on their own (other than by transforming). The subpaths mode creates a single path object (12.1), where overlapping objects are merged together as if by the path union operation (12.2). The fourth Mode button enables a spray-specific eraser: in this mode the tool selectively deletes, within its paint area, previously sprayed copies or clones of the selected object(s) but leaves any other objects alone.

4.7.1 Tracing by Spraying

At the right end of the Spray tool's controls bar, there are widgets that set up a different way of spraying: tracing the background. When tracing, Inkscape no longer spews out copies of objects blindly but looks at what is there under your mouse pointer. It samples the color and opacity of the existing drawing in the spraying area and uses those values to modify the way it sprays. (This functionality is available only when spraying copies or clones, not subpaths.)

Often, though not always, the already-existing object you're spraying over will be a bitmap—for example, a photo. Using tracing, you can literally re-paint, or at least over-paint, a photo in a striking artistic manner using any kind of objects as your "brush." This is similar to a few other Inkscape features I'll be discussing later, such as the Calligraphic pen tracing (16.6.6) and the Trace tab in the Create Tiled Clones dialog (16.6).

The first two toggle buttons in this area, those with the eye open and eye closed, let you control where you want your spray to appear. When the eye-open button is pressed, sprayed objects can appear over opaque objects; when the eye-closed button is pressed, they can appear over transparent objects, transparent

areas of bitmaps, or empty canvas. By default, both of these buttons are pressed—that is, you can spray anywhere.

Next come the two no-overlap buttons. The plain *no-overlap option* (similar to the Unclump feature, 7.5.2) ensures that the sprayed objects do not touch each other, which makes the result look much neater. When this button is pressed, you can additionally specify how far you want the objects to be from one another; negative values are allowed too, which means you can have a controlled partial overlap.

The other no-overlap button pays attention to the colors under the mouse pointer: it allows overlapping only within areas of the same background color. This way, you can pile up objects inside single-colored cells but avoid crossing the cell boundaries, as shown in Figure 4-9.

Figure 4-9: Using the Spray tool with color tracing and no-overlap options

Finally, if you press the button with the eyedropper, you enable the *pick color* mode where Inkscape picks the color and opacity from the mouse point. The rest of the buttons on the toolbar affect how exactly this picking is done and what to do with the picked values. Specifically, you can assign the color to the objects' fill (that's what Figure 4-9 demonstrates) and/or stroke; you can invert the picked color; and you can pick it from the exact point of the mouse pointer (instead of averaged over the tool's paint area).

Remember that you can always trim down or thin your sprays, without completely undoing them, using the Eraser mode of the tool. Also, the versatile Tweak tool is indispensable in the toolbox of a spray artist. With it, you can easily nudge, sway, jitter, scale up or down, rotate, or multiply your sprayed objects (6.10), as well as style them by shifting their color, opacity, or blur (8.9).

4.8 Groups

Grouping is a way to make a single object out of a number of independent objects. It is an easily reversible action: objects combined into a group can be ungrouped and made independent again. Moreover, many of Inkscape's tools disregard grouping and work directly with individual objects, whether they are grouped or not. The Selector tool by default selects the group as a whole—but even in it, there are methods to select an object inside the group without ungrouping it (5.10).

To group some objects, just select them and choose **Object ▸ Group** (Ctrl-G). Groups can be themselves grouped just like any other objects. You can even group a single object—for example, select a rectangle and press Ctrl-G. Now you

have a group with a single rectangle inside. Such groups may be useful in some situations, such as blurring a clipped object (see 18.3).

GROUPS ARE PARENTS

In SVG, a group is the g element (see A.5), which is a *parent* of its member objects. This means, among other things, that objects with unset style properties (8.2) will inherit these properties from their parent group or from an ancestor further up the tree.

When a single group is selected, the status bar indicates how many member objects are within this group—for example, *Group of 3 objects in layer Background*.

4.8.1 Ungrouping

To ungroup a group, select it and choose **Object ▸ Ungroup** (Ctrl-U or Ctrl-Shift-G). The group no longer exists, but all its former members, now released, remain where they were on the canvas—both the coordinates and the z-order of the objects are preserved.

You can select any number of groups and ungroup them all at once. Moreover, you can select any number of different objects, only *some* of which are groups, and press Ctrl-U: groups will be ungrouped and all nongroup objects will remain intact.

The Ungroup command removes only the topmost level of grouping. For example, if you have a group containing two other groups, pressing Ctrl-U will release the two groups—they will stay selected but not ungrouped. You will need to press Ctrl-U again to ungroup them. To ungroup all groups in a complex drawing, just select all objects and keep pressing Ctrl-U until the status bar says, *No groups to ungroup in the selection*.

Finally, if you have so many levels of grouping that manual ungrouping does not sound like fun, try the **Extensions ▸ Arrange ▸ Deep Ungroup** extension. It will ungroup your document all the way down to subatomic particles—or, if you prefer, you can control precisely how many levels to ungroup and how many (counting from the bottom) to leave intact.

4.8.2 Uses of Grouping

There are several good reasons to group objects:

- First, groups provide an easy way to *select sets of objects*. With the Selector tool, click any object in a group, and the entire group gets selected. In this sense, a group is a "saved selection." After selecting, it is easy to move, scale, or paint the group just as you would a number of separate selected objects.
- Second, grouping is a natural way to *organize complex artwork*. When you have thousands of objects in your drawing, sorting out what belongs together is difficult. One way to structure such complex documents is via layers, as discussed later in this chapter. Often, however, layers are too much hassle—for example, if you just want to ensure that the nose and the eyes in your portrait

are never accidentally moved relative to each other, the easiest way to do that is by grouping them together.

- Finally, sometimes groups allow you to achieve effects that would be impossible otherwise. One such effect is *group transparency* (see 8.3 for more on transparency).

When you apply transparency to a group, it is made transparent *as a whole*, which may look quite different from applying the same level of transparency to the individual objects, as shown in Figure 4-10.

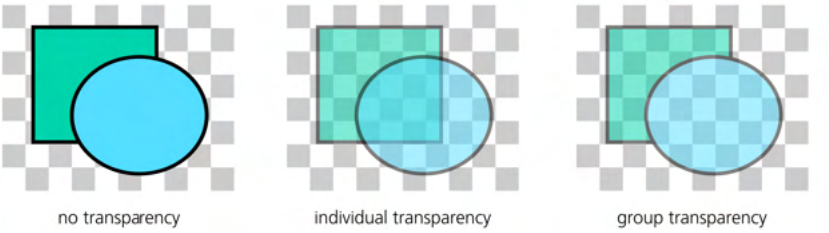


Figure 4-10: Group transparency

On the left, two independent objects are fully opaque (zero transparency). In the middle, they are made 50 percent transparent as individual objects; note that the corner of the rectangle shows through the ellipse. On the right, the original opaque objects are grouped and then the group is made 50 percent transparent; now, the checkered background shows through, but the overlapping area of the rectangle is still obscured by the ellipse and does not show through.

NOTE *Because of group transparency and other similar effects, ungrouping a group does not, in the general case, guarantee that the document's appearance will not change.*

4.8.3 Groups and Z-Order

An important thing to remember is that a group, as an object in its own right, has its own place in the z-order stack of your document, and all the members of the group share its z-order position.

This means that if object A is on top of B, and B is on top of C, you cannot group A and C without changing the z-order. If you attempt to do that, C will jump up on top of B, and you will have a group containing A and C lying on top of the object B.

Ungrouping, in that case, will not restore the original z-order. If you ungroup the "A and C" group, you will have three independent objects, but they will be now in the order A, C, B (top to bottom). (Of course, if you simply Ctrl-Z to undo right after grouping, both grouping and the z-order change will be reverted.)

For the same reason, you cannot group objects from different layers. Both layers and groups are branches of the XML tree of the document, and you cannot have a branch (group) growing from more than one parent branch (layer). If you try to group objects selected in different layers or within different groups, Inkscape will complain (with a message in the status bar) and do nothing.

4.9 Layers

Layers in Inkscape are just what the name suggests: "levels" or "floors" within a document, stacked on top of each other and containing other objects. Every layer has a name. Layers are easy to hide (make invisible), lock (prevent changes), or rearrange.

Every object belongs to one and only one layer. To find out which layer the object belongs to, just select that object and look at the status bar. It will say something like *Rectangle in layer Layer 1*; here, *Layer 1* is the name of the layer. You can, of course, select objects from different layers, in which case the status bar will say, for example, *2 objects in 2 layers*.

One of the layers in the document is always *current*. Any new objects you create, paste, or import are added to the current layer. To make layers in complex drawings easier to work with, Inkscape's current layer *follows selection*. That is, if you are in layer A and select some objects in layer B (for example, by clicking it), layer B becomes your new current layer. Conversely, if you change the current layer, your selection is deselected.

Inkscape remembers the current layer when saving the document and restores it when you load it afterward. A new document template (see 3.2) usually contains one initial layer called *Layer 1*; it is current when you load the template—so, if you don't create any new layers, all your objects will end up in Layer 1.

Just like individual objects, layers may be locked or hidden. In a *locked* layer, objects are visible but cannot be selected. In a *hidden* layer, objects are both invisible and cannot be selected. You cannot add new objects to a layer that's either hidden or locked.

Typically, layers are hidden when you want to simplify a complex piece of artwork to concentrate on some parts of it. Hiding complex layers may speed up screen redraw considerably and thus make working more comfortable. Locking layers is useful when you want some background objects to be visible but not selectable—usually to make it easier to select the foreground objects by clicking or dragging around them.

4.9.1 Layer Hierarchy

At a more advanced level, layers in Inkscape are closely related to groups. In fact, a layer is just a kind of group that Inkscape treats in a special way.

LAYERS ARE GROUPS

In SVG, both layers and groups are represented by the same element, `g`. The only difference is that a layer has the attribute `inkscape:groupmode="layer"` (A.5).

Just as groups may include other groups as members, layers may contain further *sublayers*. This is useful for organizing complex artwork. Instead of a flat list of layers, you may have a hierarchical tree, where related layers are grouped by a common parent. You can then manage your document by raising/lowering

or hiding/locking the entire branches of the tree instead of individual layers one by one.

Even better, you can *enter a group*. That means telling Inkscape to temporarily treat a group as a sublayer and to make that sublayer current. To do this, just select the group and double-click it, or press Ctrl-Enter, or right-click it and choose **Enter group** from the pop-up menu.

This trick combines the advantages of groups (a group is easy to select, move, transform, style, view its bounding box, and so on) with the advantages of layers (a layer defines a context that you can comfortably work in—for example, by adding new objects to it). In particular, the easiest way to move some object *into* an existing group without ungrouping is to cut the object (Ctrl-X), enter the group, and paste the object there (Ctrl-V, or Ctrl-Alt-V if you want to preserve its position).

To leave a sublayer—whether it's a real sublayer or just a group that you entered—press Ctrl-Backspace, or double-click an empty canvas area, or right-click anywhere and choose **Go to Parent** from the pop-up menu.

4.9.2 The Layer Menu

The most important layer commands are collected in the Layer menu:

- The Add Layer command creates a new layer and asks you for its name (Figure 4-11). You can also choose whether to place it above the current layer (default), below the current layer, or inside the current layer as a sublayer. Layer names need not be unique and can use arbitrary characters. In SVG source, the name of the layer is stored in the `inkscape:label` attribute.

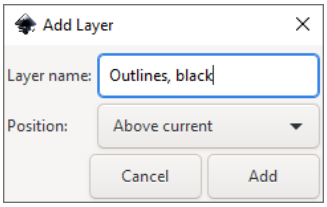


Figure 4-11: Creating a new layer

- Rename Layer, Duplicate Current Layer, and Delete Current Layer commands do what they say they do. Duplicating a layer duplicates all of the objects in it. Similarly (be careful!), deleting a layer deletes all objects that were in that layer.
- The two Switch commands just switch the current layer to one below or above it. These commands simply define the context for other operations; they do not change anything in the document and are therefore not undoable (only the commands that actually change the document can be undone).
- The next two commands, Move Selection to Layer Above (Shift-Page Up) and Move Selection to Layer Below (Shift-Page Down), take the current selection and transport it to the layer above or below the current one. (If there is no layer above or below the current one, these commands do nothing and complain in the status bar.) By crossing the layer boundaries, these

commands are complementary to the regular z-order changing commands, such as Raise or Lower (4.4), that work within the same layer.

To move an object from one layer to any other, not necessarily adjacent, cut the object (Ctrl-X), switch to the target layer, and paste it in place (Ctrl-Alt-V). If you need to extract an object from a group, select it inside its group (5.10) and use **Object ▸ Pop Selected Objects out of Group**.

- The four z-order commands—Raise Layer (Shift-Ctrl-Page Up), Lower Layer (Shift-Ctrl-Page Down), Layer to Top (Shift-Ctrl-Home), and Layer to Bottom (Shift-Ctrl-End)—are equivalent to the z-order commands for objects (4.4), except that they act on the current layer by moving it (with all of its objects and sublayers) up or down among its sibling layers. As you may notice, the keyboard shortcuts for these commands are the same as for the object z-order commands but with Shift-Ctrl added.
- Finally, the Layers... command (Shift-Ctrl-L) opens the Layers dialog (4.9.4).

4.9.3 The Current Layer Indicator

Inkscape has two main UI controls for working with layers: the basic current layer indicator in the status bar and the more powerful Layers dialog.

The *current layer indicator* displays the name of the current layer and, with the two toggle buttons on the left, indicates whether that layer is hidden (the eye button) and/or locked (the lock button), as shown in Figure 4-12.

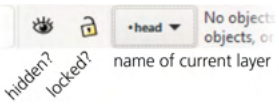


Figure 4-12: The current layer indicator in the status bar

It's an interactive control, not just a display; you can toggle the buttons and use the pop-up menu of all layers to change the current layer (Figure 4-13).

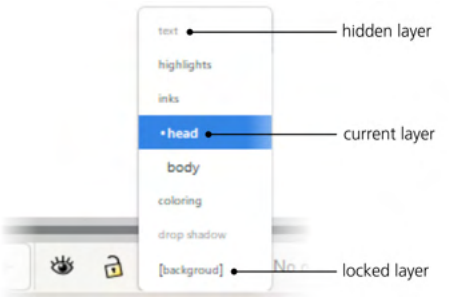


Figure 4-13: The pop-up menu of all layers in the document

In the list of the layers, the current layer is marked with bold face and a bullet. Locked layers have square brackets around their names (for example, *[background]*). Hidden layers' names are gray. A temporary layer (such as a group that you entered, 5.10) uses italics for its name.

4.9.4 The Layers Dialog

The current layer indicator is handy because it is always active and takes little space on screen. However, it is adequate only if your layer structure is small and simple. In more complex documents, it quickly becomes unwieldy. That's when you need to use the Layers dialog (Shift-Ctrl-L), shown in Figure 4-14.

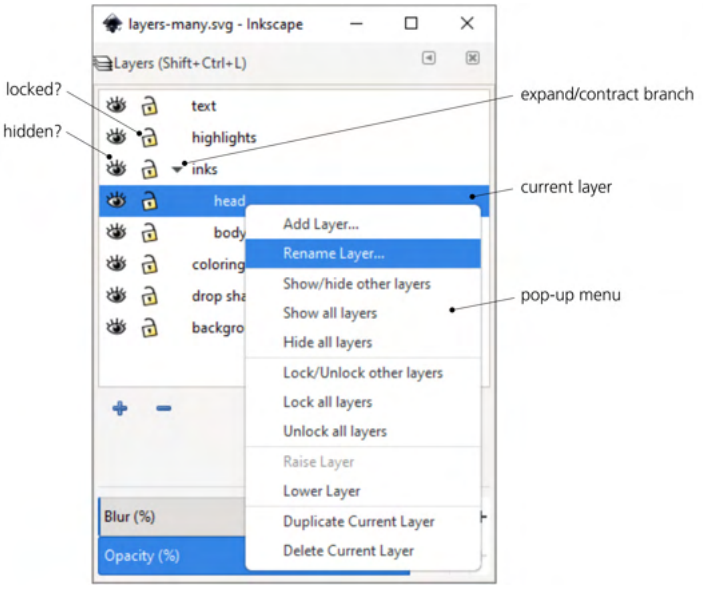


Figure 4-14: The Layers dialog

In the list of layers, you can expand and collapse branches of sublayers inside layers by clicking the triangle markers. Also, you can lock or unlock and hide or unhide any layer, without making it current, by clicking the corresponding icon to the left of the layer name.

Under the list of layers, the six buttons correspond to the following commands, left to right: Create a new layer (you need to provide a name), Delete the current layer, Raise the current layer to the top, Raise the current layer, Lower the lower layer, Lower the current layer to the bottom. To rename a layer, simply click its name in the list and type a new name, then press Enter.

Right-click a layer name to open a pop-up menu. Apart from the familiar commands for adding, renaming, lowering/raising, or duplicating a layer, it contains commands that let you hide/show and lock/unlock either all layers or all layers except the one you clicked on.

At the bottom of the dialog, there's a list of blend modes that can be applied to the entire current layer (17.2). Further down, two slider controls let you set the opacity or blur level of the entire current layer. These controls affect all the objects in the layer as a whole (see 4.8.2 about group opacity).

4.9.5 The Objects Dialog

Unlike Adobe Illustrator, Inkscape's Layers dialog does not show individual objects within layers. There is another dialog, called simply Objects (Object ▸ Objects), which does list all individual objects in the layer hierarchy (Figure 4-15).

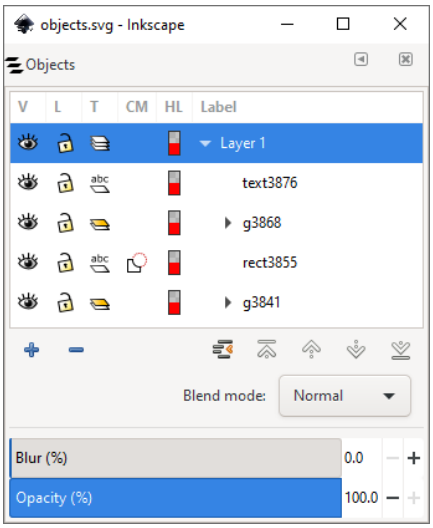


Figure 4-15: The Objects dialog

In many ways, this dialog duplicates the Layers dialog. You can add or remove layers, rearrange them, or set the blend mode (17.2), blur, and opacity levels for any layer or an individual object. You can toggle the visible (V column) and locked (L) status of any object, and you can convert groups to layers and back (click the icon in the T column for a group or layer). The dialog also marks objects that are clipped or masked (CM) and lets you choose the color of the path highlight (HL) that this object—if it's a path—will have in the Node tool (12.5.1).

4.10 The XML Editor

The XML Editor is one of the things that set Inkscape apart from other vector editors. This is where you can see the entire raw source of your document, with nothing hidden or (mis)interpreted. Simply put, if something is not shown in the XML Editor, it's not in your document.

If you're learning SVG or are simply interested in what lies behind some of the objects or properties of your document, the XML Editor is the tool for you. Here, you can do absolutely anything to your document. I might even say that XML Editor is the only essential part of the program, everything else being optional conveniences!

The XML Editor is completely synchronized with the rest of Inkscape. Any change you make in the XML tree is immediately reflected on your canvas, and any change you make with any other tool is immediately visible as an element or attribute change in the XML Editor.

The XML Editor has two main panes for the document tree and the attributes of the selected node (Figure 4-16). You can arrange these panes horizontally or vertically (with the toggle in the bottom-right corner). You can also hide the attributes pane with the Show attributes switch, bottom left.

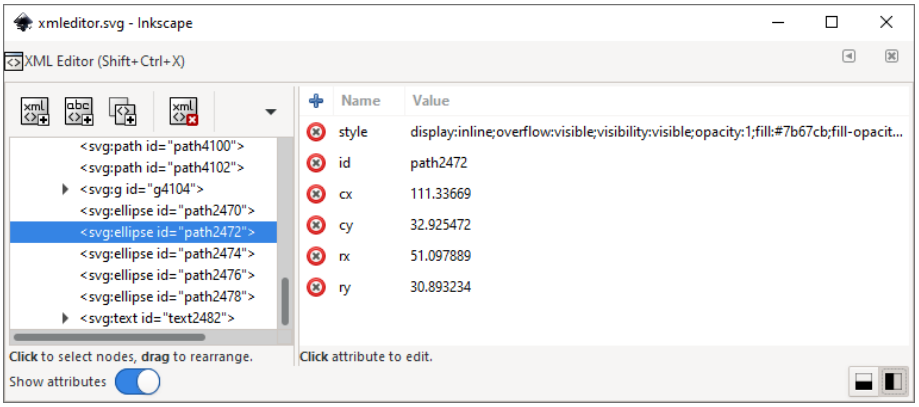


Figure 4-16: The XML Editor

4.10.1 The Tree Pane

This pane shows the entire tree of the document you are editing. Each node of the tree (each line in this list) is either an XML element or a text node. In SVG, text nodes are used only for holding the text content of text objects; all other objects are element nodes. For example, `svg:rect` is a rectangle, and `svg:g` is either a group or a layer (here, `svg` is the namespace prefix; see A.2). Of all the attributes a node might have, this pane only shows the `id` attribute (which is unique, so it's a natural way to tell nodes of the same type apart).

Not every element in the list is a visible Inkscape object. Some elements represent things (such as metadata) that you don't see on the canvas. For more on various SVG and Inkscape elements, refer to Appendix A.

NOTE *In the XML Editor, all elements are listed in the document order. This means that the objects and layers that are topmost in the z-order on the canvas are the last (bottommost) in the list. In particular, the order of layers in the XML Editor is reversed compared to the order in the* Layers *dialog or the current layer indicator.*

In the document tree, you can select any element by clicking it. This list selection is synchronized with the regular object selection in Inkscape. If the selected node corresponds to a visible object, that object gets selected on the canvas; conversely, when you select an object on canvas, the XML tree pane scrolls to the corresponding node and highlights it. This is one way to reach elements that are otherwise inaccessible, such as those that are locked or hidden (4.1).

Above the tree pane, there's a small toolbar with buttons for generic XML operations: creating a new element or text node and duplicating or deleting the selected node. The last four buttons are for moving the selected node (and its children) in the tree. You can shift a node sideways to change the nesting level (the left arrow button converts a node from a child of its parent to its sibling;

the right arrow button converts a node into a child of its preceding sibling) or use the up/down arrows to move a node among its siblings. You can also rearrange nodes by dragging them within the tree.

4.10.2 The Attributes Pane

This pane lists the attributes of the selected element node. (If you select a text node, this pane lets you edit the text of the node as if it were an attribute named "content".) Every attribute has a name and a value, listed in the two columns of a table. The order of the attributes does not matter in XML.

To edit the value of an attribute, just click it and edit in place (Figure 4-17). If the existing value is too long to fit on one line, a convenient little window pops out where you can edit the auto-wrapped value. (This is typically the case for `style` attributes; however, see 8.1 for a more convenient way to edit an object's style properties.) Pressing Enter accepts the changes.

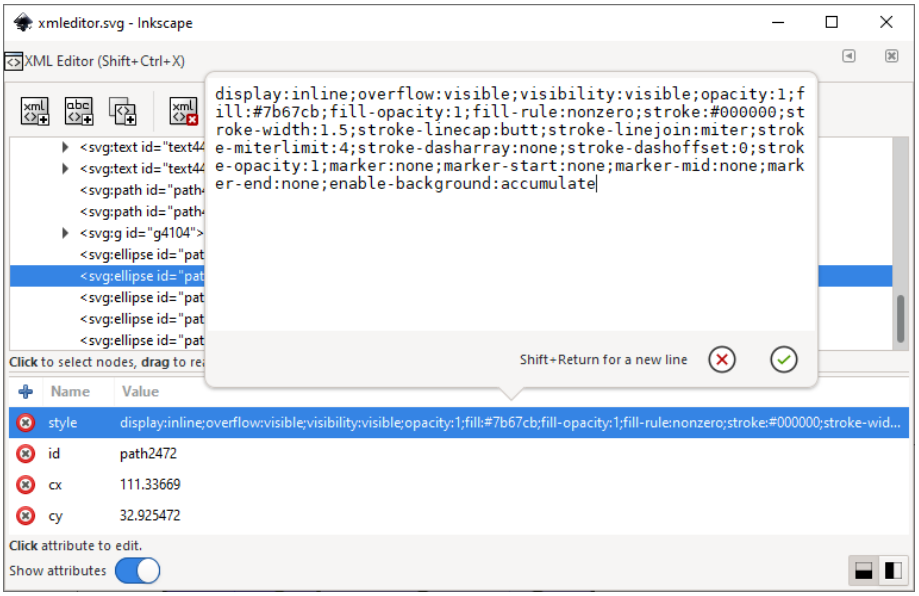


Figure 4-17: Editing the `style` attribute in the XML editor

To delete an attribute, click the red "x" button in the left column. To add a new one, use the blue plus button at the top of that column.

5

SELECTING

There are few useful things you can do in Inkscape without first selecting some object or objects. Of all Inkscape operations, selecting has, by far, the greatest number of methods, tools, dialogs, and shortcuts available. You need to know a good portion of them to be able to work efficiently—and at least a few to be able to work in Inkscape at all.

By itself, selecting or deselecting objects does not change the document in any way. Therefore, none of the techniques described in this chapter are undoable actions, and none require your document to be saved.

5.1 The Selection Cue

As you probably know by now, Inkscape's selection is just a list of objects, which may include anything—from no objects (empty selection, nothing selected) to all the objects in the document. Selection is local to the editing window; if you open a second window with the same document (3.6), that window will have its own independent list of selected objects. Selected objects can be anywhere on the canvas, in any layer or group; multiple selected objects don't have to be

children of the same parent. The only thing you *cannot* do is select an object and its ancestor at the same time (5.10).

NOTE *Even hidden or locked objects can be included in the selection. While most tools will refuse to select a hidden or locked object directly, it is still possible, for example, by using the XML Editor (4.10).*

On the canvas, each selected object is marked by the selection cue. By default, this cue is a dashed frame around an object, showing that object's bounding box (4.3). This frame is drawn on top of all objects using a contrasting color, so it is visible on any background.

You can switch to a different selection cue: a small diamond-shaped mark in the bottom left corner of each object's bounding box, as shown in Figure 5-1. To change the type of the selection cue or to turn it off altogether, go to the **Tools ▸ Selector** subpage of the **Preferences** dialog.

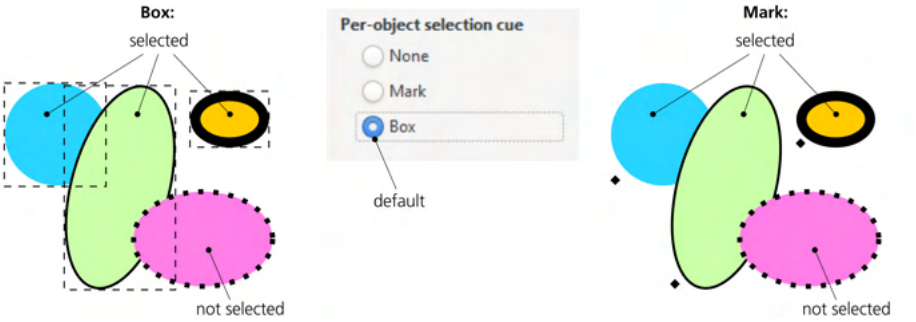


Figure 5-1: The two possible types of selection cues

Also, you can control which tools display the selection cue and which do not. For this, look for checkboxes labeled Show selection cue on each tool's page in the Preferences dialog. By default, every tool shows the selection cue, except for the Measure, Eraser, Tweak, Calligraphy, and Paint bucket tools—that is, those tools where the cue is not very useful or may be a distraction. (Of course, the selection is still there whether the cue is displayed or not—it remains the same no matter how you switch tools.)

5.2 Selection and the Status Bar

A lot of things happen throughout the program when you select or deselect objects. Inkscape instantly redirects all its attention to the new selection. It redraws the selection cues, scrolls the canvas if necessary to show the selected objects (3.12), and updates various displays and indicators all over the interface.

One of the most important sources of hints in Inkscape, the status bar, displays as much information about a selection as would reasonably fit into it. For a single selected object, it tells you the type of object, lists certain additional properties for some of the types (such as the number of nodes for paths, font family and size for text objects), names the layer in which this object resides, and gives some useful advice, as shown in Figure 5-2.

Figure 5-2: Status bar: a single selected object

For multiple selected objects, Inkscape tells you how many are selected and lists their types (Figure 5-3).

Figure 5-3: Status bar: multiple selected objects

If objects in more than one layer are selected, Inkscape gives you the number of objects and their layers along with the object types (Figure 5-4).

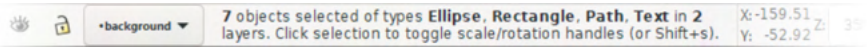


Figure 5-4: Status bar: multiple selected objects of multiple types in multiple layers

To the left of the status bar message area is the current layer indicator (4.9.3). Since selected objects can be in different layers, this widget is not directly connected to the selection. However, when you select a single object by clicking, Inkscape assumes that you now want to work in that object's layer—so it makes that layer current, which the indicator reflects.

Further to the left is the *selected style indicator*, which always reflects the style of the current selection (if many objects with different styles are selected, it displays their averaged style or says "Different" if the styles cannot be averaged). I discuss this important widget in detail in 8.2.

5.3 Subselection

Some tools allow a finer-grained type of selection so as to work on part of a selected object rather than an entire object. This selected part of a selected object is sometimes called the *subselection*.

One example is the Text tool. You can select an entire text object, but then you can select a *fragment* of text inside that object by using Shift with arrow keys, or by dragging with the mouse, just like you do in a text editor (15.1.1). That selected part of the text is the text subselection.

Another example is the Gradient tool. A gradient can have many stops (10.5), and the tool can select one or more of those stops at a time. Naturally, gradient stops are visible only for selected objects—so if you select a stop, you're creating a subselection within the regular object selection.

When you change subselection, some things in Inkscape behave the same as when you change selection. In particular, the selected style indicator in the status bar (Figure 8-2 on page 143) displays the style of the subselection—that is, the style of the selected text fragment and not the entire text, or the style of the selected gradient stop and not the entire object. Also, any style-setting commands (such as clicking a color on the palette, or pasting a style by pressing Shift-Ctrl-V) will apply to the subselection if it is present. In other words, subselection allows you to deal with parts of your objects almost as though they were objects in their own right.

5.4 Selecting by Clicking: the Selector

You may know already that the topmost tool in the toolbar is the Selector and that clicking an object with that tool selects that object. Of course, there is much more to the Selector tool than that, but let's look at this simple action in more detail.

First, note that clicking an object *deselects* any previous selection. Also, with the Selector tool, you can not only select objects but also *drag* the selected objects in any direction. Combined, these two features make it easy and natural to move objects around—almost without thinking about "selection" at all. You see an object, you click and drag it to where you want it. The click selects it, deselecting anything else, and the drag moves it.

Sometimes, this behavior can be annoying. Especially if you use a tablet with a pen instead of a mouse, you may find it too easy to nudge an object accidentally when all you wanted to do was to select it by clicking. To make this less of a problem, go to the **Input/Output ▸ Input Devices** page of the **Preferences** dialog and adjust the **Click/drag threshold** value. This is the allowed "slippage" of the mouse, the default being 4 screen pixels; if you click and drag an object by less than this many pixels, your action is still considered a click and the object does not move. Increase this value if you often end up accidentally moving objects instead of clicking them; conversely, decrease it if you find that objects annoyingly "stick" more than you like when you really want to move them.

Before you click anything, however, you want to be sure that you're clicking in the right place and that your click will not be wasted. Inkscape tries to be helpful: it changes your mouse cursor, by appending a cross-with-arrows icon to it, when you are over a clickable area of an object—as opposed to an empty canvas where the cursor is a plain arrow.

Play with this cursor-changing capability a little. You will discover that objects with no fill cannot be selected by clicking inside them and that fully transparent objects are not selectable by clicking at all (although you can select them with the rubber band, as you'll see in the next section). In Outline mode (3.14), you can select an object only by clicking its outline.

As you can imagine, this changing cursor is less helpful in complex drawings where the entire canvas is often covered with objects. However, if you separate background objects into a layer and lock that layer (4.9), those objects, now unselectable, will no longer change the cursor—so you can again sense the foreground objects by moving your mouse over them.

Also, you will notice that every clickable object has an invisible margin several pixels wide on all sides. Clicking in that margin still selects the object. This is handy for selecting small objects that would otherwise be almost impossible to click upon accurately. On the downside, this also explains why it's sometimes difficult to select the bottom object in a stack even if that bottom object protrudes a little from under the top one.

If you don't like the size of this clickable margin, you can change it on the Input/Output ▸ Input Devices page of the Preferences dialog. This is the Grab sensitivity value, the default being 8 screen pixels. Note that both this value and the Click/drag threshold are in screen pixels, which means they do not depend on

zoom; of course, you will find it easier to perform small moves and select small objects when you're zoomed in.

5.5 Selecting by Clicking: Other Tools

One of the principles of Inkscape's user interface is consistency. Unless there's a reason to do otherwise, all tools and modes strive to behave the same. This means that many tools, just like the Selector, can select an object by clicking it.

Clicking to select works in the following tools: Node (12.5), all of the shape tools (Chapter 11), Text (Chapter 15), Connector (1.2), Gradient (10.1), and Mesh (10.7) tools. The tools that *cannot* select by clicking are different for a reason—in those tools, a single click is reserved for a different function specific to that tool.

Unlike the Selector, all these tools directly select individual objects even when they are inside groups (in other words, in these tools a simple click is equivalent to Ctrl-click in Selector, 5.10). This makes sense; all of these tools work on individual objects of various types, so in most cases, selecting a group is not what you really want to do. For example, if you select a group with the Node tool, you would not be able to do anything useful with it—a group has no path nodes to edit. That's why this tool always directly selects the *path* you clicked, whether it's grouped or not.

Also, unlike the Selector, other tools don't change the cursor over selectable objects. The only exception is the Text tool that switches to the text insertion cursor when over an editable text object (15.1).

5.6 Adding to a Selection

A selection can contain more than one object. If you've already made the effort to select some objects, and then want to select some other objects as well, can you do so without starting the selection all over again?

To *add* another object to the current selection, Shift-click it. Conversely, if you Shift-click an already selected object, it will be *removed* from selection. This shortcut works as a toggle that inverts the selected status of the object that you Shift-click.

5.7 Selecting with the Rubber Band

Another way to select multiple objects in the Selector is by dragging *around* them. Imagine a rectangle surrounding all the objects you want to select and drag from one corner of that rectangle to the opposite corner. (The exact direction of the drag—top left to bottom right versus top right to bottom left, for example—does not matter.) This rectangle (shown in Figure 5-5), which is visible while you're dragging, is called the *rubber band* or *marquee*.

Selecting by rubber band is *not a toggle*; once you start a new rubber band, any previous selection is deselected. This is why, logically, single-clicking on empty space (not an object) deselects anything—it's just a zero-sized rubber band that deselects the old selection without creating a new one.

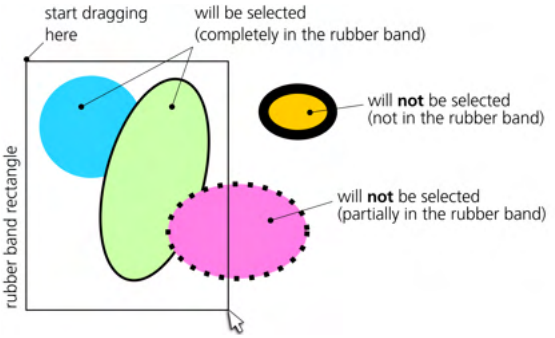


Figure 5-5: Using the rubber band (mouse drag) to select multiple objects

The rubber band selects fully enclosed objects in all visible and unlocked layers—in other words, across the z-order stack of the entire document, not the current layer only. (For this reason, selecting by rubber band does not change the current layer, even if it resulted in selecting a single object in a layer different from the current one.) Also, the rubber band selects those objects that you cannot select by clicking—those that are underneath others, those without fill and stroke, and those that are fully transparent (have zero opacity). However, objects that are hidden or locked (4.1) still cannot be selected.

In short, if an object's bounding box is completely inside the rubber band rectangle and that object is at all selectable, it *will* get selected.

RUBBER BAND BEHAVIOR

In Adobe Illustrator, the rubber band works differently. It selects all objects whose bounding box lies within *or intersects* its rectangle. Inkscape's default behavior—selecting only what is *completely* inside the rubber band—matches that of the CorelDRAW and Xara vector editors.

Inkscape also can emulate Adobe Illustrator's selection behavior. Press the **Toggle selection box to select all touched objects** button on the Selector's controls bar (fourth from left) to change the selection rubber band's behavior. In this mode, the rubber band is shown as a red dashed line.

It's tempting to assume that, as with clicking, dragging the rubber band with Shift would give you a toggle behavior. This is not quite the case, however. Shift-dragging with the Selector tool works like simple dragging except that it *always* creates a rubber band, even if you start from an object and not from an empty canvas, and *then* adds (not toggles) enclosed objects to selection. Without Shift, dragging from a (selectable) object will select and move that object, but pressing Shift and dragging *forces* the rubber band—which makes it a lot more useful in complex drawings where empty canvas may be hard to come by.

5.8 Touch Selection

Touch selection is a close relative of the rubber band. In the Selector tool, with no previous selection, drag from an empty canvas with Alt. You will see a red trail left by your mouse cursor—the *touch path*. After you release mouse button, all objects *touched by* (not included in) this trail will become selected (Figure 5-6).

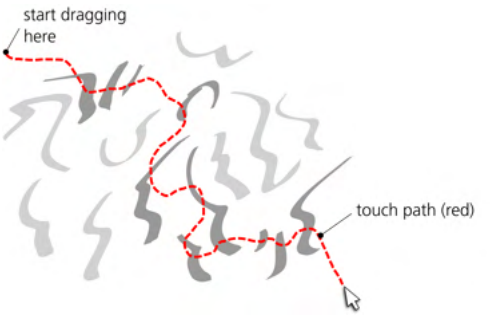


Figure 5-6: Using touch selection (Alt-drag). Objects that will be selected after you release the mouse are marked with darker gray.

This means you can literally "paint" over the objects you want to select, which is convenient when the objects you need are close together but their bounding boxes are too big (or too intertwined with other objects) to use the rubber band.

Unlike the rubber band, touch selection does not select completely transparent objects or those obscured by others—that is, objects that you cannot really "touch." In fact, touch selection behaves exactly as if it were a series of Shift-clicks along the touch path so that each touched object gets one such click.

If there is a selection, Alt-drag has a different function: it moves the selection regardless of where you drag (6.1). What if you already have something selected and want to *add* to it with touch selection? Use Shift-Alt-drag: just like Shift forces the rubber band even if you start from an object, with touch selection, it forces the touch path even if you have a previous selection. The touched objects are then *added* to the selection.

If you touch-select objects so you can then delete them, the Delete mode of the Eraser tool (14.4) performs both functions at once: drag over objects, and they'll be deleted once you release the mouse. You don't need to hold Alt in the Eraser tool.

5.9 Selecting Objects from Underneath

One of the common problems with selecting objects in complex drawings is that some objects obscure others. Even if the top object is partially transparent and you can see another object underneath, simply clicking it will still select the top object.

If the bounding box of the bottom object is smaller than that of the top one, you can Shift-drag a rubber band around the small bottom object, and this will select it without selecting the top object. However, this is not always possible.

5.9.1 Alt-click

To select objects that aren't at the top of the z-order, use Alt-click, which *selects under* the currently selected object. This means that the result of an Alt-click depends on the current selection; if nothing is selected, or if you Alt-click outside the current selection, the result is the same as if you simply clicked without Alt.

If, however, you click a selected object and there are other objects underneath it, Alt-click will deselect the selected object and instead select the object immediately beneath it. The next Alt-click will select a still deeper object, and so on. When you reach the bottommost object at the click point, the next Alt-click again selects the topmost one.

Imagine you have three stacked objects, numbered 1, 2, 3, from bottom to top. With nothing selected, Alt-clicking them selects the topmost one, 3. The next Alt-click selects 2, then 1, then 3 again, and so on.

Shift-Alt-click differs from Alt-click in the same way that Shift-click differs from a simple click: it adds to the selection or removes from it without unselecting it completely. So, in our 1, 2, 3 stack of objects, the first Shift-Alt-click selects 3; after another Shift-Alt-click, you will have 2 and 3 selected; one more Shift-Alt-click adds the bottommost object to the selection as well. So, after three Shift-Alt-clicks, you will have 1, 2, and 3 all selected. Further Shift-Alt-clicks will toggle the selected status of 3 while keeping 1 and 2 selected.

[1.1]

5.9.2 Alt-scroll

If you have a mouse with a scroll wheel, there's another way to select underneath an object. Rotating the wheel with Alt selects objects in the stack under the mouse cursor, one by one in turn, one object per notch. Rotating forward goes from top to bottom in the stack; rotating backward goes from bottom to top. Thus, in the 1–2–3 stack example from the previous section, Alt-scrolling forward over it will select objects in the order 3, 2, 1, 3, 2, . . . and Alt-scrolling backward will select 1, 2, 3, 1, 2, ...

To make this even more intuitive, Inkscape temporarily adjusts the opacity of all the objects in the stack while you Alt-scroll, overriding any opacity they might naturally have (Figure 5-7). Once you start rotating, all the objects in the stack become semitransparent, except for the newly selected object, which becomes fully opaque. This way, you can see at once where in the stack you are and how much you need to scroll to get to the object you need.

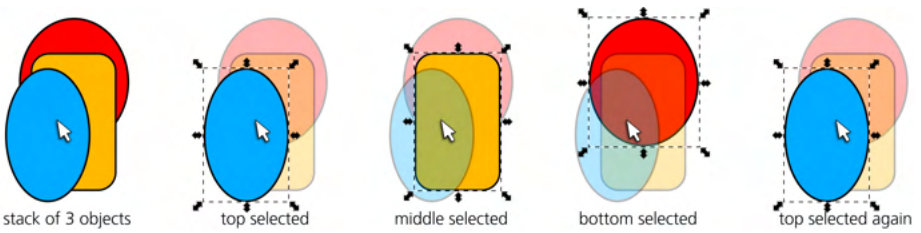


Figure 5-7: Using Alt-scroll wheel (rotating forward) to select one from a stack of objects

If you add Shift to Alt-scrolling, Inkscape will no longer deselect the object it had selected at the previous notch of wheel rotation. This way, you can add the entire stack of objects under the cursor simply by pressing Shift-Alt and rotating all the way up or down until you exhaust the stack. This is more convenient than Shift-Alt-clicking because there's no back-and-forth toggling of the top object once you reach the bottom.

Another difference between Alt-clicking and Alt-scrolling is that the latter ignores grouping and always selects individual objects, whether they are grouped or not. To do the same with clicking, you will need to add the Ctrl key to the mix, as you will see in the next section.

5.10 Selecting in Groups

Grouping is a boon when you want to treat a collection of objects as a whole: simply click any object in a group (with the Selector tool) and the entire group gets selected. However, quite often you want to select and edit an object inside a group without ungrouping it. For this, simply Ctrl-click an object in a group.

Ctrl-clicking ignores *any* grouping, no matter how many levels deep it is. For example, if object A is a member of group B, which, in turn, is a member of top-level group C, then Ctrl-clicking A will select A, cutting right through both levels of grouping. There's no way to select group B by Ctrl-clicking—it will always select the lowest-level nongroup object.

The only way to select group B, which is inside group C, is by *entering* group C (4.9.1). Entering C makes it a temporary layer. Once in that layer, you can select B with a simple click.

You can combine Ctrl-click with Alt (select under). Logically, Ctrl-Alt-click does the same thing as Alt-click, except that it disregards any grouping and browses through the z-order stack of objects at the click point as if they were all ungrouped. (As for Alt-scrolling, it already disregards grouping, so adding Ctrl to it has no effect.)

Similarly, you can combine Ctrl-click with Shift to add an object to selection or remove it from selection. Finally, you can Shift-Ctrl-Alt-click, which means "Add to selection the topmost nonselected object in the z-order stack at this point, ignoring grouping; if all objects at this point are selected, deselect the topmost one."

The only limit to the power of selecting by clicking with various modifiers is that you cannot have both an object and a group that contains that object selected at the same time. So, for example, if you Ctrl-click an object inside a group and then Shift-click (without Ctrl!) another object of the same group, thus trying to add the group to the selection, the group becomes selected, but the first selected object is deselected. Having both a group and an object inside that group selected at the same time would lead to all kinds of paradoxes, so Inkscape does its best to prevent this from happening.

5.11 Selecting with Keyboard Shortcuts

Generally, selecting is a task for a mouse or pen—because, in most cases, it is done by indicating some points or areas on the screen. However, you can also use keyboard shortcuts for two very common selection operations: selecting the *next* or *previous object* (Tab or Shift-Tab) and selecting *all objects* (Ctrl-A).

Here, *next* and *previous* actually refer to the z-order of objects inside the document (see 4.4). When you press Tab, you select the object immediately *above* the currently selected object in the z-order (or above the topmost selected object, if several objects are selected). Correspondingly, Shift-Tab selects the object immediately *below* the (bottommost) selected object.

If nothing is selected, Tab selects the bottommost object in the current layer, and Shift-Tab selects the topmost one. Since objects are typically added to the top of the current layer's z-order, pressing Shift-Tab with no previous selection is therefore a convenient shortcut for selecting the most recently added (drawn, pasted, imported) object.

You can change some aspects of this behavior on the Behavior ▸ Selecting page of the Preferences dialog (Figure 5-8).

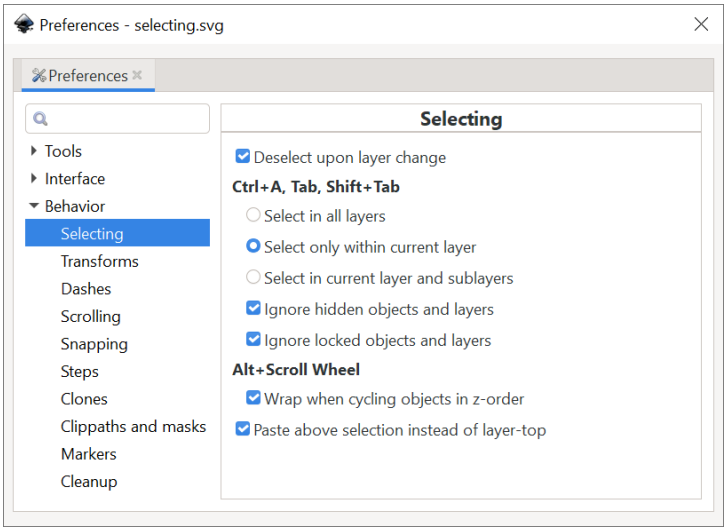


Figure 5-8: The Selecting page of the Preferences dialog

- By default, these keys are limited to the current layer (which may be a group that you have entered, 4.9.1). That is, once you reach the topmost object of the layer by pressing Tab, another Tab takes you back to the bottommost object in the same layer; pressing Ctrl-A selects all objects only in the current layer. The Preferences dialog describes this behavior as Select only within current layer. The Select in current layer and sublayers option works the same except it enters inside the sublayers of the current layer (4.9.1), and Select in all layers allows Tab or Shift-Tab to cross the boundaries of layers and Ctrl-A to select all objects in the entire document. Another way to select all objects in all layers, regardless of preferences, is by pressing Ctrl-Alt-A.

- You can uncheck the two checkboxes, Ignore hidden objects and layers and Ignore locked objects and layers, to allow the keyboard shortcuts to select those objects that are hidden or locked. If you also choose Select in all layers, you will be able to reach inside a hidden or locked layer (4.9).

5.12 Selecting by Properties

[1.1] A number of commands in the Edit ▸ Select Same menu let you select all objects in the document that are the same—according to some parameter—as the currently selected object. For the selection criteria, you can use the same fill color, the same stroke color, both fill and stroke color, stroke style (including color, width, dash pattern, Chapter 9), or the object type (rectangle, text, group, and so on).

5.13 Selecting by Searching

Searching for objects is typically only needed in complex documents—but when you do need it, you'll be glad Inkscape can do it. This chapter on selection is the best place to discuss searching simply because it is, in essence, yet another way to select objects. Even though Inkscape's Find/Replace dialog can do replacing (that is, modifying) as well as searching, you still need to search (that is, select) before you can replace.

To open the Find/Replace dialog (Figure 5-9), press Ctrl-F or use **Edit ▸ Find/Replace** from the menu.

To search, type your query in the **Find** field, check the options, then click **Find**.

Choosing the Text option applies only to text objects (Chapter 15) by searching within their text content. Otherwise, you can search in the Properties (detailed below) of all kinds of objects.

Default search includes all of the document; you can limit it to the Current layer or to Selection. The latter option is useful for a multistage search—for example, if you want to find text objects that contain both "day" and "night", search for "day" first and then search for "night" within the selection.

Search can be Case sensitive (off by default) or require Exact match (by default, partial matches will be selected too, such as a text object with the word "selected" when you search for "select").

You can choose to apply search to objects that are hidden and/or locked (individually or by being in hidden/locked layers).

You can search in the ID attributes of all objects (4.1), which are unique for each SVG element (though not very meaningful, unless you edit them yourself).

You can also search within the Style of objects. As this is a simple text field, to use this feature, you need to have at least some idea of how style is encoded in SVG (see A.8). When in doubt, you can always check the `style` attribute of an object in the XML Editor (4.10). Here are a few examples:

Finding all red-filled objects

This is possible, but only for one specific red color. You need to know how to convert that color to the RRGGBB form. Hovering your mouse over a color on the palette (8.5) in the selected style indicator (8.6) will give you

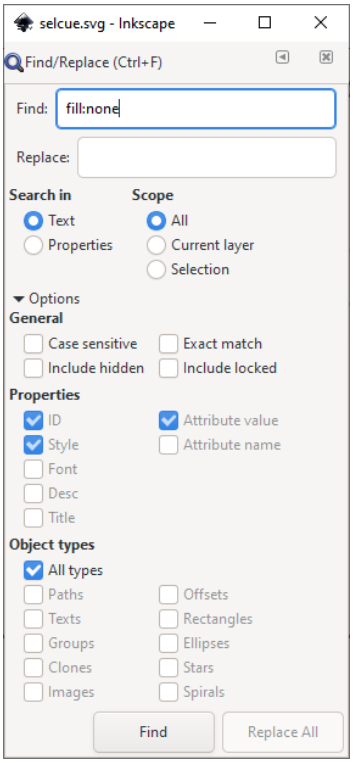


Figure 5-9: The Find/Replace dialog

the color's RRGGBB in the tool tip. For example, if you're searching for the bright red fill (`#ff0000`), in an object's style string the color is preceded by the name of the property, `fill`, separated by a colon. So your search string will be `fill:#ff0000`.

Finding all objects without stroke or fill

Search for `stroke:none` or `fill:none`.

Finding all fully transparent objects

Search for `opacity:0;` (note the separator semicolon at the end; without it, it will also find all *partially* transparent objects—for example, those with `opacity:0.5;` in their style).

Separately, you can search text objects by the name of the font they use (even though it's usually also in the `style` attribute).

More generally, you can search for a string in the value or name of any attribute. For example, all objects that you export to bitmap using selection export get the attribute `inkscape:export-filename`, which stores the filename of the bitmap export file. So searching for this attribute will select all objects in the document that were ever exported (following which you can, for example, export them all again, see 18.6).

Finally, you can limit your search to specific Object types: uncheck the default **All types** and select which types you are interested in.

5.14 Following Links

As you will see in the following chapters, objects in Inkscape can be linked to each other in various ways. You will often want to follow such links to find the object to which your current selected object is linked. Inkscape has a universal keyboard shortcut for this, Shift-D. Depending on what kind of object is selected, pressing Shift-D will select:

- The original of a clone (16.4), if a clone is selected.
- The path to which the selected text-on-path object is attached (15.2.4).
- The path to which the selected linked offset object is attached (12.4).

Some document objects may be stored in the `defs` (see A.4), in which case, you can reference them from the document, but they are not visible anywhere on the canvas. If your onscreen clone, text-on-path, or linked offset refers to an object in `defs`, Shift-D will fail with an error message in the status bar.

5.15 Deselecting and Inverting

In almost any tool, context, or mode, pressing Escape will deselect the current selection. If you have a subselection, typically pressing Escape once will deselect the subselection and the second Escape will deselect the regular selection.

Another way to deselect is by clicking the empty canvas (or a locked object with nothing underneath it). This, however, works only in Selector and in those tools where a simple click selects objects (5.5). In complex drawings where accessing empty canvas may be difficult, Escape is by far the most convenient way to deselect.

Also, by default Inkscape deselects when you switch the current layer (see 4.9). You can disable this on the Behavior ▸ Selecting page of the Preferences dialog.

Sometimes, it's easier to identify things you do *not* want to select. Inkscape makes this easy—just select the objects you don't need and do **Edit ▸ Invert Selection**. What was selected loses selection; what wasn't selected (within the current layer) becomes selected.

5.16 Selection Miscellany

When you undo an action, your current selection is usually (but not always) deselected; whether this happens depends on what kind of action you're undoing. A similar problem is that selection is often lost after an extension (Chapter 19) is run. Moreover, if an extension opens a configuration dialog and you check the Live preview checkbox in it, you cannot change selection (or do anything else on the canvas) while that dialog is open.

Inkscape's powerful command line interface has a special parameter for selecting objects, `--select` (C.6). To use it, you need to know the IDs of the objects you want to select. This makes it possible to script fully automated Inkscape editing sessions where a single command loads a document, selects some objects, performs some actions on them, saves the document, and quits—all without any user interaction.

6

TRANSFORMING

In Inkscape (and in SVG), *transformation* has a very narrow meaning. It does not refer to any change of an object but only to the four simple operations that affect an entire object in the same way: *moving*, *scaling*, *rotating*, and *skewing*. Nothing else is considered a transformation.

Metaphorically, transforming objects is moving around the furniture in your house without repainting the walls or opening up any cabinets.

THE TRANSFORM ATTRIBUTE

Transformations applied to an object are often (but not always) stored as the `transform` attribute in SVG; for more on when this attribute is or is not written, as well as the relevant preferences settings, see A.7.

For the mathematically inclined, the term is actually an abridgment of "affine transformation," where *affine* means that such a transformation preserves straight lines, parallel lines, and the ratios of the lengths of segments on a line, but may not preserve sizes or angles.

6.1 The Selector: Moving

After selecting objects, transforming them is the second most important function of the Selector tool. And of all the kinds of transforming, moving is the easiest: just grab an object (if it is not selected yet, clicking will select it; you just need to be in the object's clickable area for this to work) and drag.

Such *free dragging* is easy and inspiring. You likely will do a lot of it in the early stages of your work when you move things around to establish the overall composition.

During later stages, however, you're more likely to use *constrained dragging*. The most common constrained mode is horizontal or vertical dragging with Ctrl (Figure 6-1). If you press and hold Ctrl while dragging, Inkscape creates two invisible "rails," one horizontal and one vertical, which intersect at the point where the dragging started. The selection can then move only along those rails, jumping from one to the other depending on which is closer.

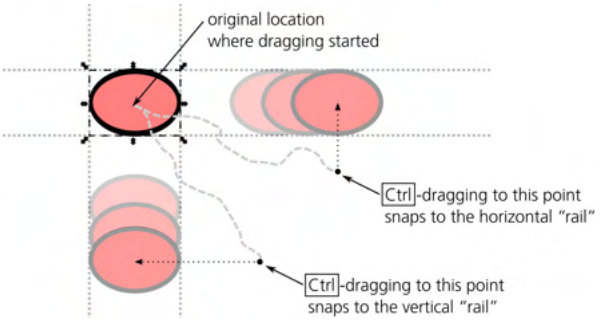


Figure 6-1: Constrained dragging with Ctrl

Using Ctrl has the same general effect of constraining your actions in many other situations as well. (In this, Inkscape is different from Adobe applications that normally use Shift for a similar purpose.) Many more ways to constrain object movement are available via snapping, discussed in the next chapter.

As you saw in the previous chapter, selecting just the objects you need may be difficult; you may have to use Alt-clicking (selecting under) and Ctrl-clicking (selecting inside groups) to get to the objects you need. Now, if you want to drag these selected objects around, you may run into another problem: any drag starts with a click, and that click may easily ruin your carefully constructed selection by selecting something else, such as the object that happens be on top of your objects or the containing group.

Often, the easiest workaround for this problem is to move the selection with the arrow keys, not by dragging the mouse (6.5). If you absolutely need to mouse-drag, start dragging while pressing Alt. This forces the current selection to be dragged without inadvertently selecting other objects. You can even start your Alt-drag to move the selection from any point—not from the selection itself but from an empty canvas or any other object.

You might wonder how this Alt-drag is compatible with Alt-click—which *does* change selection by "selecting under." Unlike regular click-to-select, however, "selecting under" happens not when you click your mouse button but when you *release* it after a click. If, between click and release, you didn't move your mouse

by more than the click/drag threshold (5.4, the default is 4 screen pixels), this is considered a click, and "selecting under" is performed. Otherwise, the current selection is moved.

Finally, moving with Shift lets you temporarily suppress snapping (7.3). This is sometimes convenient when you normally use snapping and don't want to disable it, but just need to place a certain thing exactly where snapping won't let it go.

6.2 The Selector: Scaling

Our next topic is *scaling* the selection—making it bigger or smaller. Scaling is not the same as zooming; when you zoom, you just view your drawing from closer up or from farther away without changing it. Scaling means actually resizing the objects; this is an undoable action.

For scaling your selection, the Selector tool displays eight handles, four at the corners and four on the sides of the selection's bounding box, as shown in Figure 6-2. Dragging the side handles scales the selection in one direction (horizontally or vertically); dragging the corner handles scales it in both directions.

NOTE *Are the handles too small on your screen and hard to use? You can select the most comfortable handle size in* Preferences ▸ Interface ▸ Handle size. *This setting also affects the nodes in the Node tool (12.5).*

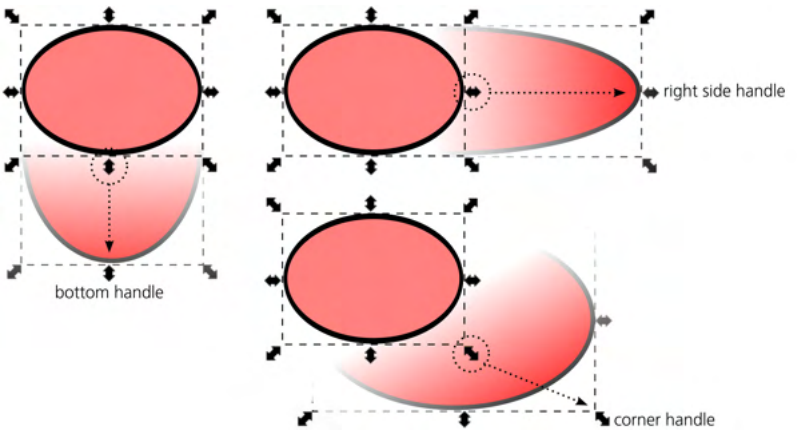


Figure 6-2: Scaling with the Selector tool

By default, the corner handles can move freely in any direction. This means that the ratio between width and height (also called *aspect ratio*) of the selected objects may *not* be preserved. You can stretch or squeeze your selection, or you can make it taller *and* narrower, or lower *and* wider, in a single drag. Side handles also do not preserve the aspect ratio as they scale in one dimension only.

The simplest way to lock the aspect ratio is to scale with Ctrl. This makes scaling proportional for both corner and side handles. Another way to achieve this is via the lock toggle in the Selector controls bar above the canvas, between the W and H editable fields, as shown in Figure 6-3.

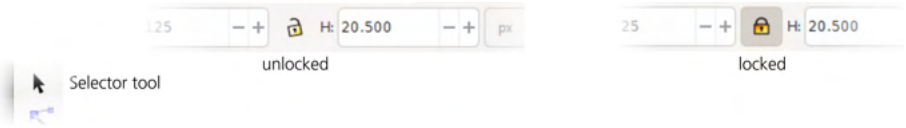


Figure 6-3: The aspect ratio lock in the Selector controls bar

When this lock is on (pressed), corner handles always scale proportionally. Side handles, however, are not affected by the lock and still scale in only one dimension.

Normally, scaling works in such a way that the opposite side (for side handles) or the opposite corner (for corner handles) of the selection's bounding box remain fixed. Sometimes, however, you will want the center of the selection to be fixed so that it is scaled symmetrically out from the center. This is what Shift does. To make this obvious, the fixed point of a transformation is always indicated by a cross mark, as shown in Figure 6-4.

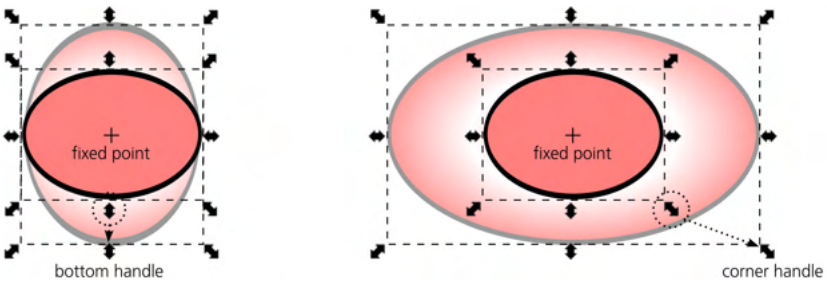


Figure 6-4: Scaling around the center of selection with Shift

NOTE *Here's a useful mnemonic. When typing text, pressing Shift makes the letters bigger (uppercase); when scaling, it also makes the result twice as big as it would be without Shift (since it scales on both sides of the center).*

You can combine Shift-scaling with Ctrl for ratio-constrained central-symmetric scaling. Not limited to the center or corners, you can place the fixed point of scaling anywhere, as you'll see in 6.4.

The remaining modifier, Alt, also has a role during scaling. It lets you scale the selection by integer multipliers: up to 2, 3, 4, . . . times the original size or down to ½, ⅓, ¼, . . . of the original size (along one or both dimensions). You can combine Alt with Ctrl to lock the aspect ratio and/or with Shift to scale around the center of the selection.

6.3 The Selector: Rotating and Skewing

How do I rotate objects? is a surprisingly common question on Inkscape users' forums and mailing lists. The trick is to switch the Selector tool into *rotate mode*, and here's how this is done: by a *second click* on the selection. (It must be a *distinct* second click, not a double-click.) A third click toggles the Selector back into scale mode, a fourth returns it to rotate mode, and so on, as shown in Figure 6-5.

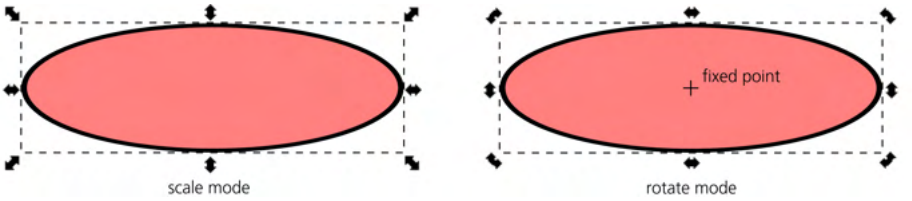


Figure 6-5: The scale and rotate modes of the Selector tool

There's one problem with the second click. Just as with moving, it may not always be possible to click the selection without selecting something else (if, for example, selected objects are in groups or under other objects). Instead of a second click, just press Shift-S to switch to the rotate mode or back.

NO SECOND CLICK IN ADOBE ILLUSTRATOR

Inkscape didn't invent this; the separate Selector modes are borrowed from the CorelDRAW and Xara vector editors. Not surprisingly, people who have problems with this convention tend to be Adobe Illustrator users.

Once you are in rotate mode, rotating the selection is as easy as dragging the corner handles. Dragging the side handles skews the selection.

NOTE *When your canvas is rotated (3.13), the scale and rotation handles may look weird because they stick to the original orientation; when the canvas is rotated 180 degrees, they even end up inside the selection's bounding box instead of outside of it. This seems to be just a cosmetic bug, however; no matter how it looks, the handles work as expected in the rotated coordinate system.*

When rotating, Ctrl constrains the rotation angle to fixed angle steps, by default every 15 degrees (Figure 6-6).

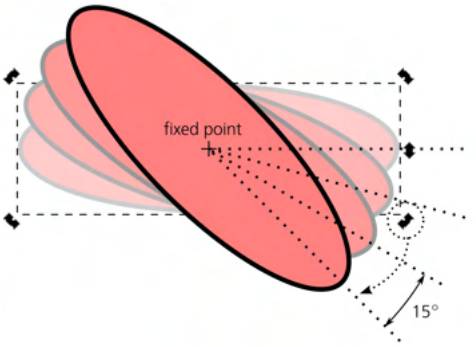


Figure 6-6: Rotation constrained by Ctrl

You can change this angle step in the Behavior ▸ Steps page of the Preferences dialog. In the Rotation snaps every drop-down menu, you can choose a value from 0.5 degrees to 90 degrees (all the values there are divisors of 360), or None for no constraint.

Skewing with Ctrl sticks to the same angle steps, as shown in Figure 6-7.

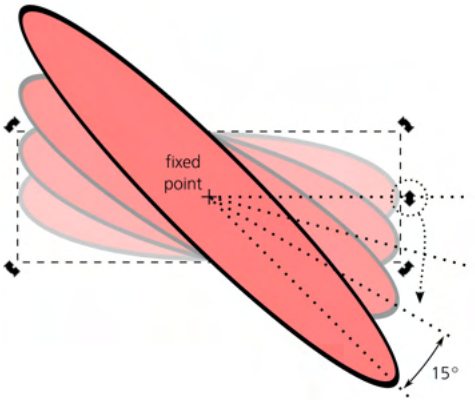


Figure 6-7: Skewing constrained by Ctrl

Shift-rotation temporarily moves the fixed point of rotation to the opposite bounding box corner. For example, Shift-rotating the top-right handle makes the bottom-left corner fixed.

6.4 The Fixed Point

Once you switch the Selector tool into rotate mode, you'll see a cross mark in the center of the selected object or objects. This is the *fixed point* of rotation. This point is remembered for each object as one of its permanent properties—moving it is an undoable action, and fixed points of all objects are saved with the document. Scaling, rotating, and skewing are performed around this point not only with the Selector tool but with most other methods as well (such as in the Transform dialog or when transforming by keyboard shortcuts).

NOTE *In Inkscape's UI (and in the previous edition of this book), the fixed point is called the "center of rotation," which is doubly confusing as it need not be in the center and it applies not only to rotation.*

By default, the fixed point is in the geometric center of the bounding box of an object. With the Selector tool, you can freely drag it to any location (inside or outside the object itself). When dragged, the fixed point snaps to the edges of the object's bounding box, to the geometric center (its original location), and to the horizontal and vertical axes going through the geometric center. This makes it easy to snap the fixed point back into the original position or to a corner of the bounding box. Drag it with Shift to suppress snapping; drag it with Ctrl to limit its movement to horizontal/vertical axes.

If you move the object (by any means, not only by dragging with Selector), its fixed point moves along with it, so it always stays in the same relative position to the object. There's no way to move an object's fixed point using the keyboard—you can only drag it with the mouse.

When more than one object is selected, the selection as a whole also has a fixed point. By default, of course, it is in the geometric center of the bounding box. However, if the object that you selected *last* (if you've been adding to selection with Shift-clicks) or the one that is the *topmost* in the z-order (if you've

selected by rubber band or by Ctrl-A) ever had its fixed point moved from the default position, then the entire selection will have the same fixed point as that object.

If, with multiple objects selected, you drag the selection's fixed point, it applies to *all* selected objects—each of them will have this new fixed point position. For example, if you draw a wheel with multiple spokes, you can select all the spokes and move the fixed point to the center of the wheel just once. After that, even if you select a single spoke, it will conveniently rotate around the center of the wheel.

Also, the fixed point is inherited when you duplicate or clone (16.1) an object. For example, you can draw a single spoke, move its fixed point to the center of the wheel, then go on duplicating (or cloning) that spoke and simply rotate each copy using any method (dragging corner handles with the Selector tool, keyboard shortcuts, or the Transform dialog). The new spokes will remain firmly set inside the wheel, as shown in Figure 6-8.

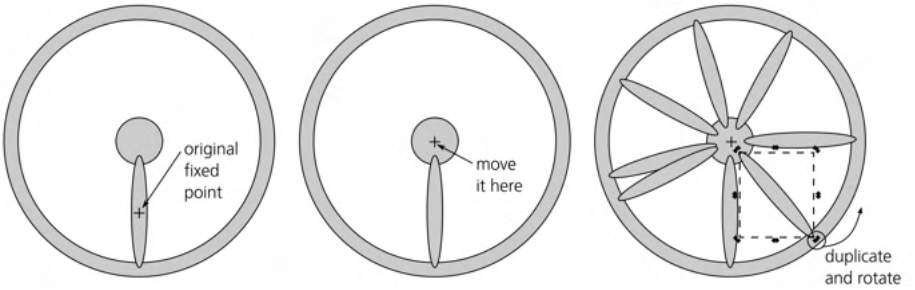


Figure 6-8: Using a fixed point to ensure spokes belong to the wheel

6.5 Transforming with Keyboard Shortcuts

Transforming objects by dragging them or their handles on the canvas is easy and intuitive, but it's not very precise (even when using various constrained modes) and may sometimes feel quite clumsy. Many long-time Inkscape users prefer to use keyboard shortcuts for most of their transformations.

NOTE *You can move, rotate, and scale from the keyboard, but you cannot skew (skewing is not used all that often, though).*

Keyboard shortcuts for transforming objects are easy to remember and, often, can make your work a lot easier. Another reason to learn these shortcuts is that, as you will see in the rest of the book, they are used consistently in many other tools and contexts to perform analogous functions—for example, to transform nodes in the Node tool, text characters in the Text tool, or gradient handles in the Gradient tool.

6.5.1 Moving

The ←, →, ↑, and ↓ arrow keys move the selection. The amount of move depends on the modifiers.

- Without modifiers, arrow keys move selection by 2 px (2 SVG pixel units, not screen pixels, A.6). You can change this default value in the Behavior ▸ Steps page of the Preferences dialog.
- With Alt, arrows move selection by 1 screen pixel (not SVG pixel). This means the actual distance will depend on the zoom level—you can do finer moves when you are zoomed in or coarser moves when zoomed out. This is one of the most useful shortcuts because of its precision and adaptability: with Alt, you can move your selection by the minimum distance that is still noticeable at the current zoom.
- With Shift, arrows move selection by 10 times the distance that they would without Shift. So simple Shift-arrows move by 20 px (by default), and Shift-Alt-arrows move by 10 screen pixels at the current zoom.

The ease and predictability of the keyboard move commands make them useful in a lot of different situations. For example, sometimes I need to do something to objects obscured by a large foreground object. Alt-clicking to "select under" works but is not super convenient; moving the foreground object to a new layer and hiding that layer is also a good option but quite cumbersome. In such cases, I often select the foreground object and move it off (to the right) with several → or Shift-→ keystrokes. When finished, I will select it again and move it back into place with the exact same number of movement keystrokes in the opposite direction. (Do not use Alt-arrows in this case because you may be at a different zoom level when moving the object back, and that would affect the distance.) This "move away and then back" trick is also handy for analyzing complex compositions when I'm trying to figure out which parts of the drawing correspond to which objects.

6.5.2 Scaling

The < and > keys (angle brackets) scale a selection down and up, respectively. Keyboard scaling always preserves aspect ratio and is performed around the geometric center of an object's bounding box (not around its movable fixed point). Again, the amount of scaling depends on the modifiers.

- Without modifiers, the < and > keys scale so that the larger of the two dimensions of the bounding box grows by 2 px (2 SVG pixel units, not screen pixels). For example, if the object is wider than it is tall, its width grows by 2 px (the left and right edges of the bounding box move by 1 px each in opposite directions), and its height grows by a proportionately smaller amount. You can change the default value of 2 px in the Behavior ▸ Steps page of the Preferences dialog.
- With Alt, the < and > keys scale so that the larger of the two dimensions of the bounding box grows by 2 screen pixels (not SVG pixels). The edges of the bounding box move by 1 screen pixel each in opposite directions. As with Alt-arrows, this means the actual scale depends on the zoom level; you can do finer scaling when zoomed in or coarser when zoomed out.

- Shift has no effect on the < and > keys. This is because on some keyboards, you need to press Shift just to get those characters. By the way, the , (comma) and . (dot) work as < and >, respectively, because on many keyboards they are physically the same keys.
- With Ctrl, the < and > keys scale by a factor of two—that is, they make the selection twice as small and twice as large, respectively. This is convenient when you need to scale something by a large ratio; for example, start by pressing Ctrl-> a few times to get within the ballpark of the required size, then adjust it precisely using the same key with Alt or without modifiers.

6.5.3 Rotating and Flipping

The [and] keys (square brackets) rotate selection counterclockwise and clockwise, respectively. Rotating is performed around the fixed point that is visible in Selector's rotate mode; unless you moved it, it's the geometric center of the object's bounding box. Again, the amount of rotation depends on the modifiers.

- Without modifiers, the [and] keys rotate by 15 degrees. This is the same angle step that is used by mouse rotation with Ctrl. You can change it in the Rotation snaps every drop-down menu in the Preferences dialog.
- With Alt, the [and] keys rotate by such an angle that the bounding box corners of the selection move by 1 screen pixel. This means the actual rotation angle depends on the zoom level, and you can make finer rotations when zoomed in or coarser when zoomed out.
- Shift has no effect on the [and] keys, for the same reason it has no effect on the < and > keys.
- With Ctrl, the [and] keys rotate by 90 degrees (one quarter of a full circle).

Also, some common transformations are accessible both as keyboard shortcuts and as buttons on the Selector tool's controls bar (Figure 6-9):

- Rotate selection by 90 degrees counterclockwise and clockwise (also accessible via Ctrl-[and Ctrl-]).
- Flip (mirror) selection horizontally and vertically (also accessible via the H and V shortcuts).

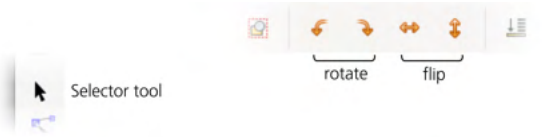


Figure 6-9: The Selector tool's controls bar: transformation buttons

To make two objects exchange places, use the *double flipping* trick. First, select both objects and flip them as a whole. Then, select each one in turn and flip it individually to restore it. To make the exchange two-dimensional, simply perform

this procedure twice, once with horizontal flipping (H) and then with vertical flipping (V).

NOTE *Flipping is equivalent to scaling by –100 percent in the respective dimension.*

Artists sometimes use flipping to check their work. It's easy to grow progressively more blind to errors in your drawing—simply because you're looking at it for so long. That's when flipping the entire drawing horizontally or vertically gives you a fresh look and makes many problems with shape, balance, or composition painfully obvious. You can also use canvas rotation (3.13) for the same purpose.

6.6 Transforming with Numbers: X, Y, W, and H

Sometimes, neither mouse nor keyboard transformation methods are quite up to the task because you want to move objects a specific distance or to a specific point, or rotate them a specific angle. Inkscape allows you to specify exact numeric values for transforming.

One way to do this is via the Selector tool's controls bar, as shown in Figure 6-10.

Figure 6-10: The Selector tool's controls bar: the X, Y, W, and H fields

Here, the X and Y values give the position of the selection, and W and H give its width and height. These values automatically update as you select and deselect objects or move or scale the selection; they also allow you to type in any values to move or scale the selection correspondingly.

NOTE *By default, the coordinate origin for the X and Y values is in the top-left corner of the page, but you can flip it to the bottom left (4.2).*

After typing in a value, press Enter to activate it, or press Tab to activate it and move to the next field. To jump to the first field from the keyboard, press Alt-X; to quit the field without editing, press Escape.

To the right of the editable fields, a drop-down menu selects the unit in which the values are expressed. Supported units include in (inches), pt (points, each point being 1/72 of an inch), mm (millimeters), and cm (centimeters). The default unit is px (SVG pixel, A.6, which is 1/96 of an inch now, but it was 1/90 of an inch in Inkscape versions before 0.92).

One of the most useful units is the % (percent). It lets you scale an object not to an absolute size but as a percentage relative to its current size. For example, to scale the selection 1.5 times up, switch the unit to **%** and type **150** in the **W** and **H** fields (or, if you click the lock button between them, only in one of them).

6.7 The Transform Dialog

The most powerful tool for transforming objects numerically is the Transform dialog (Shift-Ctrl-M). It has separate tabs for each of the four transformation types we've been discussing (moving, scaling, rotating, and skewing) as well as a tab for the complete transformation matrix. Generally, you choose one of the tabs, type the values you want, and click **Apply**.

Let's look at these tabs in order.

6.7.1 The Move Tab

We'll begin with the Move tab, shown in Figure 6-11.

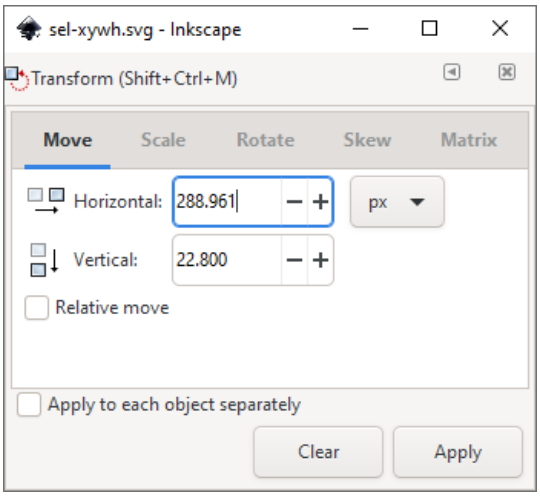


Figure 6-11: The Move tab of the Transform dialog

Unlike the X and Y values in the Selector controls bar, which always show you the absolute coordinates of the selection, in the Move tab, you can view and specify either absolute coordinates or relative displacements. By default, the Relative move checkbox is on, so the Horizontal and Vertical fields display zeros, and any numbers you type there will move selection *by* that much (for example, the values of `3` and `0` move the selection by 3 units horizontally to the right).

Now, uncheck the **Relative move** checkbox. You will see that the fields now display the current coordinates of the selection—the same as the X and Y fields in the Selector controls bar. The units drop-down menu also works similarly.

6.7.1.1 Moving to Adjust Intervals

If several objects are selected and both the Apply to each object separately and Relative move checkboxes are turned on, each object will be shifted, not relative to its own previous position, but relative to its neighbor—the nearest selected object to the left (for X) or below (for Y). This makes it possible to "space out" or "tighten up" collections of objects.

For example, if you need to space out a horizontal row of objects, select them and move them horizontally by 5 px with both checkboxes checked. The leftmost object will shift by 5 px to the right, the next one by 10 px, and so on until the rightmost selected object is displaced by 5*n* px where *n* is the number of selected objects. As a result, each interval between adjacent objects will increase by 5 px, and the whole row will be spaced out, much like how a letterspacing adjustment spaces out a text string (15.4.5). Moving these objects by –5 px will, conversely, squeeze them tighter together, the leftmost one moving by 5 px to the left, the next one by 10 px, and so on. For vertical moves, the effect is the same except that it starts from the object closest to the top (that is, the object with the smallest Y coordinate).

NOTE *The order in which objects are shifted may not be obvious if they overlap. The rule is that either the left edges (for horizontal moves) or tops (for vertical moves) of the objects' bounding boxes are sorted to determine which object to move relative to which. The order in which you selected the objects or their z-order makes no difference.*

6.7.2 The Scale Tab

Next we'll look at the Scale tab, shown in Figure 6-12.

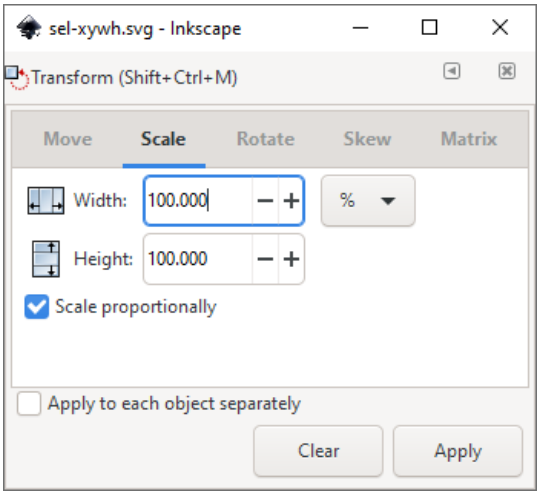


Figure 6-12: The Scale tab of the Transform dialog

Here, the default unit is %, which allows you to scale selection by a given ratio (such as 200% to scale it up twice or 50% to scale it down to half the size). The units drop-down menu also contains all the absolute units you may need (switching to them changes the displayed values from 100% to the width and height of the selection in the chosen unit). The Scale proportionally checkbox is analogous to the lock button on the Selector's controls bar.

So far, everything this tab can do is also possible via the Selector tool's W and H controls. However, the Apply to each object separately checkbox is unique to this dialog. It applies the same scaling to each selected object, scaling it around its own fixed point (6.4) instead of scaling the selection as a whole around its center. For the % unit, this means scaling each selected object by the same ratio; for all other units, this results in all selected objects getting the same specified width and height.

Unlike the Move tab (and the Selector controls), values on the Scale tab do not update automatically when you transform a selection by other means (such as by dragging handles), nor when you just select a different object: the values you typed stay put. This is handy if you want to apply the same size to many different objects, but sometimes this can be a problem—you may assume that the displayed value in absolute units is that of the currently selected object when that's not the case. To reset the values back to 100% of the current selection, click the **Clear** button.

6.7.3 The Rotate Tab

Now we'll move on to the Rotate tab, shown in Figure 6-13.

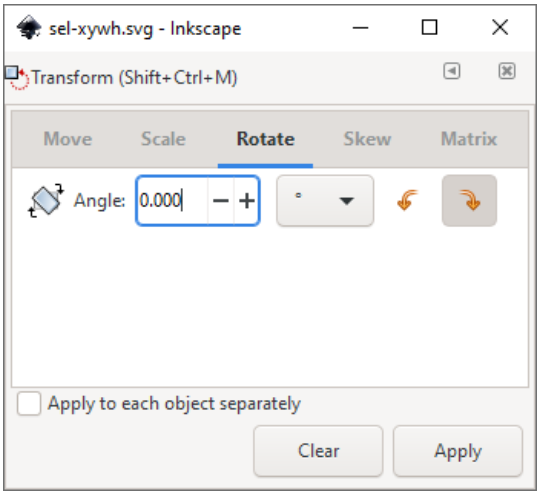


Figure 6-13: The Rotate tab of the Transform dialog

The Rotate tab contains a single editable field for the rotation angle. Positive values rotate counterclockwise; negative values rotate clockwise. The default unit is the degree (360 degrees to a full circle), but you can switch it to radians ($2 \times \pi = 6.283$ radians to a full circle). The Apply to each object separately checkbox works as expected: instead of rotating the entire selection around the selection's fixed point (6.4), it rotates each object separately around its own fixed point.

6.7.4 The Skew Tab

Next is the Skew tab, shown in Figure 6-14.

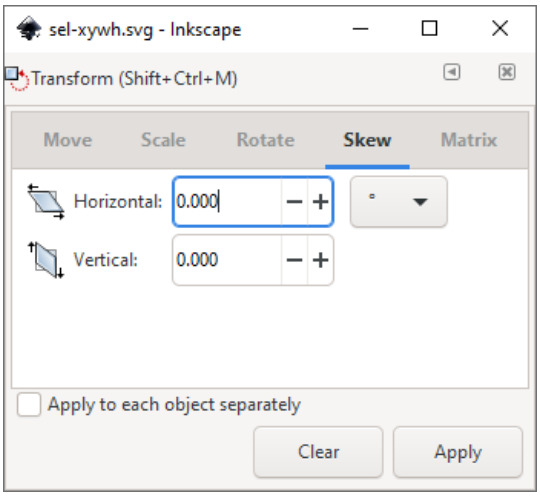


Figure 6-14: The Skew tab of the Transform dialog

The Skew tab contains two editable fields for horizontal and vertical skewing, which basically means displacing one of the four sides of the bounding box along itself, slanting the adjacent perpendicular sides. The Apply to each object separately checkbox works the same as for scaling and rotating, skewing each object around its own fixed point.

The units drop-down menu contains absolute length units, percent, and angular units (degree and radian). Here's how they work:

Absolute units
: With an absolute unit, the Horizontal value specifies the absolute displacement of the top edge of the selection to the left (positive) or to the right (negative); the Vertical value specifies the displacement of the left edge up (positive) or down (negative).

Percent
: With the % unit, the amount of displacement is calculated as the given percentage of the adjacent perpendicular side of the bounding box. In other words, this percentage is equal to the *tangent of the skew angle*. For example, skewing an upright rectangle vertically by 100% results in its formerly horizontal sides becoming slanted by 45 degrees.

Angular units
: This allows you to set the slant angle directly —that is, the angle of the sides adjacent to the side being moved. For example, vertically skewing an object by 45 degrees is the same as skewing it by 100% or by the absolute value of the object's width. Skewing by 90 degrees does not work, because it would create an infinite-sized object.

6.7.5 The Matrix Tab

Finally, we'll look at the Matrix tab, shown in Figure 6-15.

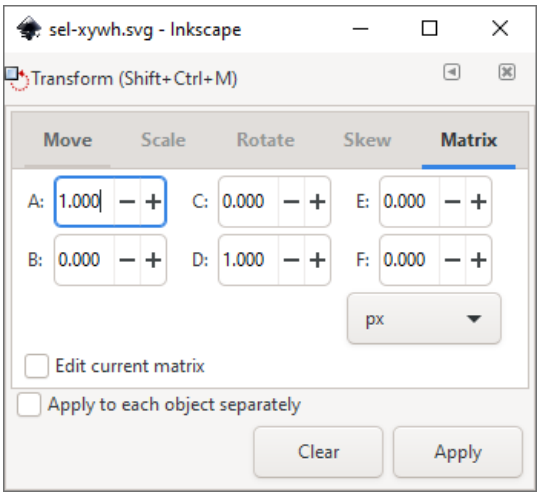


Figure 6-15: The Matrix tab of the Transform dialog

This tab allows you to edit the object's *transformation matrix*, which is stored in its `transform` attribute (A.7). A detailed explanation of the algebra of transformation matrices is outside the scope of this book; for practical purposes, it's enough to know that the two rightmost values (labeled E and F) represent the displacement of the object (how far it has been moved from the origin), and the four other values collectively encode its scaling, rotation, and skew.

By default, the Edit current matrix checkbox is unchecked and the tab presents an *identity matrix* where A and D are 1 and all other values are 0. If you change any values and click Apply, this matrix will be *post-multiplied* with the current matrix of the object—that is, applied to the selected object on top of its current transformation. If you check the Edit current matrix checkbox, the tab will show and let you directly edit the current matrix of the selected object. (If more than one object is selected, the matrix of the first or bottommost selected one will be displayed.)

Here's how you can reset the `transform` attribute of an object without going to the XML Editor. Check **Edit current matrix**, click **Clear** (this resets the values to the identity matrix), and then click **Apply**.

6.8 Pasting Sizes

A quick way to assign a specific size to an object without using any dialogs is by *pasting* that size on it. First, copy (Ctrl-C) an object whose size you want to assign to other objects. Then, use the commands from the **Paste Size** submenu in the **Edit** menu.

The commands of this submenu perform various combinations of pasting:

- Pasting width only (height is unchanged), height only (width is unchanged), or size (both width and height).
- Pasting to the selection as a whole or to each selected object separately (analogous to the Apply to each object separately checkbox in the Transform dialog).

For example, if you imported a number of bitmap images and want to create a gallery of thumbnails out of them, you can unify their sizes by drawing a rectangle of the desired thumbnail size, copying it to the clipboard, selecting all the images, and choosing Edit ▸ Paste Size ▸ Paste Size Separately.

6.9 The Measure Tool

[1.1] To transform by numbers, you first need to know what these numbers are—you need to *measure*. As you've seen, selected objects constantly measure themselves, reporting their coordinates and dimensions. But Inkscape also has a versatile Measure tool (M, the ruler icon on the toolbar on the left) that will cover more of your measurement needs than you ever thought existed.

6.9.1 Hovering

The first thing you'll notice, even before you click anything, is that the tool displays basic information about any objects you *hover* upon. This means you don't need to select an object to look up its size or X/Y coordinates—just switch to the Measure tool, move your mouse, and read it from your screen.

Just as in Selector, the tool by default treats groups as objects, reporting information about the group. If you need to look up an object inside a group, simply press Ctrl (again, no clicking, just Ctrl-hover). For paths (Chapter 12) and shapes (Chapter 11), the tool also reports the Length of the path, as shown in Figure 6-16.

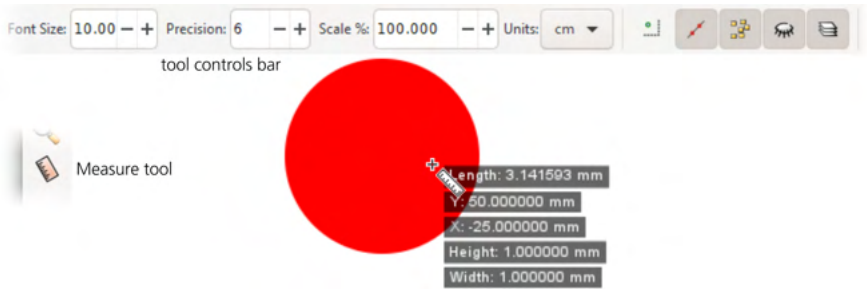


Figure 6-16: Using the Measure tool: the Length of a circle with the diameter of 1 equals π.

The first few controls on the tool's controls bar let you adjust aspects of the overlay display: its Font size (which does not depend on zoom), Precision (number of fractional digits) of the numbers, Units of measurement, and the Scale (default 100%), which allows you to see all values scaled by some coefficient. In Figure 6-16, I upped the precision to 6 to get more digits of π.

6.9.2 Dragging

Imagine a spring-loaded measuring tape: you hook it at some point and stretch it to another point, then you read the distance between them. That's exactly what you do in the Measure tool when you click and drag. In addition to the *length*, it also displays the *angle* (red arc) of your measuring tape (blue line) from the horizontal (red line), as shown in Figure 6-17.

Figure 6-17: Measuring the distance and angle between two points

After you release mouse, the measuring tape you just pulled out stays put on the canvas; its ends become two round markers that you can freely (or constrainedly with Ctrl) drag around and see the numbers continuously update. If you want to set precise coordinates for one of the handles, Shift-click it and enter the coordinates in the position dialog that pops up. You can also swap the start and end of the measurement line with the Reverse measure button on the controls bar.

The whole measurement overlay disappears when you switch to another tool—but it will pop back up when you go back to the Measure tool. Click-and-drag somewhere else to replace the old measurement with a new one; I have found no way to "retract the tape"—to simply remove the existing measurement.

6.9.3 Measuring and Constraining Angles

The angle of your measurement line from the horizontal is readily available; measuring arbitrary angles between lines (neither of which is horizontal) is possible too but a little tricky. If you have two lines and want to measure the angle between them, start at their intersection and drag along one of the lines. Then, press and release Ctrl (without releasing mouse); this will reset the base of the angle measurement to the current direction. Then, drag to the second line and read the angle.

Dragging with Ctrl, as expected, constrains the angle of your measurement line to angle steps (go to **Behavior ▸ Steps** in **Preferences** to change). However, since pressing Ctrl *while dragging* resets the base of the angle measure, you need to press and hold Ctrl *before* you start to drag if you want to constrain (for example, to measure strictly horizontally or vertically).

6.9.4 Measuring Segments

The Measure tool can do something you can't easily do with a measuring tape: measure multiple distances along the same line. If you let your measurement line cross any objects, you will see that each intersection divides the line into segments, and the tool reports the length of each segment.

For example, if you cross an uppercase *B* with a vertical measurement, you will get as many as six separate readings: the widths of all three horizontal stems, the gaps between them, as well as the overall height of the letter, as shown in Figure 6-18. (At the bottom, it also reports the total length of the measurement line, but you're not interested in that because you could start and end it in random points so long as it crosses what you want to measure.)

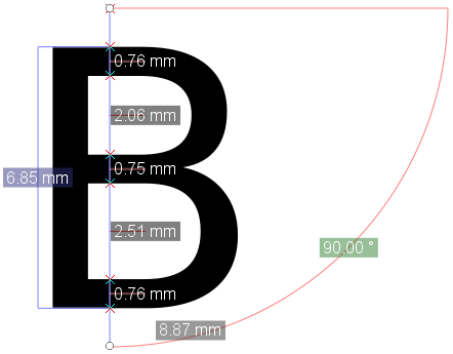


Figure 6-18: Measuring the stems and gaps of a letter shape: the red X marks are the intersection points; each segment shows its length.

This also works for any number of different objects that your line crosses: the tool will report segment lengths for each one it encounters, covering their widths as well as the gaps inside and between objects. When crossing stroked (Chapter 9) paths or shapes, the tool reports only intersections with the path centerline, not with the visible edge of a stroke.

You can create a measurement line in some strategic axis of your drawing and just leave it there as you keep working (it is not shown in other tools, so it does not get in the way). At any time, switch to the Measure tool to read updated numbers for the moved or newly created objects that cross your axis.

On its controls bar, the tool has a number of options for how it treats intersections, as shown in Figure 6-19.

Figure 6-19: The Measure tool: options for intersections

- You may want to reduce clutter by reporting only intersections and lengths for *selected objects*, as opposed to all objects (the default).

- By default, the tool does not show lengths of the *first and last segments* because they depend on the position of your measurement line's ends, which may be somewhat arbitrary; you can change that.
- The third button controls reporting the lengths between and inside objects; by default it's on, but if you turn it off, you will see only the first and last intersections on the line and the length between them.
- The fourth button controls whether the tool marks *hidden intersections*. Here, "hidden" does not refer to hidden objects (4.1) but to those intersections that are obscured (where they cross the measurement line) by other objects on top of them. By default, hidden intersections are shown, but you can hide them to reduce clutter.
- Finally, the fifth option lets you limit intersections and measurement to the current layer; by default, the tool looks at objects in all visible layers.

6.9.5 Phantom Measurements

Whenever you drag to start a new measurement, the old one disappears. What if you want to have two or more measurements visible at the same time? That is possible if you turn the current measurement into a *phantom* by clicking the button with the camera (that is, "taking a snapshot") on the controls bar.

A phantom measurement behaves almost like a normal measurement, except you cannot drag its ends and it is shown in uniform gray instead of red/blue. Unfortunately, unlike the current "live" measurement, phantoms do not survive switching to another tool and back, which makes them not too useful in practical work.

6.9.6 Creating Measurement Objects

The constructions that the Measure tool creates—both the current measurement and any phantoms—are not part of your document. It's just an informational overlay. But if you want to have a real flesh-and-blood object that shows a measurement, the Measure tool can do that, too.

Before going into objects, however, let's look at *guides* the Measure tool can also create for you. Guides (7.1) are not quite real either in that they are not printed and not shown by SVG viewers outside Inkscape; however, they are definitely more real than Measure tool's machinery because they are visible in any tool (though you can hide them) and are saved/restored with the document. When you press the To guides button on the controls bar, the current measurement generates a bunch of guides—one for the measurement line itself, two horizontal/vertical crosses for its start/end points, and one slanted for each of the intersections, as shown in Figure 6-20.

Figure 6-20: The Measure tool can generate guides.

The Measure tool's guides are purple-colored (unlike the regular guides that are blue by default) and are helpfully labeled. Still, if you convert more than a couple measurements to guides, your canvas may look rather messy, with bundles of guides continuing infinitely in all directions. In many cases, if you want to save your measurement to posterity, it is better to convert the whole shebang to an object. That's what the next button on the controls bar, Convert to object, does (Figure 6-21).

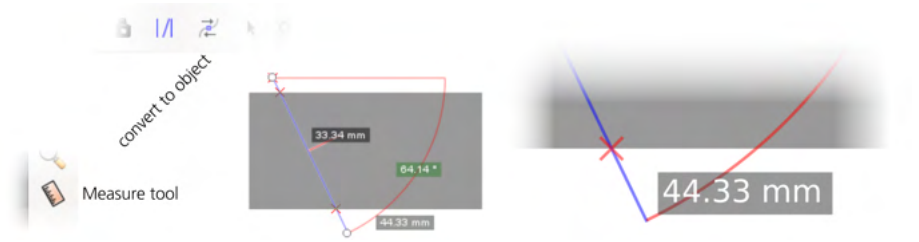


Figure 6-21: The Measure tool can generate objects, too!

At first blush, the newborn object is undistinguishable from the measurement overlay: it faithfully reproduces all the lines, cross marks, and semitransparent labels. And yet it is an object—you can drag it around, zoom into it, ungroup, delete objects inside its group, and so on.

The next button, Mark dimension, presents yet another approach to object creation. Instead of trying to mimic the Measure tool's overlay, it creates a plain, solid black dimension marking with arrowheads plus a text object giving the length (Figure 6-22). This applies only to the overall length of the measurement and ignores any intersections or angles. The only parameter you can adjust is the distance from the measurement line where this object will be placed (Offset on the controls bar, always in px units).

Figure 6-22: The Measure tool can generate a dimension marking.

6.10 Transforming with the Tweak Tool

The Tweak tool (W or Shift-F2) is a versatile instrument that applies the same "soft brush" metaphor not only to transforming objects but also to styling them (8.9) and to path editing (12.6). This tool has a number of *modes*; in this chapter, I describe those modes that deal with transforming, duplicating, and deleting objects. Using these modes, you can *draw by scattering*—sweep, sway, or sculpt fields of objects, such as Spray tool effusions (4.7) or clone tilings (16.6), into complex and naturalistic images.

All modes of the Tweak tool share a number of common features. In all of them, you use a circular soft-edged brush (the orange circle you see under the cursor) controlled by the Width and Force parameters on the controls bar and optionally affected by the pen pressure (if you have a pressure-sensitive tablet). You "paint" with this brush over the selected objects to act on them.

The *width* of the brush varies from 1 to 100, but these aren't absolute units; at any zoom, 1 gives you a very small brush and 100 gives you a brush approximately the size of your screen. The tool shows the current brush width as a circular brush outline centered on your mouse cursor. To change the width, use the **Width** control on the toolbar or the ← (narrower) and → (wider) keys. The brush is soft; its action peaks in the center and smoothly decreases toward the edge following a bell-like profile.

Similarly, *force* ranges from 1 (very weak—you need to stroke an object several times to get a visible effect) to 100 (very strong—the first stroke of the brush applies the maximum effect). To change the force, use the toolbar control or press the ↓ (weaker) and ↑ (stronger) keys.

To get a feeling of how the Tweak tool works, create a number of small objects in a test drawing—for example, with the Spray tool (4.7) or just by manually duplicating (Ctrl-D) or cloning (Alt-D) an object and dragging away the copies. Now, select all the objects you created, switch to the Tweak tool, adjust the **Width** so the circle covers several objects at once, set **Force** to a moderate value of 20, enable one of the modes (described below), and start dragging over the objects. Figure 6-23 shows what will happen.

Move mode

Moves the selected objects under the brush in the direction in which you drag the brush. This is not entirely unlike just dragging these objects around—except you're not dragging them: your brush softly "sweeps" the objects it runs across. Note that the Tweak tool only ever affects selected objects (even though it doesn't show the selection cue); it's usually best to separate the objects you're tweaking often into a layer, so you can make it current and select them all with Ctrl-A.

Attract/Repel Objects mode

Attracts the selected objects under the brush toward the cursor (default) or repels them away from the cursor (with Shift pressed). This way you can "pull in" (make denser) or "push out" (make sparser) different areas of your scattering of objects.

Figure 6-23: Transforming a grid of semitransparent objects with the modes of the Tweak tool

Move Jitter mode

Moves the selected objects under the brush in random directions by random distances. The harder you press (with a pressure-sensitive pen) and the higher the Force, the heavier the jitter is. This is a way to "shake up" your composition in places where it's becoming too uniform.

Scale mode

Scales the selected objects under the brush down (by default) or up (with Shift pressed). This way you can introduce smooth scale nonuniformities to your scattering—to make objects larger in some parts of the image and smaller in others.

Rotate mode

Rotates the selected objects under the brush clockwise (by default) or counterclockwise (with Shift pressed). This way you introduce rotation nonuniformities to your scattering—for example, to "bend" your scattering texture to match the dominant directions in different parts of the image.

Duplicate/Delete mode

Randomly duplicates some of the selected objects under the brush (by default) or deletes some of them (with Shift pressed). The chance that an object is duplicated or deleted, as always, depends on force and pen pressure. This completes the "scatter drawing" toolset by providing a way to thicken or thin out your scattering wherever necessary.

As with the regular Duplicate command (4.5), duplicating with the Tweak tool places the copies right over the originals, so it is a good idea to use the Jitter mode after duplicating to ruffle them apart. The duplicates created

by the tool are automatically added to selection if the original objects were selected.

6.11 What Transformations Affect

Now that you can transform your objects using all kinds of methods, let's look at what exactly gets transformed when you do it. As it turns out, Inkscape can optionally transform or not transform some specific parts or aspects of objects. This is controlled by the four toggle buttons on the right end of the Selector controls bar, as shown in Figure 6-24.

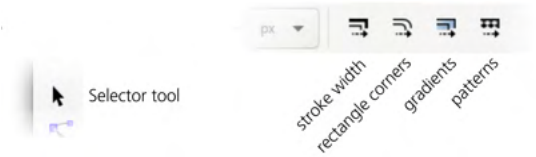


Figure 6-24: The Selector tool's controls bar: the toggle buttons for transformation options

Stroke width
: This option applies only to scaling and has no effect on moving, rotating, or skewing. When it's on, any scaled object that has stroke (8.2) will have the width of the stroke scaled by the same ratio. For example, if you scale an object with a 2-px-wide stroke to twice its current size and this button is on, the resulting object will have 4-px-wide stroke. If the button is off, it will have the same unchanged stroke width of 2 px.

  When you scale an object without preserving the aspect ratio, if this button is on, stroke gets scaled by the square root of the product of vertical and horizontal scale ratios. (I know!) A slightly simpler way to put it is that the stroke scale ratio will be the *geometric mean* of the horizontal and vertical ratios. For example, if you scale an object to twice its width horizontally but leave its height unchanged, the stroke becomes wider by a factor of 1.415 (the square root of 2).

  Keeping stroke width unscaled is more commonly useful. For example, in a plan or draft, you normally want all your objects to have the same fixed stroke width, unaffected by scaling, so you would toggle this button off. It only makes sense to turn this on when you're using stroke as a purely visual element (such as brush-like strokes in a freehand drawing).

NOTE *Profiled strokes that the Pen and Pencil tools create (Chapter 14) always scale their width with the object, even if this button is off.*

Rounded rectangle corners
: This option also applies only to scaling and has no effect on moving, rotating, or skewing. It controls whether Inkscape scales the rounded corners of rectangles (11.2.2). When it's on, a rectangle is scaled as a whole, just as it would be scaled if converted to path; this may lead to the rounded corner becoming larger, smaller, and/or nonround. When it's off, Inkscape preserves the rounded corners exactly as they were. Turning this off can be

useful, for example, in a flowchart diagram where you want all your boxes to have the same rounded corners regardless of their sizes.

Gradients
: This button controls whether transformations are applied to the gradients in objects' fills or strokes (10.1). Since gradients (more precisely, positions of gradient stops) can be moved, rotated, and skewed as well as scaled, this button applies to any kind of transformations. When it's on, the gradient stops are transformed as a whole with the object carrying the gradient. When it's off, the gradient stays glued to the canvas (unchanged in position, direction, or scale) while the object is transformed.

  For example, with this button toggled off, it is possible to move an object *out* of its own gradient or scale it up so that more of the gradient becomes visible. This is useful when the gradient's position is coordinated with other objects in your drawing whereas the object itself is simply a "window" onto that gradient, and you want to move or adjust the edges of that window without touching the gradient itself.

Patterns
: This button is analogous to the Affect gradients button except it applies to patterns (10.8) instead of gradients.

With a path, another way to "transform" it without affecting stroke width, gradients, or patterns (regardless of these Selector options) is by selecting all of its nodes in the Node tool and transforming the node selection (12.5.7.3).

NOTE *There's an annoying but harmless bug: regardless of the status of the transform options, interactive transforming of objects by mouse in the Selector tool always shows stroke, rounded corners, gradients, and patterns as fully transformed while you're dragging the mouse. As soon as you release the mouse, the display of the object is fixed.*

7

SNAPPING AND ARRANGING

As a sequel to the previous chapter on transformation, let's look at how Inkscape allows you not only to transform objects freely but to do it quickly, precisely, and, to some extent, automatically. The two main approaches are snapping and arranging.

Snapping refers to making some points, lines, or paths work as magnets, so that when you are moving something in the vicinity of such a magnetic attractor, the thing you're moving "snaps" into the exact desired position. *Arranging* moves a large number of objects in a regular way so that they are aligned, distributed, scattered, or lined up into a grid exactly as you want them to be.

7.1 Guides

The easiest way to snap objects into alignment is by creating a guide (guideline). *Guides* are visible, infinitely long straight lines on the canvas that you can use for alignment and snapping. Guides are not objects: they cannot be selected, they don't print, and they don't show up on bitmap exports. Although they are saved with the document, they are not part of the SVG standard and are invisible in

SVG viewers other than Inkscape. They are just Inkscape-specific helpers for organizing your artwork.

For example, guides may separate parts of your layout, helping you visualize the edges of various areas or cells. A guide can be a centerline so you can draw symmetrically around it. Or you can use a guide to verify whether objects are really aligned or just look like they are.

To create a guide manually, the canvas rulers must be visible; if they're hidden, display them with Ctrl-R. Click anywhere on a ruler and, without releasing the mouse, drag your cursor out to the canvas. A new guide appears that you can now drop (by releasing the mouse) where you want it. Naturally, dragging off the horizontal ruler creates *horizontal guides*, and dragging off the vertical ruler creates *vertical guides*; what's more interesting, dragging from the very ends of the rulers creates *diagonal guides*, slanted (by default) at 45 degrees, as shown in Figure 7-1.

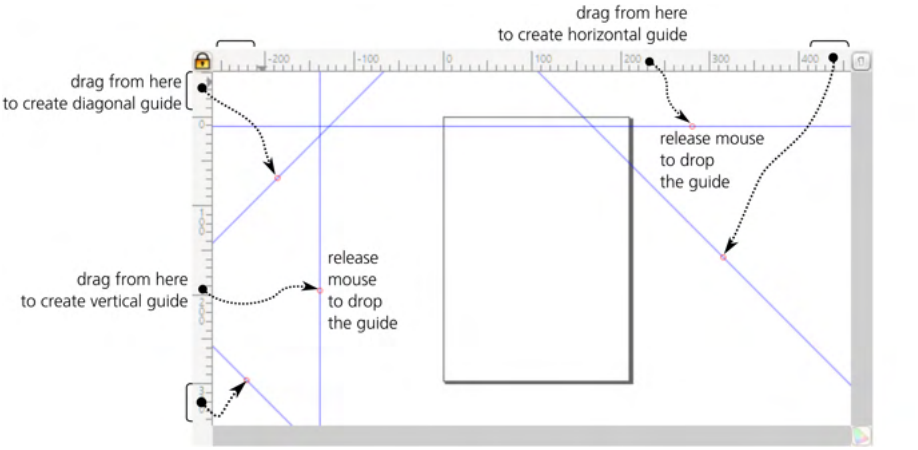


Figure 7-1: Creating guides by dragging from rulers

Moving a guideline is easy. With the Selector or Node tool, hover over a guide; notice that it becomes red, which means it's now grabbable. Click and drag the guide to where you want it to be. To delete a guide, simply drag it back onto a ruler and drop it there, or hover and press Delete while it is red.

To toggle visibility of all guidelines in the document, press the | key (this is the vertical pipe character, usually on the same key as the \ but requiring Shift). This key hides the guides without deleting anything; if you press | again (or try to create a new guideline), all hidden guidelines will be displayed again. This key controls both guide visibility and snapping to guides (so there's no snapping to invisible guides); if you want to disable snapping without hiding the guides, use the % key (the global snapping toggle, 7.3).

7.1.1 Guide Anchor

Each guideline has a special point called an *anchor*, which looks like a small circle. A guide's label, if any, is displayed next to its anchor. This is also the axis around which you can *rotate* a guide manually by Shift-dragging it.

Usually the anchor is the point where you last released the mouse when you created or moved the guide. It is also the point whose X/Y coordinates you view and edit in the Guideline dialog (see the next section). To slide the anchor along its guide without moving the guide, Ctrl-drag it.

When you're snapping to the guide, its anchor has magnetic power of its own. That is, when you're near the anchor, you will snap not just to the guide but precisely to its anchor.

7.1.2 The Guideline Dialog

As often is the case in Inkscape, in addition to dragging guidelines around manually, you can also use numbers to specify the precise position and angle of a guideline. This is done in the Guideline dialog (Figure 7-2), accessed by double-clicking any guideline.

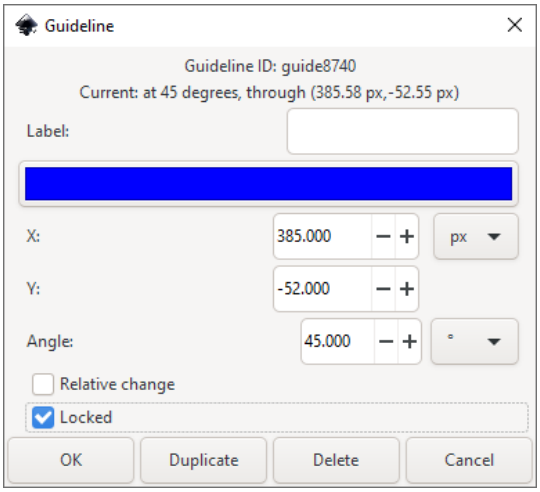


Figure 7-2: The Guideline dialog

- The Label is any text string identifying this guide. It will be displayed along the guide next to the anchor.
- Below the Label, a color swatch shows and lets you change (click it for a color picker) the color (but, for some reason, not the opacity) of the guideline.
- For horizontal guides, the Angle is 0, while Y determines the height of the guideline.
- For vertical guides, the Angle is 90, while X determines the horizontal position of the guideline.
- For diagonal guides, the Angle sets the slant (default is 45 degrees), while the X and Y values together determine the anchor point through which the guideline goes.
- The Relative change checkbox zeroes out all editable fields in the dialog; now, any value you type into them will *add* to the current values, not replace them.

- If you Lock the guide, it won't respond to hovering and you won't accidentally drag it away (and it won't prevent you from selecting objects in its vicinity). It will still be possible to double-click the guide to unlock it via this dialog, although you may need to do it over an empty canvas so you don't double-click some object instead.
- OK accepts your changes; the Duplicate and Delete buttons do what they say (a duplicated guide is initially placed on top of the original one); Cancel closes the dialog without change.

Note that the Guideline dialog is *modal*, which means it locks the rest of Inkscape while it is displayed.

7.1.3 Document Properties for Guides

The Guides tab of the Document Properties dialog (Shift-Ctrl-D) contains a few general options that affect all guidelines, as shown in Figure 7-3. These options are here rather than in the global Preferences dialog because they are considered local to the document and are saved with it, so that different documents can have different guide setups.

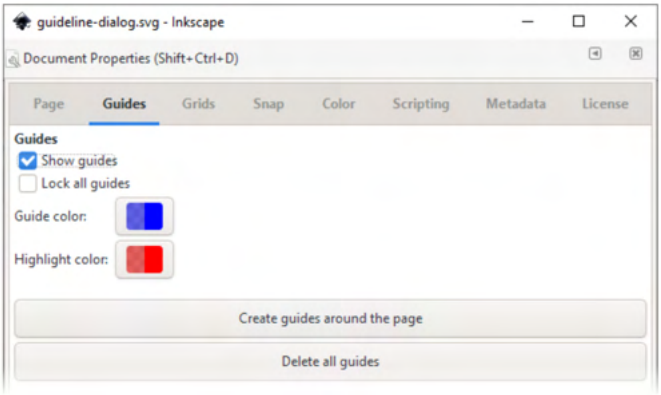


Figure 7-3: Guide settings in the Document Properties dialog

- The Show guides checkbox does the same as the | key: it toggles visibility (and snappability) of all the guides in the document, without deleting them.
- Lock all guides makes all guides unresponsive to hovering or dragging, which may make your work (after you've set up all the guides you need) much easier. Caveats: newly created guides will still not be locked (though you can relock them if you go back to this dialog and turn Lock all guides off and then on again); guides that you made locked individually (7.1.2) will be unlocked too whenever you turn off this checkbox.
- The Create guides around the page button generates four guides along the four sides of your canvas so you can easily snap or align to the page frame.
- The Delete all guides button will be a godsend when you've ended up with way too many guides in your document and want to start over (but don't want to lose the content).

- You can change the color and opacity of the guides in their normal state (default is half-transparent blue) and under mouse hover (default is half-transparent red). Clicking the color swatches opens a small color chooser dialog (Figure 7-4). The A slider sets the *alpha* (transparency). The color you set here affects only those guides for which you didn't set their own color (7.1.2) as well as new guides you create after the color change.

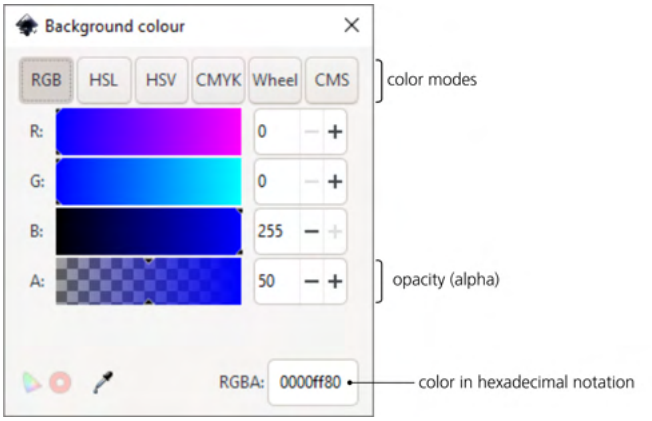


Figure 7-4: Color chooser dialog

7.1.4 Guides from Objects

You've already seen measurements (6.9.6) and the page frame converted to guides, but you can convert any object to guides as well. Just select one or more objects and choose **Object ▸ Objects to Guides** from the menu, or press Shift-G.

NOTE *By default, this destroys the selected objects and replaces them with a bunch of guidelines. To prevent this, toggle the* ***Keep objects after conversion to guides*** *checkbox in the* ***Tools*** *page of* ***Preferences****, or simply press Ctrl-D before the conversion.*

Figure 7-5 shows how this command works for different types of objects:

- For paths or rectangles, it replaces each straight path segment or rectangle side with a coincident guide. Thus, converting a rectangle to guides will create a rectangular area delineated by two pairs of parallel guides. To quickly create a slanted guideline through two given points, switch to the Pen tool (14.1.1), click one point, double-click the other (to finalize the straight line path), and convert this path to a guide.
- For 3D boxes (11.3), all 12 edges of a box are converted to guidelines.
- For everything else, the four sides of the object's bounding box are converted to two vertical and two horizontal guides.

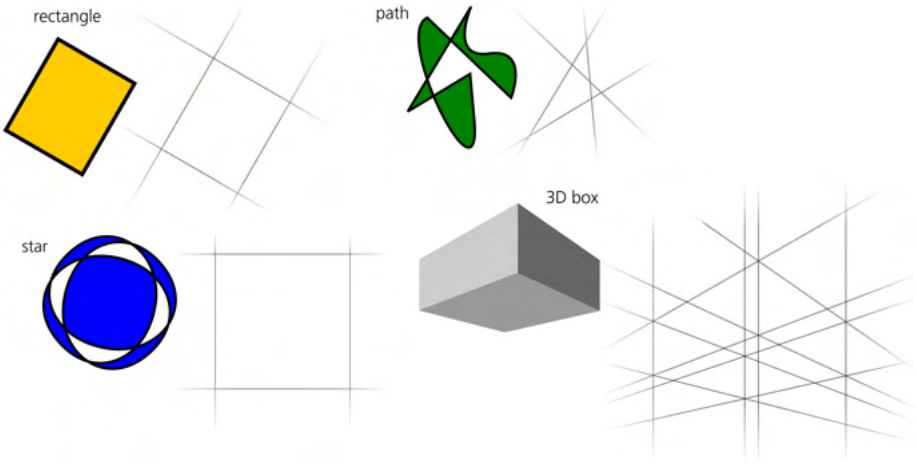


Figure 7-5: Converting various objects to guides

7.2 Grids

Think of a *grid* as a development of the idea of guides: it is a regular pattern of guides that covers the entire canvas. Grids are often used for snapping, alignment, equispaced distribution of objects, drawing to the pixel grid, or axonometric drawing (Chapter 24).

Inkscape supports two types of grids: *rectangular* and *axonometric* (Figure 7-6). One document can display more than one grid, which can be different in type and/or spacing. To create a new grid in your document, go to the **Grids** tab of the **Document Properties** dialog, select the desired grid type, and click **New**. A new grid will light up in the document.

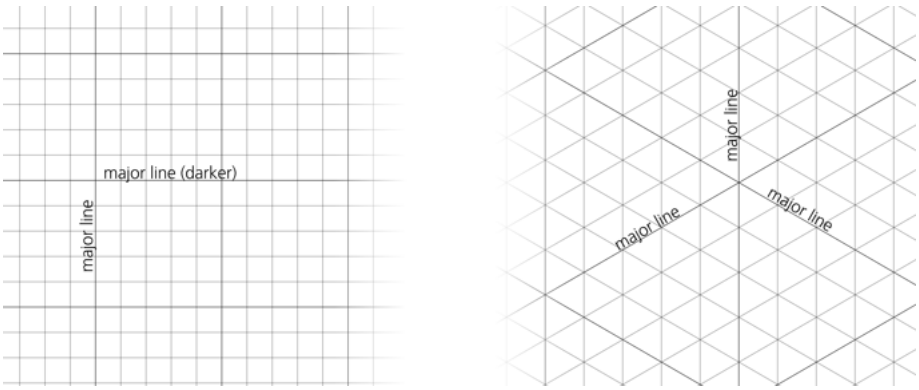


Figure 7-6: Rectangular and axonometric grids

As you can see, a rectangular grid consists of two perpendicular sets of guides, whereas the axonometric grid has three sets, one vertical and two others at an angle (by default at 30 and –30 degrees). The rectangular grid is the most common type; the axonometric grid is typically used for drawing pseudo-3D scenes without vanishing points.

You will notice that Inkscape keeps the visual density of the grid lines within certain limits independent of zoom. As you zoom out, the lines get closer to each other—but then Inkscape weeds the grid by showing only every tenth grid line. Such decimation of grid lines happens again and again as you zoom out so as to keep the grid display reasonable. Inversely, as you zoom in, Inkscape reveals more and more grid lines so as to fill your screen evenly.

New lines stop appearing only at a zoom level where *all* the lines are visible; the default rectangular grid spacing is 1 px. Just as a ruler where centimeter marks are larger than millimeter marks, Inkscape makes every fifth line (called a *major line*) a little darker than the others (*minor lines*); you can see this in close zooms.

To hide all grids in your document, press the # key; another # will display all hidden grids again. This key controls both grid visibility and snapping to grid, so you would never snap to an invisible grid; if you want to disable only snapping without hiding, use the % key (global snapping toggle, 7.3).

7.2.1 Grid Options

You can change some aspects of your document's grids in the Document Properties dialog. Once you create a grid, you will see a new grid properties panel at the bottom of the dialog, as shown in Figure 7-7.

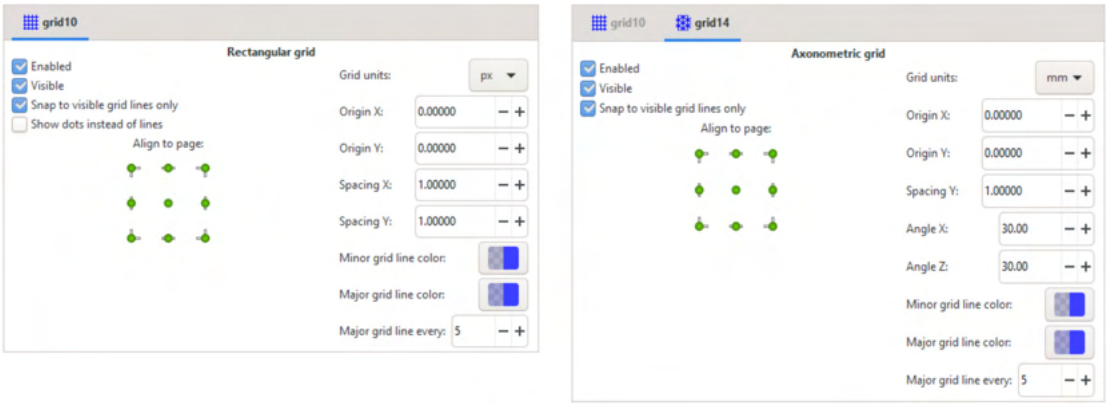


Figure 7-7: Grid properties for a rectangular grid (left) and for an axonometric grid (right)

You can create more than one grid in the same document, and by default, they will all be visible at once—although you can enable or disable any of them separately. In the dialog, each grid will have its own tab.

Here are the settings you can change for a grid:

- Enable or disable the grid (which basically means turning snapping to it on or off), and toggle whether it is Visible. Disabling a grid hides it, but hiding it does not disable it—so here, if you prefer, you can set up snapping to an invisible grid. Separately, you can allow snapping to visible grid lines only, ignoring those lines that are hidden due to the zoom level. For rectangular grids only, you can switch to showing grid intersections as dots instead of lines.

- The grid origin settings let you shift the grid as a whole. You can set the Origin X and Origin Y numerically, or you can use one of the nine Align to page buttons on the left to move the origin to one of the page's corners, mid-sides, or center.
- The Spacing settings define the density of the lines (remember, if you make the grid too dense, Inkscape won't show all the lines anyway until you zoom in close enough). For a rectangular grid, there are two spacing values, X (horizontal) and Y (vertical). For an axonometric grid, there's only one spacing value—the distance between the intersections along the Y axis; the intervals between the vertical lines are fully determined by this distance and the axonometric angles. All these values are, as usual, coupled with a unit selector (a drop-down list).
- Angles are available only for axonometric grids. These are the angles of the two sets of diagonal lines. By default, both are 30 degrees (measured counterclockwise from horizontal for Angle X and clockwise for Angle Z), but they can be anything from 0 (no vertical guides at all) to 90 degrees (careful, your canvas will be entirely covered with vertical guides!), and they don't have to be equal to each other.

  If you have added an axonometric grid to your document, new diagonal guides (7.1) will be created with the same angles as those used by the grid.
- You can set colors separately for the Major and Minor grid lines. By default they are both blue, but the minor lines are less opaque (so less visible).
- You can adjust the frequency of the major lines; by default, every 5th grid line is major.

You cannot "convert a grid to objects," but if you need a grid of real objects (which, unlike Inkscape's helper grids, other SVG software will also display), try the clone tiler (16.6) or **Extensions ▸ Render ▸ Grids** (19.2).

7.3 Snapping

Now that you know everything about guides and grids, let's look at what guides and grids are most often created for: snapping.

The idea of snapping is simple: Inkscape tries to place things where you *want* them instead of where you actually move them with your always shaky mouse or tablet pen movements. As soon as you move a *snappable* (something that can be snapped) close enough to an active *snap target*, it jumps right into place. Snapping is fundamentally interactive; it happens only when you drag something with your mouse (but not, for example, when you move objects with the arrow keys on your keyboard or use the Align and Distribute dialog).

Snappables can be entire objects as well as various nodes or points (path nodes, gradient handles, fixed point, and so on) inside objects. Snap destinations, apart from guides and grids, can be other objects and their parts (that is, objects can snap to objects, if you allow them). Snapping is a big topic—but it's not deep: once you get the basic idea, the rest is just remembering the numerous modes and toggles.

Snapping is extremely useful when you need it—but, admittedly, extremely annoying when you don't. If snapping gets in your way, just remember that there's a *master snap toggle*—the % key—that enables or disables all kinds of snapping in the entire program. (Here's a mnemonic: view the % character as two nodes about to snap to a slanted guideline.) Unlike the | (guides toggle) and # (grid toggle), this key turns snapping behavior on or off but does not hide or show anything. Also, in the Selector and most other tools, dragging objects or nodes with Shift temporarily disables snapping as you drag.

7.3.1 The Snap Controls Bar

Most of the snapping controls are collected in the *snap controls bar*, by default displayed vertically at the rightmost edge of the Inkscape window (Figure 7-8). Use **View ▸ Show/Hide ▸ Snap Controls Bar** to turn it on or off.

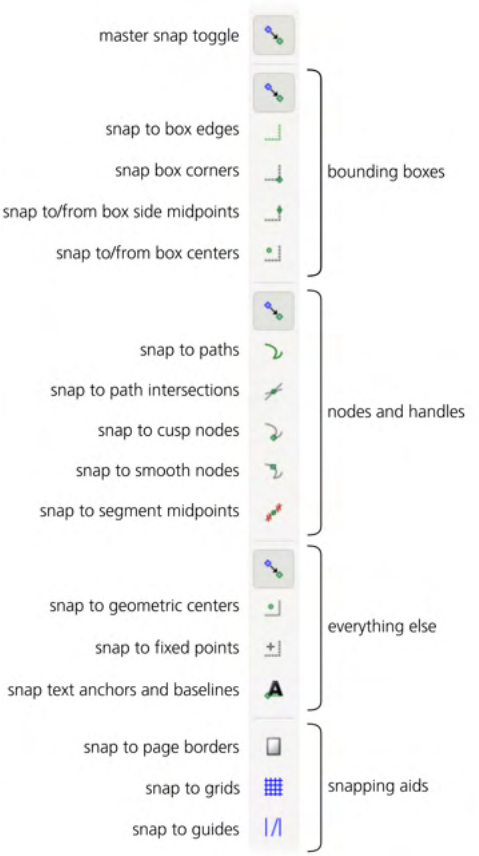


Figure 7-8: The snap controls bar

The topmost button on the bar is the master snap toggle; when it's off, all other buttons on the bar are disabled.

Things that snap and are snapped to are classified into three families that correspond to the three main sections of the snap controls bar (a fourth section at the bottom controls general snapping aids such as grids and guides). Each section of the bar has its own master toggle button at the top of it; somewhat confusingly (even if somewhat consistently), these sub-master toggles look exactly the same as the master snapping toggle at the very top. (Inkscape could really benefit from contributions by talented icon designers!) The three families are:

- Bounding boxes and their elements in the Selector tool (4.3).
- Paths, path nodes, and path node handles (12.5.1).
- Everything else: gradient handles (10.1), guide anchors (7.1.1), transform fixed points (6.4), text anchors and baselines (Chapter 15).

Sometimes, members of one group can snap to members of another one, but in general, they tend to snap within their own family. In particular, nodes and handles never snap to bounding boxes (although they can snap to paths).

The last section of the bar, at the bottom, controls major snap destinations that work for all three groups: the edges of the page (2.3), grid (7.2), and guides (7.1). These buttons affect only snapping; unlike the | and # keys, they will not hide the grid or guides.

In the icons on the snapping bar buttons, *blue* is used for the *snappables* (what snaps, which can only be points) while *green* depicts *snap targets* (what you snap *to*, which can be lines or points). If the same kind of entity can be both a snap target and a snappable, its button's icon uses green. (An exception to this is the grid and guide icons, which are snap targets but are traditionally blue.)

Let's look at the three families of snappables and snap targets.

7.3.1.1 Bounding Boxes Snapping

If you just need to align whole objects and aren't interested in nodes or other special points, use bounding box snapping, as shown in Figure 7-9.

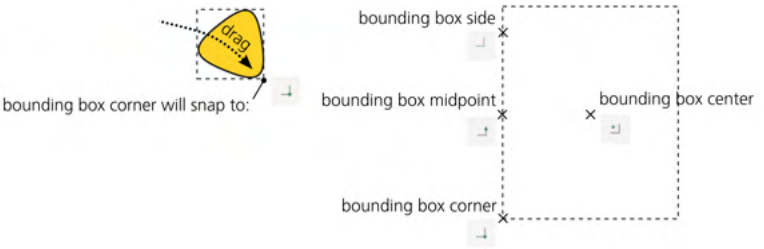


Figure 7-9: Bounding box snapping options

The first button in this group (Figure 7-8) is the local master switch. Disabling it disables the whole group, so you no longer need to worry about snapping of or to bounding boxes.

Further down, the buttons control, in order:

- Snap target: *edges* of the bounding boxes. When on, edges of the bounding boxes of all objects (selected or not) work as guides.

- Snap target and snappable: *corners* of the bounding boxes. When on, dragging anything in the Selector lets the corners of the selection bounding box snap. Figure 7-10 shows what happens when these two buttons are on and you drag one ellipse near another.

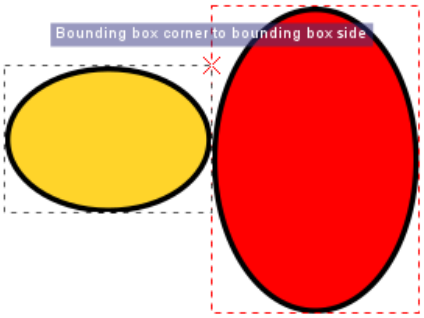


Figure 7-10: Snapping a bounding box corner to bounding box side

When snapping happens, Inkscape flashes a cross mark and a helpful tip at the snap point. If the bounding box you've snapped to is that of an unselected object, it will also be temporarily visualized as a red dashed box.

- Snap target and snappable: *midpoints* of the bounding box sides (four per box). You can think of it as snapping the central axes of objects; it makes most sense for objects that are symmetric around that axis.
- Snap target and snappable: *centers* of the bounding boxes. With this enabled, you can, for example, easily snap central-symmetric objects into symmetric compositions.

When you have both a line and a point on that line as snap targets (such as a bounding box side and its midpoint), the point target overtakes when you are close enough to it. Otherwise, you will snap to the nearest point on the line (and easily slide along it).

7.3.1.2 Paths and Nodes Snapping

The second group of buttons (Figure 7-8) controls snapping of path nodes (12.1) as you edit them in the Node tool, as shown in Figure 7-11. It also controls gradient/pattern handles (10.1, 10.8) that you can edit in the Gradient tool and other tools.

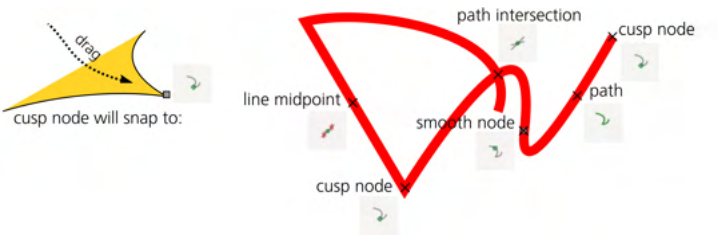


Figure 7-11: Paths and nodes snapping options

Again, the first button in this group serves as a local master switch for the group; when it's off, all other buttons in this group are grayed out.

Further down, the buttons control, in order:

- Snap target: *paths*. You can snap and slide a node along any path in your drawing (it need not be selected), but you won't snap to points on that path unless you activate one of the following buttons.
- Snap target: *path intersections*. Points where two paths intersect, or a single path intersects itself.
- Snap target and snappable: *cusp nodes* (12.5.5), such as corners on a path.
- Snap target and snappable: *smooth nodes* (12.5.5), such as quadrant points on an ellipse.
- Snap target: *midpoints* of straight line segments.

7.3.1.3 "Everything Else" Snapping

The third section of the snap controls bar (Figure 7-8) includes everything not covered by the other categories. Note that despite what the tool tips say, snapping of gradient nodes is in fact controlled in the previous node-snapping section (7.3.1.2); you cannot snap *to* a gradient node, as shown in Figure 7-12.

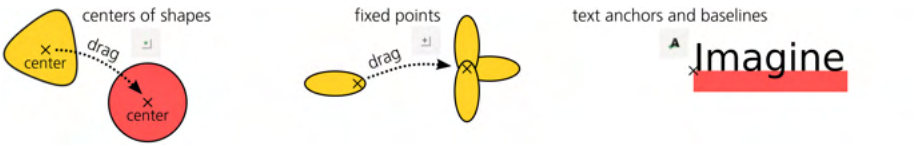


Figure 7-12: "Everything else" snapping options

Again, the first button in this section is a local master switch. Further down, the buttons control, in order:

- Snappable: *centers of objects*. Only symmetric shapes, such as stars/polygons (11.5) and ellipses (11.4), have a defined center; in the general case, it is not the same as the center of the bounding box.
- Snappable: *fixed points* of transformation (6.4).
- Snappable: *text objects' anchors* (15.4.4).

7.3.2 Snapping Preferences

On the Snap tab of the Document Properties dialog (Figure 7-13), you can adjust the snapping distances—how close you must get to the snap target for a snap to happen. You can set these distances (stored per document) separately for snapping to objects (including bounding boxes, paths, and nodes), grids, and guides.

For grids, the default setting is Always snap—that is, snapping happens at any distance (which, for a grid, cannot be more than the grid spacing anyway). If you zoomed out and some of the grid lines are hidden (7.2), snapping will happen only to visible grid lines.

For objects and guides, snapping is triggered, by default, at a distance of 20 screen pixels or closer. Using the screen pixel unit ensures that the snapping force is independent of zoom; if you want to move a snappable close to a snap destination but avoid snapping, you can do it at a closer zoom level.

You can also control whether snapping to paths will include clipping paths and masks (18.3), as shown in Figure 7-13.

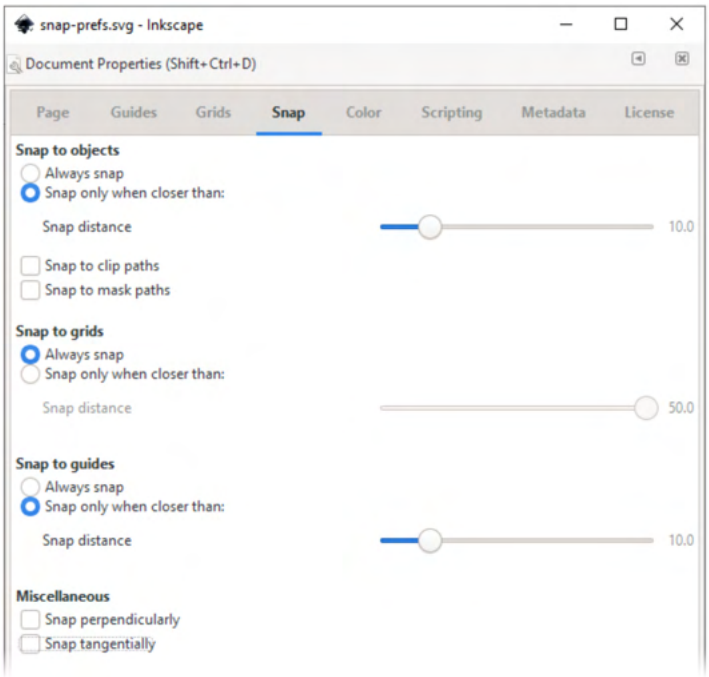


Figure 7-13: Snapping options in Document Properties

The global Preferences dialog (3.1.1) also has a Behavior ▸ Snapping page, shown in Figure 7-14.

- Snap indicator (on by default) is the cross mark and text label that flash on the canvas to tell you what just snapped to what (Figure 7-10). They pop up whenever snapping happens and disappear after the Snap indicator persistence delay that you can adjust (by default, it's 2 seconds).
- Only snap the node closest to the pointer tells Inkscape to ignore all other nodes you may be dragging except the one to which your mouse cursor is the closest. For example, if you plan to snap a corner of a rectangle, simply grab the rectangle near this corner for dragging—then the three other corners will not get in your way. This reduces "snapping noise" that may get very distracting in complex documents. If this option is on (by default, it's off), Inkscape will display a round mark over the closest snappable node while you drag.

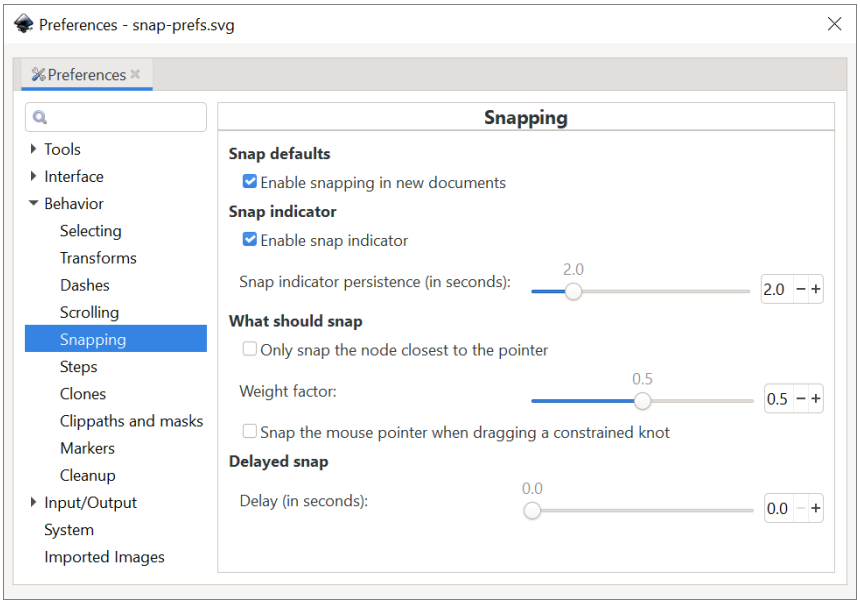


Figure 7-14: Snapping options in Preferences

- Weight factor is another parameter that controls Inkscape's choice of which of the dragged points to snap if more than one gets within snapping distance of a target. If set to 0, Inkscape snaps whichever point is the closest to its target, ignoring the position of the mouse cursor. Setting it to 1 is the same as the Only snap the node closest to the pointer option described above: now, among all nodes that *want* to snap, snapping will happen for the node that is closest to the cursor (which may or may not be the same as the node closest to its snap target). A value between 0 and 1 allows you to balance these two snapping strategies.
- Delay is the time, in seconds, that Inkscape waits after your mouse pointer stops before doing the snap. Increasing this value makes snapping more "reluctant"—which may be a good thing if your document is complex and you have many snapping modes enabled, so everything wants to snap to everything. In this case, increasing the delay and decreasing the snap distance will make your work easier.

7.4 Aligning

Aligning, like snapping, is a way to put objects in precise positions. Unlike snapping, it is not interactive: you just select your objects, choose a command, and the objects move into their positions all at once. Alignment commands are collected in the powerful Align and Distribute dialog (Shift-Ctrl-A or Object ▸ Align and Distribute from the menu).

Let's look at the align part of the dialog first (Figure 7-15).

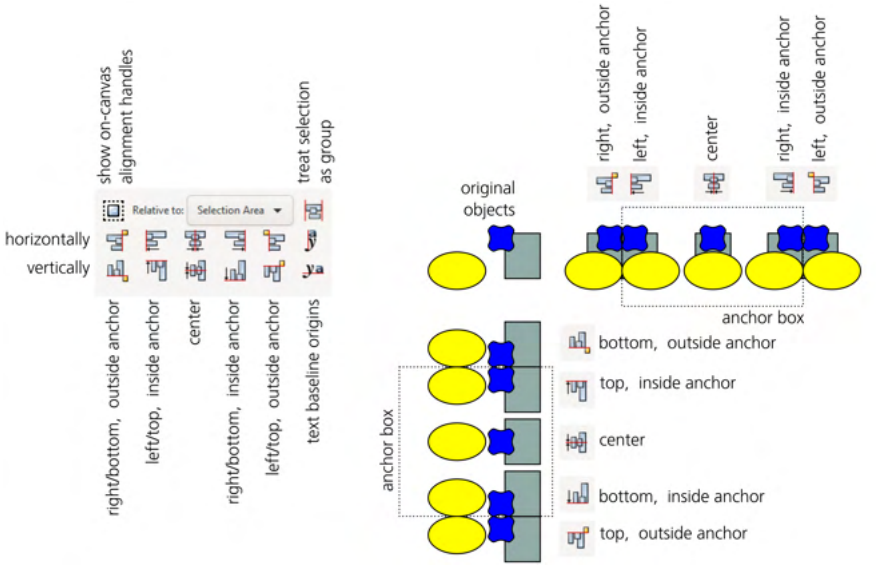


Figure 7-15: Aligning objects

To make sense of the Align dialog's many buttons, observe the following:

- Buttons in the top row align things by moving horizontally; those in the bottom row, by moving vertically.
- All alignment buttons move objects relative to the *anchor box*; use the Relative to list to determine what that anchor box is. The default that works in most cases is Selection Area—that is, the combined bounding box of all selected objects. You can also select the bounding box of one of the selected objects (the one that was the first or last to be selected, or the biggest or smallest one) as well as the entire page or entire drawing's bounding box.

  For example, if you have a collection of small objects and want to align them against the top of a large background object, select everything and choose **Biggest object** in the **Relative to** list.
- In the middle of this group, two buttons in the two rows align centers of selected objects at the center of the anchor box horizontally (top row) and vertically (bottom row). Around these two are the four buttons that press objects against the edges of the anchor box *on the inside*. Together, these six buttons are perhaps the most commonly used in the entire dialog. The four remaining buttons, two on the left and two on the right, align the objects at the edges of the anchor box but *on the outside*.
- The two buttons on the right end—those with letters on them—apply only to *text objects*. Every text object has a *baseline origin point* (15.4.4), and these buttons align the selected text object by these points. If, instead, you align text objects as regular objects by their bounding boxes, the characters with and without *descenders* (such as the bottom stroke of a *y*) and *ascenders* (such as the stem of a *d*) will not keep the line, as shown in Figure 7-16.

Figure 7-16: Aligning text objects

7.4.1 Aligning by Handles

By default, a second click on the selection in the Selector tool (or, alternatively, pressing Shift-S) switches the onscreen handles from the scale mode to rotate mode and back (6.3). If alignment is something you do often, you will appreciate that you can enable a *third* mode, the *align mode*, which is included in the click or Shift-S cycle if you toggle the Enable on-canvas alignment handles button in the Align and Distribute dialog (to the left of the Relative to drop-down).

Make sure the Enable on-canvas alignment handles button is on, select some objects, and then click the selection two times (don't double-click) or press Shift-S twice. The first click/key switches to the rotate mode, and the second one switches to the align mode with distinct handles. Now, to align all selected objects relative to the bounding box of the selection, just click or Shift-click one of the nine handles. Figure 7-17 shows how that works.

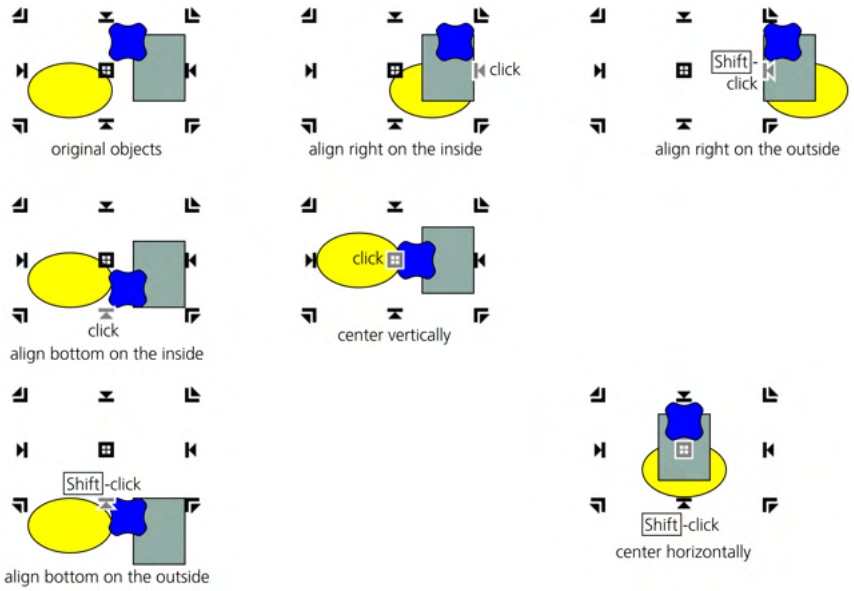


Figure 7-17: Aligning objects by on-canvas handles (click selection twice to get the align handles)

7.5 Distributing

Distributing selected objects moves them so that the intervals between them, if measured in a certain way, become equal. The same Align and Distribute dialog has eight object distribution buttons and two text distribution buttons, differing in how they measure these intervals. As with alignment buttons, half of these buttons (the top row) move horizontally and the other half (the bottom row) move vertically (see Figure 7-18).

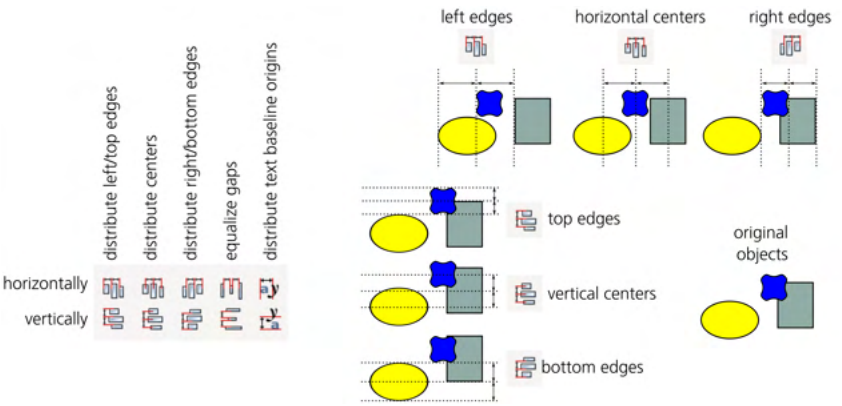


Figure 7-18: Distributing objects

Right under the centering buttons in the Align section, the Distribute section has two buttons for *distributing centers*. These buttons move selected objects in such a way that their centers (more precisely, the centers of their bounding boxes) are at equal distances from each other horizontally or vertically.

To the right and left of these two buttons, four more buttons perform the same equispaced distribution but for the right and left, bottom and top sides of the bounding boxes.

Two more buttons further to the right, instead of equalizing distances between same-named edges of bounding boxes, make the *gaps* between them equal. For example, in horizontal gap equalization, the distances between the right edge of one object to the left edge of the next one will become the same. When objects have different widths, this may make more visual sense than evenly distributing their centers. Figure 7-19 shows the difference between distributing centers and equalizing gaps.

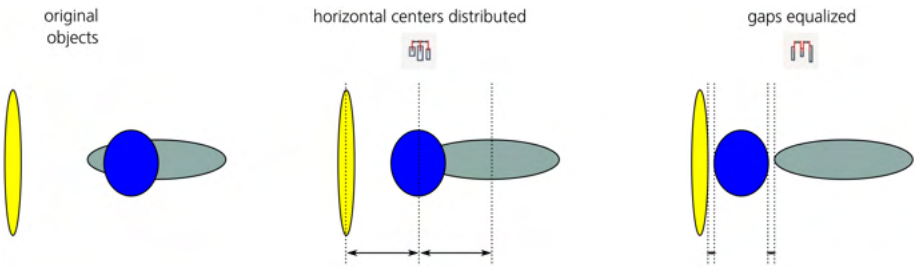


Figure 7-19: Distributing centers of objects vs. equalizing gaps between them

Finally, the two buttons furthest to the right distribute the baseline origin points of the selected text objects.

Unlike alignment buttons, distribution buttons always use the selection's bounding box as its anchor box (that is, the Relative to choice does not affect the Distribute section of the dialog). In other words, distributing always preserves the bounding box of the selection because it never moves the leftmost and rightmost (for horizontal) or topmost and bottommost (for vertical) objects, but only those that fall between them. (One consequence of this is that distributing only makes sense for three or more selected objects.)

So, if after distributing you find that your objects are too dense or too sparse, just select one of the objects at the edge of your distribution and move it inward (to make it denser) or outward (to make it sparser). Then, reselect all objects again and distribute them once more.

Note that selected objects may overlap, both before and after distributing. Even when equalizing gaps, objects' edges may overlap; such overlaps are treated as *negative gaps*, and they are still made equal to each other when you click the button. In other words, when there's not enough room, adjacent objects are made to overlap by the same amount.

When you are using the Node tool (12.5), the Align and Distribute dialog shows four buttons that allow you to align or distribute, horizontally or vertically, the selected path nodes (Figure 7-20).

Figure 7-20: Aligning and distributing path nodes (in the Node tool)

[1.1] 7.5.1 Exchanging Places

All the basic alignment and distribution commands described so far have one thing in common: they work only in one dimension—either horizontally or vertically. The same Align and Distribute dialog, however, has a number of more interesting commands that move objects in both dimensions at once.

One common operation is *exchanging places*. For two objects, this moves the center of object A to where the center of B was, and vice versa; for multiple objects, this moves A to the place of B, B to the place of C, and so on until Z jumps to the former place of A.

The dialog has three buttons for variations of this simple idea (Figure 7-21). They work exactly the same with two selected objects but differ in how they determine the A, B, . . . , Z order when more than two objects are selected.

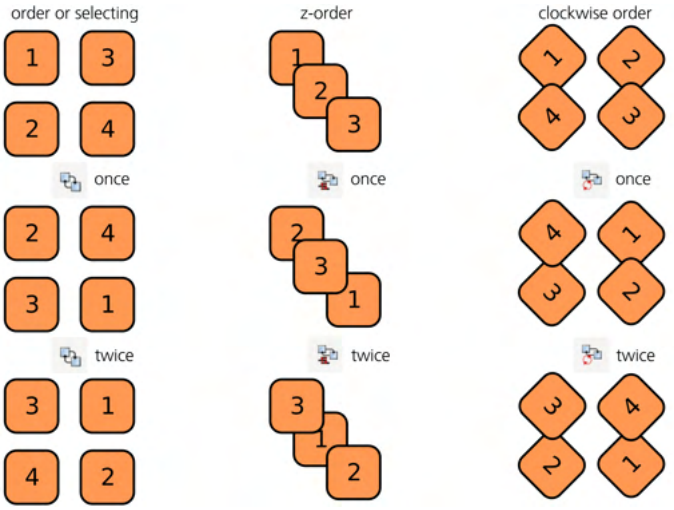


Figure 7-21: Exchanging places of multiple selected objects

Selection order
A is the last selected of the objects; Z is the first selected.

Z-order (4.4)
A is the topmost object; Z is the bottommost. Note that this command exchanges places without affecting the z-order; if you want to reverse the z-order of objects without affecting their placement, use the Restack extension (see Figure 4-6 on page 63).

Clockwise order
Inkscape enumerates the selected objects as if looking at them from the center of selection and going clockwise. Think of your objects like numbers on a clock face. Pressing this button will move 12 to 1, 1 to 2, and so on until 11 moves to the place of 12.

7.5.2 Randomizing and Unclumping

Randomizing simply moves each selected object to a random position within the selection's bounding box. Each click of the button results in a new random placement—you can repeat this until you find a randomization you like, as shown in Figure 7-22.

NOTE *When you click the* Randomize *button repeatedly, only the first click remembers the current bounding box of the selection; subsequent clicks will use this remembered bounding box as the area within which to scatter objects—even if the bounding box changes after randomization. This prevents the objects from drifting away as you randomize them repeatedly.*

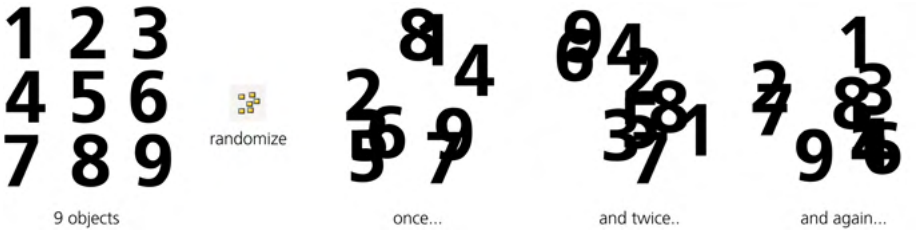


Figure 7-22: Randomizing object positions

Unclumping is similar to equalizing gaps but works in both dimensions at once, trying to equalize the closest distances within all pairs of adjacent objects in the selection. It works best for a large number of objects, as shown in Figure 7-23.

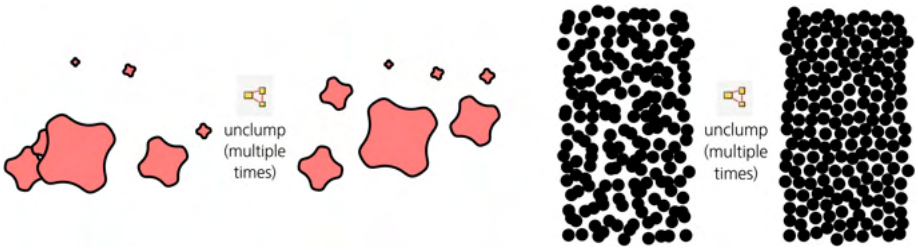


Figure 7-23: Unclumping objects

Unclumping is an iterative operation, so you may want to use it repeatedly until you are satisfied with the result. It does not replace your current arrangement entirely but improves it in small steps, nudging each object toward a point equidistant from its immediate neighbors (which themselves move). The most noticeable result of this is a reduction of *clumps*—places where objects are too dense.

In most cases, iterative unclumping converges—that is, it reduces visual unevenness in distribution of the objects, so that each subsequent unclumping has progressively less of an effect. At some point, further attempts to unclump an already-unclumped swarm of objects will just slowly disperse and migrate them in random directions without improving uniformity.

Unclumping objects produces a texture that looks random and man-made at the same time. It is similar to what a human would produce if asked to fill a space evenly with random dots. Examples of such textures include polka-dot patterns (10.8.4) and dot engravings where shading is created by smoothly (without clumps) varying the density of scattered dots.

NOTE *Unclumping does not take into account the actual shape of your objects—only their bounding boxes. It therefore works best with objects of a simple consistent shape, such as circles or stars. You will see more examples of unclumping in the section on the clone tiler (16.6).*

7.5.3 Removing Overlaps

Unlike unclumping, *removing overlaps* is a deterministic operation. It moves each selected object by a strictly minimum distance required for all objects to be free from overlaps—and it usually succeeds on the first attempt (I say "usually" because, in general, this is not a trivial operation and may sometimes fail). The amount of moving it performs, of course, depends on the initial arrangement—but in most cases, the layout of objects after this operation will still be similar to the original layout, even though it may have done some drastic reshuffling to free up the objects.

For this command, you can additionally set the horizontal and vertical spacing (in px units) added to the objects' bounding boxes, as shown in Figure 7-24.

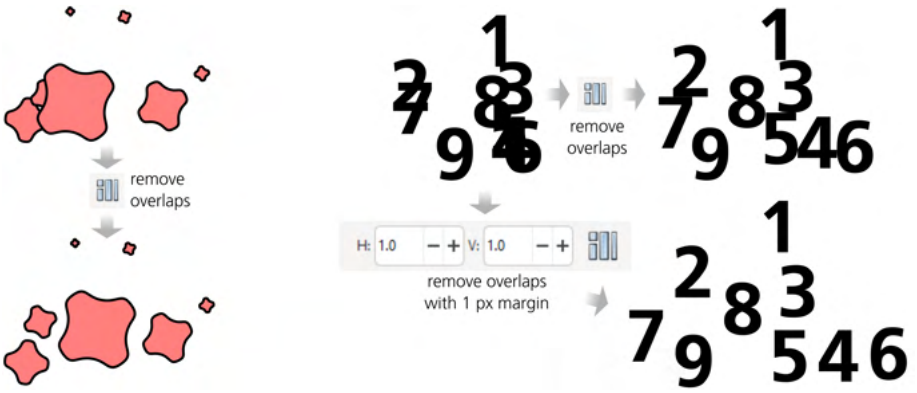


Figure 7-24: Removing overlaps between objects

Notice that the objects that did not overlap with any other selected object did not move.

Yet another button in this dialog, Connector network layout, is supposed to nicely rearrange a diagram made with the Connector tool (14.5). In theory, this should move the connected objects around so as to minimize the length and/or intersections of connector lines; in practice, it looks like this command works no better than plain Remove overlaps.

7.5.4 Arranging Objects

Even with all the possibilities in the Align and Distribute dialog, you may have some unfilled object arrangement needs. The Arrange dialog (accessed from the Object menu) fills two of them: it will arrange your objects into either a two-dimensional table or a circle, both with plenty of spacing and alignment options (as you would expect from Inkscape).

This dialog is most useful for *different* objects that you already have, such as imported bitmaps that you want to arrange into a gallery. If you just need a pattern, table, or grid of *identical* objects (or objects that differ only in size or style), look into the clone tiler (16.6).

7.5.4.1 Arranging in a Grid

The dialog's Rectangular grid tab (Figure 7-25) takes the number of selected objects and calculates the Rows and Columns that would make the most compact table (with the number of rows as close to the number of columns as possible). If you change any of these numbers, the other one recalculates. For example, if there are 29 objects selected, the dialog will suggest 5 rows and 6 columns; if you scroll the number of rows up, the number of rows and columns will drop—so you will get 6 by 5, 8 by 4, 10 by 3, and so on.

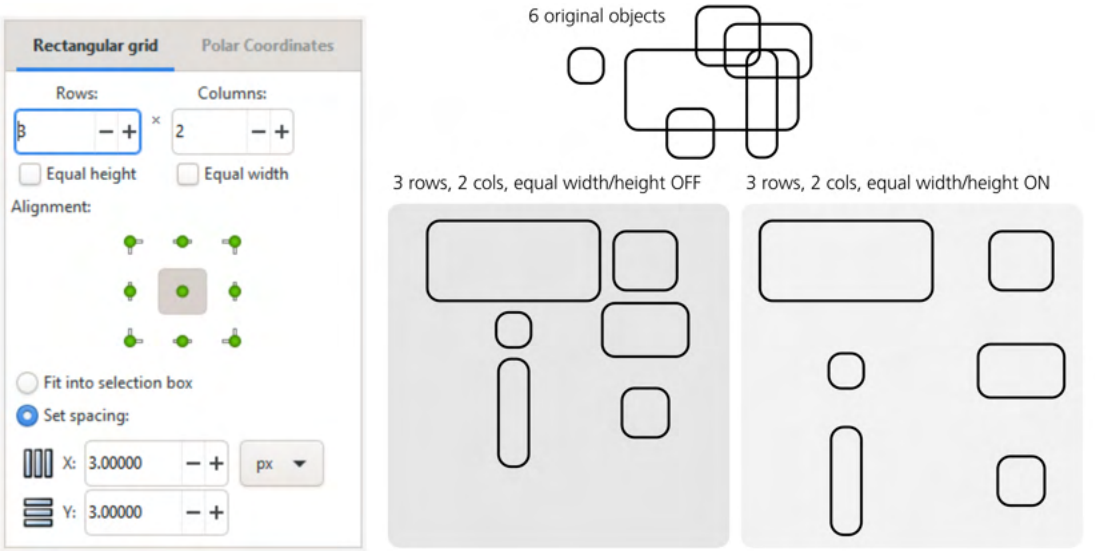


Figure 7-25: Using the Arrange dialog to create a grid of objects

Next, two checkboxes control whether all rows and all columns will have equal height and width, respectively. With the Equal height on, all rows in the entire grid will be as high as the tallest of *all objects* (plus spacing, see below). If it's off, each row will be only as high as the tallest of all objects *in this row* (plus spacing). Note that even with Equal width off and with zero spacing, your objects will not necessarily touch in the grid if the widest-in-column objects happen to be not in adjacent cells.

Further down, a group of buttons select which point of each object is being aligned into a grid. This is how you control the alignment of objects within their rows and columns; by default, they are centered in both dimensions.

The rest of the controls define how large your grid will be. The Fit into selection box option ensures that the arrangement will be exactly as wide and as tall as was your selection before you clicked Arrange. In other words, after creating the grid, Inkscape will set the spacing between rows and columns so that the overall dimensions of the selection don't change (even if this results in overlaps). For example, if you want to make a grid sparser, undo arranging, drag one of the objects away from others, reselect all objects, enable **Fit into selection box**, and **Arrange** again. Alternatively, you can Set spacing for rows and columns explicitly by typing the desired values and choosing the unit.

The biggest problem with the grid arrangement is that the row/column position of each object is somewhat unpredictable. Even if your objects are already in a rough table and you just want to line them up, using this dialog may make some objects reshuffle to obviously wrong places. If you're trying to fix this by swapping places of a couple objects, try the object exchanging commands in the Align and Distribute dialog.

7.5.4.2 Arranging in a Circle

The other tab of the Arrange dialog, Polar Coordinates, lets you put a number of objects into a circular or elliptic ring, with equal angular distances between them. For example, with six selected objects, you get a hexagon, as shown in Figure 7-26.

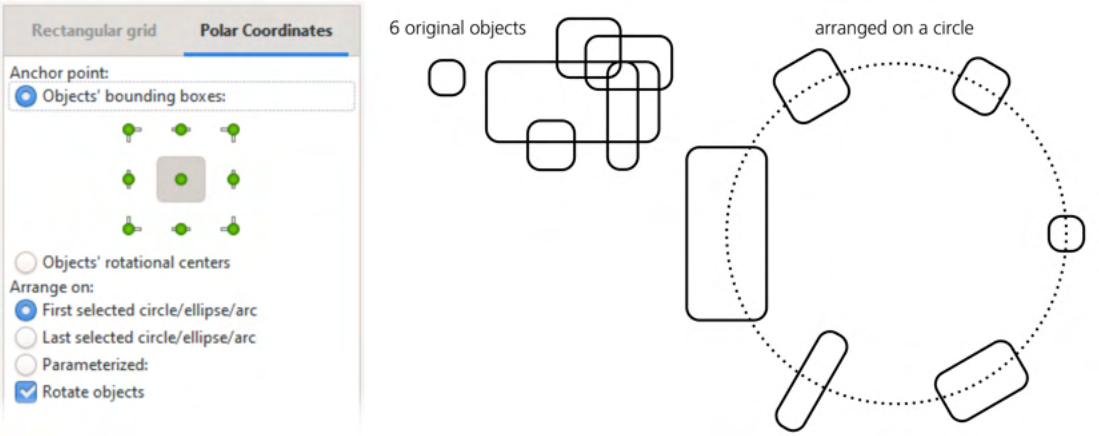


Figure 7-26: Using the Arrange dialog to create a grid of objects

You can select which point of each object is aligned to the circle; by default, it's the center, but you can switch to any of the corners or midpoints of the bounding box, as well as use each object's rotational center (that is, the fixed point, 6.4). As for the circle to which you're arranging, it can be an actual circle, ellipse, or arc (11.4) included in the selection, or you can switch to Parametrized and specify the arrangement guide via coordinates. The final option selects whether the objects will be rotated when placed into their positions (the default) or moved without rotation.

8

STYLE: COLOR AND OPACITY

Now that you know all about selecting and transforming objects, let's start a new topic that will entertain us for the next several chapters: styling. The way an object in an Inkscape document looks is determined by that object's *style*, which consists of separate *properties*. To use Inkscape, you need to be familiar with at least some of the style properties and the tools for editing them.

In this chapter, I describe the most basic style properties, including the common types of *paint* (used for fill and stroke) and *opacity*. After this, Chapter 9 covers stroke style, Chapter 10 deals with gradients, meshes, and patterns, and Chapter 15 describes text style properties. Filter effects, such as blur, are also part of an object's style; I discuss those in Chapter 17.

One style-related command worth knowing before all others is Edit ▸ Paste Style (Shift-Ctrl-V). It takes the complete style of the object you last copied to the clipboard and applies it to all selected objects. You'll often want some objects to be mostly, but not completely, identical in style. It's much easier to paste the same style on all of them first and then change only the properties that need to be different. Also, you've already seen how to search objects by style properties (5.13) and how to select objects with the same style as the selected object (5.12).

8.1 Style Properties and Selectors

[1.1] Inkscape has many dialogs, commands, and tools that deal with objects' style; you're probably already using some of them. However, before looking at any specific style properties or tools, let's talk a bit about how style is encoded in SVG. If you're not interested in the technical details, feel free to skip this section.

The way object style is recorded in the SVG code of a document is not defined by the SVG standard; for this purpose, SVG reuses another standard called *Cascading Style Sheets (CSS)*. You are probably familiar with CSS if you ever did any kind of HTML coding because HTML also uses CSS. (SVG uses only a subset of CSS; refer to the SVG standard at *http://w3.org/TR/SVG* as well as to Appendix A for details.)

In CSS, a style is a collection of named *properties*. Each property has its own rules for what values it can take. If you've ever browsed objects in Inkscape's XML editor (4.10), you probably noticed that the `style` attribute is present on most objects, and it looks like a list of `name:value` pairs separated by semicolons. Each of these pairs defines a property. Example properties are `fill`, which specifies the fill paint (8.2), `stroke-width` for stroked paths, or `font-family` for text objects.

> **CSS PROPERTIES AS XML ATTRIBUTES**
>
> Technically, a property also can be saved in an attribute of its own, instead of being lumped together with all other properties in the `style` attribute. Inkscape can read such SVG as well, but when it changes the file it moves the properties to the `style` attribute.

If you want to edit style at the level of CSS properties, the XML Editor is not very convenient. A specialized tool for editing style is the Selectors and CSS dialog (Ctrl-Shift-Q), shown in Figure 8-1. Here, you can not only edit the properties of objects, but also create *selectors*—styles that apply to more than one object so that you can update all of them at once.

The dialog's first half (you can arrange the two areas of the dialog vertically or side-by-side) deals with the selected object. Even if you don't use any of the advanced features, here you can conveniently edit all of the properties of the selected object's style. Inkscape will prompt you with available property names and will validate the value you type. The + button at the top of the list adds a new property; the x in the red circle next to a property deletes it.

The magic starts when you click the + button in the bottom left corner of the dialog. This creates a new *selector*, and you are prompted to give it a name. You can think of a selector as similar to a CSS class: a collection of elements and a style that applies to all of them. Initially, only the selected object is in the selector, and that selector has no style properties of its own. To add more objects to the selector, select them and click the + button next to the selector you want to apply.

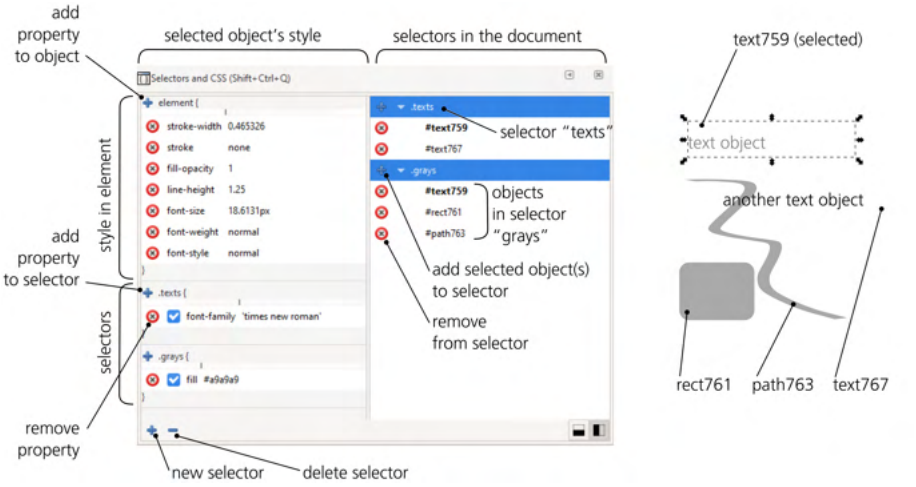


Figure 8-1: The Selectors and CSS dialog showing the style of the selected object and the selectors defined in the document

One object can belong to more than one selector; for example, in Figure 8-1, `text759` (each object is identified by its `id` attribute, 4.1) is included in the `texts` and `grays` selectors. By the way, when you click a selector's name in the dialog, its objects become selected on canvas—this makes this dialog useful for "storing" complex selections even if you don't use selector styling.

To add style properties to a selector, select any object included in it and click yet another + button next to the selector's name in the top (or left) part of the dialog that deals with the object's style. Any value you provide here will apply to all objects in this selector, overriding any value they may have for the same property in their individual styles. In Figure 8-1, the `texts` selector has the property `font-family: 'times new roman'` and the `grays` selector contains `fill: #a9a9a9` (gray color). The selected object `text759`, being included in both these selectors, has both gray color and the Times New Roman font.

SELECTORS IN SVG

In the document's SVG code, the selectors and their style properties are stored in the `style` element under the root (again, just like in HTML).

8.2 Paint

As you probably already know, objects can have *fill* and *stroke*, which have some kind of *paint* applied to them. One could say that an object is filled by one paint and stroked by another. A paint may be of several different types.

None (no paint)

An object with its fill set to none has only stroke, and an object with its stroke set to none has only fill. If both fill and stroke are none, the object is completely invisible (and cannot be selected by clicking, 5.4), just as it would be with zero opacity.

Flat color

This is the most common type of paint: a plain solid color. Later in this chapter, you'll see many ways to view and change the fill and stroke colors.

Gradient, gradient mesh, or **pattern**

Both fill and stroke may have various types of smooth color gradients, gradient meshes, or repeating patterns. These complex paint types are the subject of a separate chapter (Chapter 10).

Swatch

A swatch is a flat color that you can assign (as fill or stroke paint) to more than one object, so that when you edit it, all objects with that swatch update. A swatch belongs to the document in which it was created and is saved with it. Swatches are created and edited in the Fill and Stroke dialog, but they are also listed, and can be applied from, the Auto palette (8.5.1). This feature is still somewhat clumsy to use; also, in SVG a swatch is encoded as a fake one-stop gradient, which is not very natural. If you need reusable style, I recommend using CSS selectors instead (8.1).

Unset

This means that a fill or stroke is *not specified*, and therefore may be *inherited* from an object's ancestors (those above it in the document's XML tree); this is not the same as none (which simply forces invisibility). If an object's ancestors have no stroke or fill of their own (as is usually the case for groups or layers), an object with unset fill or stroke will get the SVG defaults: an invisible stroke and a solid black fill. Unsetting is useful with clones (Chapter 16) because it allows you to create clones painted differently from their originals.

NOTE *A path need not be closed in order to be filled; unclosed paths (12.1.1) can still be filled as if their ends were connected by an unstroked straight line segment.*

The main tool for editing an object's style, the Fill and Stroke dialog (Shift-Ctrl-F), has a number of buttons on its Fill and Stroke paint tabs that correspond to the different paint types. The pressed button indicates the current paint type of the selected object; if multiple objects with different paint types are selected, no button is pressed and the dialog says *Multiple styles.*

Another important control is the *selected style indicator* (2.8) in the status bar. Always present onscreen, it lets you quickly look up the paint type of the selected object, as well as change it via the right-click menu: right-click the fill swatch at the *top* to change the *fill* paint and the stroke swatch at the *bottom* to change *stroke* paint (Figure 8-2).

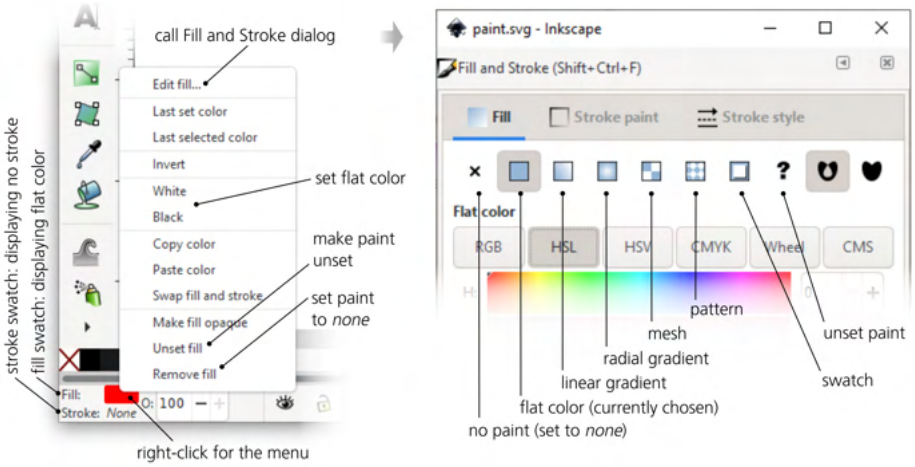


Figure 8-2: Paint types in the selected stroke indicator (left) and the Fill and Stroke dialog (right)

One important property that applies only to fill but not stroke is the *fill rule*. It may take one of two values. The value of `evenodd` means that *any* self-intersections or inside subpaths (12.1.1) create holes in the fill of a path. The value of `nonzero` means that most holes are covered with fill; more precisely, if the inner subpath has the same direction as the outer one, the hole will be painted, and otherwise it will be a hole without paint.

NOTE

Some of the inner subpaths can still produce holes even with `nonzero` fill rule. This depends on the direction of the subpath (12.1.1): if it is counterdirected relative to the outer path, the hole will remain unfilled.

To change the fill rule property, use the two toggle buttons in the **Fill and Stroke** dialog, as shown in Figure 8-3.

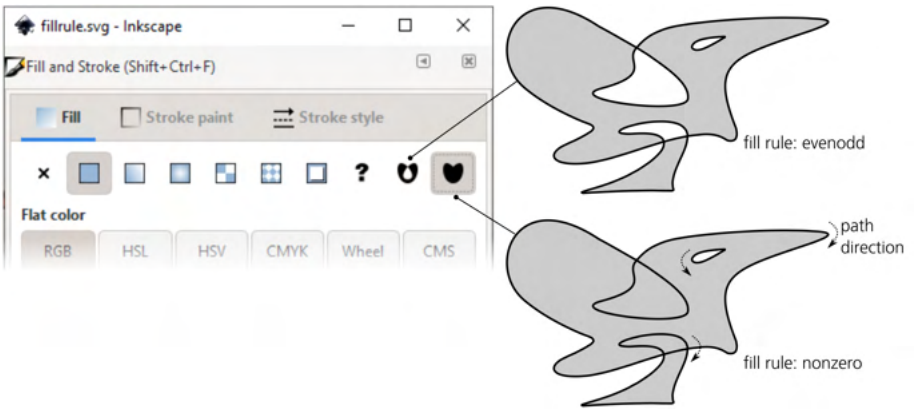


Figure 8-3: Changing the fill rule in the Fill and Stroke dialog

Not all object types can or should have fill and stroke:

Bitmaps
: Bitmap objects (Chapter 18) cannot have either fill or stroke. However, you can convert a bitmap into a rectangular path filled by that bitmap as a pattern (just press Alt-I), which can then have a stroke applied to it.

Groups
: You would rarely want to assign fill or stroke to a group by itself, because the objects inside a group usually have their own paint and ignore what is set on their parent group. In fact, if you try to set a fill or stroke on a group in Inkscape, it will set it on the group but will *also* assign it recursively to all the group's members. Only explicitly *unset* fills or strokes of the group's members will be left alone; however, they will inherit the new fill or stroke from their ancestor anyway.

Clones
: If you try to set fill or stroke on a clone (Chapter 16), it won't have a visible effect unless the original of that clone has its fill or stroke *unset*.

8.3 Opacity

On top of the fill or stroke paint, an object may have *opacity* that you can edit in the O: field in the status bar or with the Opacity slider in the Fill and Stroke dialog. Just like filters (Chapter 17), opacity applies to the entire object without any distinction between fill and stroke.

In fact, in SVG an object can have three kinds of opacity: *master opacity* (which is called simply *opacity* in most places in the UI as well as in this book), *fill opacity*, and *stroke opacity* (Figure 8-4). The two latter kinds apply only to fill or stroke, respectively, and while Inkscape allows you to view and change them, it generally discourages them in favor of master opacity. Situations where you might need your stroke opaque but fill transparent or vice versa are not too frequent in practice; in most cases, master opacity is more natural and easier to use.

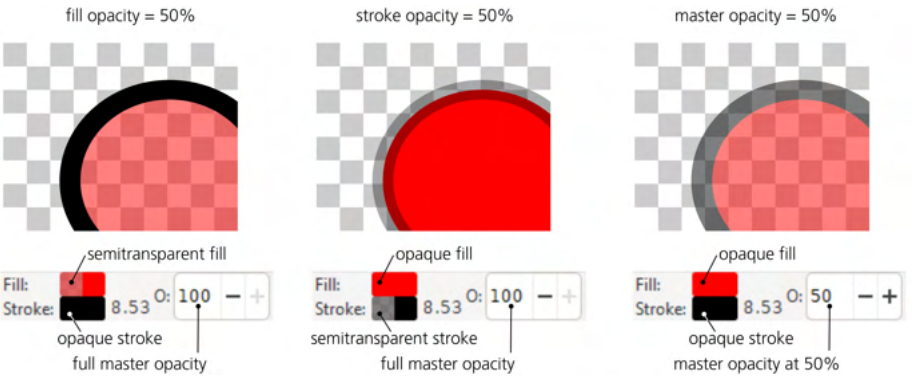


Figure 8-4: Fill opacity, stroke opacity, and master opacity

One consequence of lowering stroke opacity is that you can see the fringe of the object's fill, normally obscured by its own stroke. Per SVG rules, stroke is normally drawn on top of fill (in the object's own internal z-order; however, see 9.6), and fill is bounded by the midline of the stroke, so with a semitransparent stroke, you will see three distinct boundaries: the outer boundary of the stroke, the stroke's midline where its fill starts, and the inner boundary of the stroke where it overlaps the fill.

Master opacity, unlike stroke opacity, has no such problems—it applies to the object as a whole, without revealing any untidy fringe. In the same way, when applied to a group, it makes the group transparent as a whole, which may be different from the same level of opacity assigned to individual objects in the group (Figure 4-10 on page 69).

Also, unlike fill or stroke opacity, master opacity is applicable to objects that have no fill or stroke of their own, such as bitmaps and most clones.

NOTE *Opacity and transparency look at the same thing from opposite angles: when an object's opacity is zero, it is fully transparent; conversely, when its transparency is zero, it is fully opaque. Another synonym for opacity that you may encounter is alpha.*

8.4 Color Models

The most important building block of style is *color*. A discussion of how color is represented on computers could easily fill a book; Inkscape may not be the most powerful color program out there, but it is nevertheless quite rich and complex in this area.

When working with color, the first step is to choose the *color model*. Inkscape supports several popular color models that represent the same color differently.

8.4.1 RGB

In the most common color model, *RGB*, every color is represented as a mix of red, green, and blue components or *channels*. This model is implemented by nearly all computer displays (which are typically composed of tiny light sources of these three colors) and is the primary color model used in SVG as well as most other computer graphic formats.

Depending on the software you use, the value of each of the three channels in an RGB color may be either a fractional number between 0 and 1 (such as 0.5) or an integer between 0 and 255 (such as 127). These two systems are equivalent; for example, a color with R = 0, G = 0.5, B = 1.0 is the same as R = 0, G = 127, B = 255. Inkscape normally uses the 0–255 integer format, which I also prefer in this book; however, in some parts of the Inkscape UI you will see a choice between the integer and the fractional formats.

In RGB, higher values of the channels make the color lighter, and lower values make it darker. Thus, RGB 0/0/0 is pure black and RGB 255/255/255 is pure white. Any RGB color where the values of all channels are equal is a shade of gray; making the channels unequal adds saturation to the color, and the bigger the inequality, the greater the saturation. So colors such as pure red (RGB 255/0/0) or pure yellow (RGB 255/255/0) have maximum possible saturation.

The Fill and Stroke dialog lets you edit colors in RGB using either numeric input fields or sliders that move within graduated-color grooves (see Figure 2 in the color insert). Note that the colors of the grooves change as you move the sliders; each groove shows you the colors that you will get by moving the slider within it if the other sliders remain where they are. The fourth slider at the bottom, labeled A (alpha), represents the fill or stroke opacity (and thus is not, strictly speaking, part of the color).

An RGB color is often expressed as a string in the form `RRGGBB` where each of the three components is represented by a two-digit hexadecimal (base 16, instead of the conventional base 10) number. Hexadecimal numbers can use digits `0` through `9` and letters `A` through `F`. The maximum integer value of a channel—`255` in decimal—is `FF` in hexadecimal. For example, `000000` is black, `FFFFFF` is white, `660000` is dark red. This form of representing colors is used in the SVG source of Inkscape documents, as well as in a lot of other software and languages (such as HTML). For example, when you choose Copy color from the right-click menu of the fill or stroke swatch in the selected style indicator (Figure 8-2), the `RRGGBB` representation of the color is copied to the clipboard. Also, you can view and edit an `RRGGBBAA` representation (with two more digits appended for the fill/stroke opacity) in the RGBA field in the Fill and Stroke dialog.

If multiple objects are selected that have different fill or stroke colors, both the Fill and Stroke dialog and the selected style indicator display the *averaged color*. If you change that averaged color, it will be assigned back to all the selected objects, in effect flattening any color differences those objects might have had, as shown in Figure 8-5.

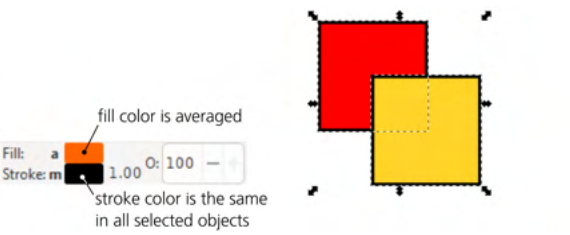


Figure 8-5: Averaged color in the selected style indicator

NOTE *This averaging is done in RGB—that is, the averaged color's R, G, and B channels are the arithmetic mean of the corresponding channels in the input colors. The same averaging formula, per SVG rules, is used in gradients for interpolating between stop colors (10.2).*

8.4.2 CMYK and CMS

Similar to RGB, in the *CMYK* model a color is obtained by mixing color channels. However, CMYK has not three but four channels: *cyan*, *magenta*, *yellow*, and *black* (see Figure 3 in the color insert). This color model is best suited for printed output because many professional printers and printing presses use CMYK. However, even when you specify a color in CMYK, what gets written into SVG is its RGB approximation, and conversion from CMYK to RGB and back typically introduces distortions because not all CMYK colors can be exactly represented in RGB and vice versa.

NOTE *Traditionally, numeric values of CMYK channels are in the range of 0 to 100, not from 0 to 255 as in RGB.*

The CMS (Color Management System) tab of the Fill and Stroke dialog allows you to edit colors in a *calibrated space*, which may include true calibrated CMYK (not the approximation of the CMYK tab) or Adobe RGB. This requires you to set up the color management in Input/Output ▸ Color management in Preferences (18.8).

8.4.3 HSL and HSV

Neither RGB nor CMYK are very intuitive; they represent the way computers and printers deal with color, not the way artists use it. Inkscape offers a more natural color model: *HSL (Hue, Saturation, Lightness)*. Unlike RGB and CMYK, the channels of HSL are not individual colors mixed together; rather, they are the *properties* of the color that, together, define it unambiguously.

The *hue* channel is the rainbow of maximum-saturation colors, starting from red and going through yellow, green, blue, purple, and back to red. The *saturation* channel positions a color somewhere from maximum colorfulness, through drab and dull, to pure gray—at the same brightness level. Finally, the *lightness* channel goes from black to the given color and then to white; this means any color with maximum lightness is white and any color with zero lightness is black, regardless of hue and saturation.

The HSL color model, once you get used to it, is very intuitive. When you feel that some color isn't quite right, you may think it needs to be made lighter or less saturated—not that it needs more red or green. Many color-related tools in Inkscape prefer HSL for choosing and changing colors.

In the Fill and Stroke dialog, there are two tabs that allow you to edit a color in the HSL model (see Figure 4 in the color insert). The first one (HSL) uses traditional linear sliders. The other one (Wheel) has a circular ring for the H channel (which therefore has a better resolution: if straightened, it would be longer than a linear slider fitting into the window) and a triangle that encodes the saturation and lightness channels. Rotating the mark on the hue ring rotates the triangle too, so that its maximum-saturation tip always points to the current hue on the ring.

Inkscape also supports a similar color model called *HSV (Hue, Saturation, Value)*. Other programs may call it HSB (Hue, Saturation, Brightness). The only difference from HSL is that with HSV, the Value component changes only from black to a given color (such as red). To get pure white in HSV, you need to set Value to maximum *and* Saturation to zero. In HSL, the Lightness component ranges all the way from black through a given color to white, so it is symmetric with regard to light and dark.

8.5 The Palette

The simplest way to assign a color to the fill or stroke of selected objects is by clicking one of the swatches in the *palette*, usually located at the bottom of the editing window above the status bar (see Figure 5 in the color insert). Simply

clicking assigns fill color to the selected objects; Shift-clicking assigns stroke color. The leftmost button on the palette *removes* fill or (with Shift) stroke.

Apart from clicking, you can drag and drop colors from the palette onto objects. This is one of the few ways to change objects without selecting them: when you drop a color on an object, only that object will change, even if it is not selected and even if some other object is selected. Dropping also works for changing stroke color: just drop the color swatch precisely onto the outline of a path or shape.

Inkscape comes with many stock palettes. On the right end of the palette, there's a button with a triangle mark; click it to open a menu listing available palettes—such as a web-safe palette, palettes used by Ubuntu and Windows for their UI, and various monochromatic palettes (greens, golds, and so on).

In the same menu, you can choose one of the standard *sizes* of swatches, their *width* (narrow swatches are harder to click but may get rid of the palette's scroll bar by fitting the entire palette to your screen width), as well as enable the palette to *wrap around* (which makes it taller but also gets rid of the scroll bar). Also, if you prefer, you can turn on the border around the palette's swatches.

In addition to the always-docked horizontal palette, Inkscape has the Swatches dialog (Shift-Ctrl-W), shown in Figure 8-6.

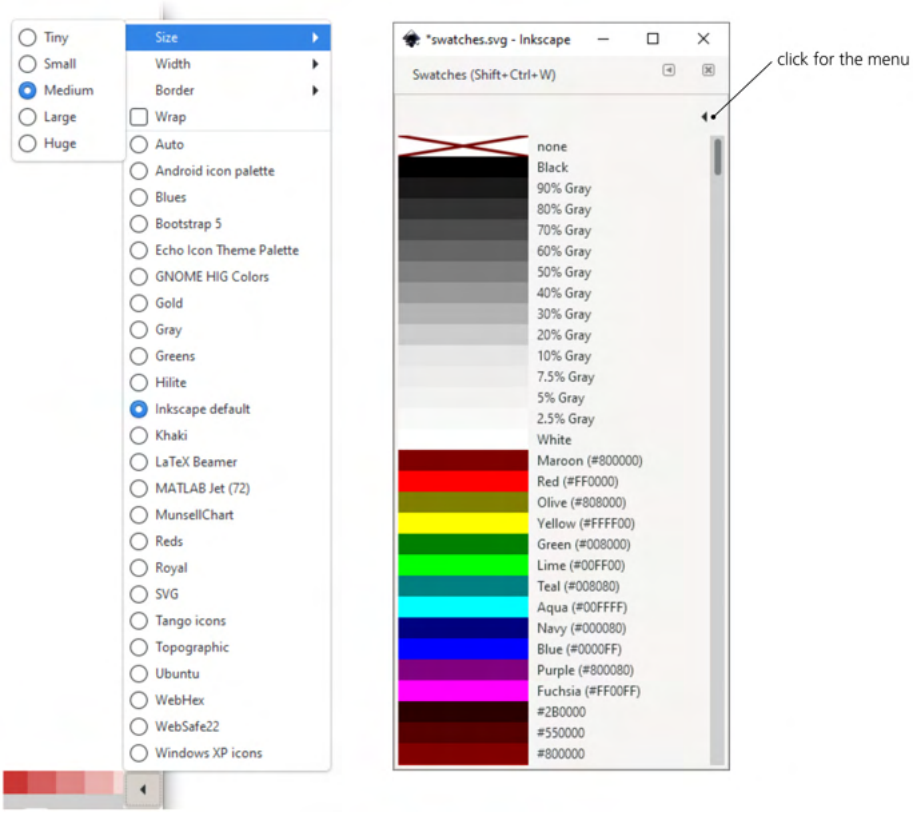


Figure 8-6: The palette menu (left) and the Swatches dialog (right) in the List format showing colors with their names

It has all the same palettes and size options, except that it has an additional List option where it lists all the colors along with their names. Being a dialog, it can float wherever you need it or be placed in the dock on the right and squeezed into a narrow vertical strip.

8.5.1 Editing Palettes

Each Inkscape palette is a text file in a simple format that you can edit. If you have created a new palette and want Inkscape to use it, go to **Preferences**, **System** page, and click **Open** next to **User palettes**. This will open a folder where you can place your palette file; on the next launch, Inkscape will add it to its list of palettes.

Inkscape's palette format is that of the GIMP raster editor, so you can drop any palettes from a GIMP distribution into that folder to use them in Inkscape. Here's the beginning of the default palette file, *inkscape.gpl*; the color components (R, G, B) are in the 0–255 format:

```
GIMP Palette
Name: Inkscape default
Columns: 3
# generated by PaletteGen.py
  0   0   0  Black
 26  26  26  90% Gray
 51  51  51  80% Gray
 77  77  77  70% Gray
...
```

The Auto option in the list of predefined palettes lets you create a custom palette saved with the current document. To add, or edit, a color in the Auto palette, you need to create a swatch in the Fill and Stroke dialog (8.2).

8.6 The Selected Style Indicator: Paint Commands

Apart from the various paint styles, the right-click menu of the selected style indicator (Figure 8-2 on page 143) has several useful commands for dealing with paint. Note that the fill swatch (top) and stroke swatch (bottom) are distinct even though their context menus are similar; be sure to right-click the half that you want to change.

Edit fill/stroke . . .
: Choosing this, as well as simply clicking the fill or stroke swatch, opens the Fill and Stroke dialog.

Last set color
: This assigns to the selection the color that was last *set* on the corresponding paint (that is, on fill or stroke) of any selected object. For example, if you just painted something a particular shade of blue and now want *all* your objects to have that fill, select all and call this command from the fill swatch's right-click menu.

Last selected color
: This assigns to the selected object the corresponding paint (that is, fill or stroke) of the object that was *selected* before the most recent selection change. For example, select an object whose fill color you like, then select some other object, and use this command on its fill swatch to assign the same color to it.

Invert
: This replaces a flat-color paint by the (RGB) inversion of its color—for example, white becomes black and yellow becomes blue.

White and Black
: These commands simply assign those colors to the paint.

Copy color and Paste color
: These commands allow you to exchange objects' colors via the system clipboard. For example, you can copy one object's stroke color and assign it to another object's fill, or paste a color (as an `#RRGGBB` string) into any other program where you might need it.

Swap fill and stroke
: This command exchanges the fill and stroke paints on selected objects. For example, if a selected object had no stroke and a blue fill, after applying this command, it will have a blue stroke and no fill. This command can be useful when you use both the Pencil (14.1.2) and the Calligraphic pen (14.2) tools for drawing and want the result to use the same color, even though the Pencil creates stroked paths without fill and the Calligraphic pen creates filled paths without stroke.

Unset fill/stroke
: Make fill or stroke *unset* (8.2). This is not the same as removing it.

Remove fill/stroke
: Set fill or stroke to `none` (8.2). This command has a shortcut: middle-click the fill or stroke swatch to set the corresponding paint to `none`; if the paint is already removed, middle-clicking creates the default fill or stroke (usually black). Thus, to get rid of stroke on selected objects, Shift-click the "no paint" swatch on the palette or middle-click the bottom of the selected style indicator.

Make fill/stroke opaque
: Remove any fill opacity or stroke opacity from paint (8.3). Master opacity (shown by the O control to the right of the swatches) remains unchanged.

8.7 The Selected Style Indicator: Color Gestures

Apart from its other numerous functions, the selected style indicator has a convenient and somewhat unique method for quick adjustment of colors: *color gestures*. Simply grab the fill or stroke color swatch and drag it out onto the canvas, with or without modifier keys, as described below.

NOTE *Color gestures work only when the swatch displays a flat color; they do not work when a swatch displays* None *(no paint),* N/A *(nothing is selected), or a gradient (although you can select one or more gradient stops in the Gradient tool and color-adjust them by color gestures just as you would with objects).*

Color gestures work in the HSL color space (8.4.3). Dragging without any keyboard modifiers adjusts the hue channel, dragging with Shift adjusts saturation, and dragging with Ctrl adjusts lightness.

Think of it as "rotating" the color swatch: by dragging from it, you pull out an invisible handle and use it to rotate it away from the original direction, which is assumed to be 45 degrees to the northeast. When you click and drag the color swatch, imagine a line—a *no-change axis*—going from the point where you clicked, diagonally, into the drawing area of the window. By dragging *below* or *to the right* of that axis, you *decrease* the corresponding color channel, down to the minimum at the lower edge of the window; by dragging it *above* or *to the left*, you *increase* it, up to the maximum at the left edge of the window. If you hover your mouse exactly over the no-change axis, there is no change (see Figure 6 in the color insert). The status bar reports, as you drag, the channel you are adjusting, the original value of that channel, the new value, and the difference.

The angular nature of the color gestures means that it's easy to adjust the precision. When you drag close enough to the swatch, any small movement produces a big change of color. If you need a finer adjustment, just drag farther away from the swatch, toward the center of the Inkscape window or even to its upper-right corner, where the same movements will make much smaller changes in color.

You can switch channels while you drag—that is, you don't need to drag from the swatch again and again if you want to adjust all three channels. You can do it all in one drag by pressing and releasing Ctrl and Shift as necessary. The moment you change the keyboard modifiers while dragging, the zero-change axis for the new channel is temporarily moved up to go through the current mouse position; this way there are no sudden changes in color if you happen to switch modifiers away from the original 45-degree line.

The Alt modifier is special: pressing Alt while dragging means "do nothing." This allows you to move the mouse, without releasing, to a more convenient place from which you can continue tweaking the color after letting go of Alt. As with the other modifiers, releasing Alt temporarily redefines the no-change axis to go through the point where Alt was released. For example, imagine you made your color darker by Ctrl-dragging toward the bottom edge of the window. If you then need to make it less saturated but can't Shift-drag it any lower because there's no room, without releasing the mouse, simply Alt-drag it upward to a convenient spot and then Shift-drag downward as needed. You can also *start* dragging from the swatch with Alt to avoid any change of color until you reach a more convenient position for adjusting it.

For example, select a green rectangle and first turn it greenish-blue by dragging away from the fill swatch and slightly above the 45-degree line. Then, without releasing the mouse, press Ctrl and drag a bit to the right to darken the color; next, press Shift, release Ctrl, and adjust saturation. You can press and

release Ctrl and Shift as many times as necessary during a drag; when you are finally satisfied with your color, release the mouse to commit the change.

In addition to fine adjustments, here are some quick color gestures for common color changes:

- Ctrl-drag the swatch to the right and down to paint all selected objects black.
- Ctrl-drag the swatch up and to the left to paint all selected objects white.
- Shift-drag the swatch to the right and down to desaturate the selected objects' color (turn them to gray).
- Shift-drag the swatch up and to the left to maximize saturation of the selected objects' color.

When several objects or gradient stops with different colors are selected, the selected style indicator shows their *averaged* color. If you adjust that color by gesturing, the changed color will be assigned back to all selected objects or stops, in effect eliminating any difference between them. This makes the feature less useful than it could be; if you want to adjust many different-colored objects preserving their relative differences, use the color modes of the Tweak tool (8.9), color adjustment extensions (8.10.1), or color filters (8.10.2).

8.8 The Dropper Tool

The Dropper tool allows you to pick a color (and, optionally, opacity) directly from any point or area of the drawing and assign it to the selected objects (or to selected gradient handles, 10.4.2). With this tool, you can use your own drawing as a palette, easily reusing colors you have already created for something else. It is also indispensable when you need to combine and blend vector objects with an imported bitmap.

The important thing about this tool is that when picking colors, it pays no attention to what *object* you click; instead, it simply takes the color of the clicked *pixel* in the rendered screen image. This means you can easily pick colors from bitmaps, stacks of semitransparent objects, blur fringes (17.1), or the middle of a gradient. The tool sees exactly what you see; if an object is too small to be rendered at the current zoom, you can't pick out its color with the Dropper tool either. Also, if you click the edge of a black object on a white background, you may get a midrange gray color of the anti-aliasing pixel in that point (see Figure 1-1 on page 2).

8.8.1 Sampling

When you switch to the Dropper tool (F7 or D), the status bar starts to report the color located directly under the cursor. So, if you want to know the color of some area, you don't even need to click—just hover your mouse over that area and read the status bar. Also, at any time you can press Ctrl-C to copy the color under the cursor to the clipboard (in the form of an `RRGGBBAA` hex string); from there you can paste it, for example, into the Fill and Stroke dialog's RGBA field or to any external program.

8.8.2 Assigning

You can assign colors to selected objects (or gradient handles) either by clicking or by dragging. With a click, you just take the color of the screen pixel that happens to be under your mouse cursor. With a drag, you create a circular area centered at the point where you started dragging, and when you release the mouse, the tool *averages* the colors of all pixels inside this circle, as shown in Figure 8-7.

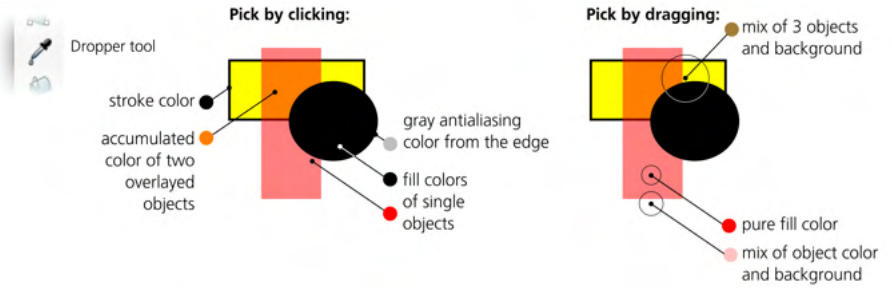


Figure 8-7: Using the Dropper tool

Picking with averaging is especially useful for sampling colors from bitmaps. For example, if you need to create a vector object that would blend with the cheek of a face on a photo, picking single pixels from the cheek is unlikely to work: the picked colors will be too light or too dark due to the nonuniformities of the photo's texture. If, however, you average the colors from a circular area covering a good part of the cheek, the result will be much more realistic.

By default, the Dropper assigns its colors to the fill of the selected objects. When you Shift-click or Shift-drag, it will instead change their stroke.

8.8.3 Opacity

Using the two toggle buttons on the controls bar of the Dropper tool, you can change the way it treats transparency (Figure 8-8). The Opacity: Pick button controls whether the opacity under the cursor is picked, and the Assign button controls whether that picked opacity value is assigned to selected objects. (When Pick is off, the Assign button is disabled.)

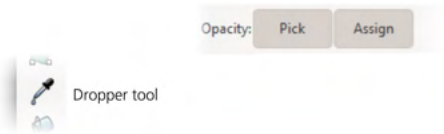


Figure 8-8: Toggle buttons of the Dropper tool

NOTE *The opacity that the tool picks is the accumulated opacity of all objects at the click point. For example, if you click a single object with 50 percent opacity, the picked value is 50 percent. If two such objects overlay, the picked value is 75 percent. Empty canvas is always considered fully transparent; the Dropper tool ignores the opacity of the page background (as set in Document Properties, 3.5.3).*

Suppose you have a bright red (FF0000) object with 50 percent opacity, which makes it appear pale red. Now, imagine you select some other object and try to pick the color from the red one. What happens?

Opacity: Pick **button is off**
: The tool will pick the pale red color (FF8080) and no opacity. In other words, the opacity will already be "multiplied into" the color. The status bar will report *FF8080 under cursor* and that is the color the selected object will get—whereas the opacity of the selected object, if any, will not change. Only in this case, the visible color of the page background will be mixed in.

Opacity: Pick **button is on**
: The tool will pick the actual bright red color (FF0000) and the 50 percent opacity *separately*. The status bar will display *FF0000 alpha 0.5 under cursor*. Now, when you click, the result depends on the other button, Assign.

  If Assign is on, both color (FF0000) and opacity (50 percent) will be assigned to the selected objects' fill or (with Shift) stroke. Note that the opacity will become the fill or stroke opacity of the selected objects, not their master opacity (8.3).

  If Assign is off, the bright red color (FF0000) is assigned to the selected objects' fill or stroke, whereas the picked opacity is simply discarded and 100 percent opacity is assigned. For example, if you select an object with semitransparent fill and click that object *itself* with Pick on and Assign off, the object's fill will lose its opacity but keep its color.

8.9 Color Tweaking

You've already seen some of the capabilities of the Tweak tool in the chapter on transformations (6.10). Let's look at its two modes for changing colors of objects, *Color Paint* and *Color Jitter*. Both of these modes work on flat color paint as well as on gradients (10.6). Refer to 6.10 to review how the tool's Width and Force parameters work.

8.9.1 Color Paint

The Color Paint mode is used to apply color to the selected objects *under the brush*, as shown in Figure 8-9. The color being used—more precisely, the *style* because it includes both fill and stroke—can be seen in the style swatch at the right end of the tool's controls bar (above the canvas). To change this applied style, simply click the color palette (or use any other style-editing command, such as the Fill and Stroke dialog) while you are in this mode of the Tweak tool.

NOTE *Unlike all other tools, in the Tweak tool in Color Paint mode, you cannot assign style directly to selected objects. Any style-setting commands are "intercepted" to change the tool's style instead.*

The fill from the tool's style applies to the fills of the painted objects, and the stroke applies to the strokes. If the tool's style has no fill or no stroke, it won't affect fills or strokes, correspondingly. For example, if you want to color the fills of objects blue but leave their strokes untouched, assign blue fill to the tool's

style (just click blue on the palette) but set its stroke to none (middle-click the **Stroke** swatch in the status bar).

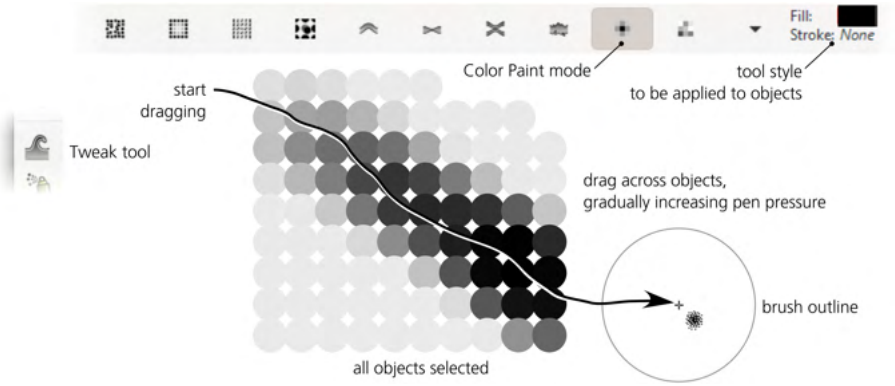


Figure 8-9: Using the Color Paint mode of the Tweak tool

This mode literally paints on objects, *gradually* shifting their colors toward the target color. For example, if you keep painting with yellow fill over a blue-filled object, the object will first become greenish blue, then green, then yellowish green; eventually it will be the exact yellow color you're painting with.

Painting with Shift pressed *inverts* the color you're applying (for example, when painting with yellow, Shift will gradually apply blue).

The speed of this gradual transition depends on both the Force value and, if you have a pressure-sensitive tablet, the pen pressure. Also, since the brush is "soft," objects touched by the periphery of the brush are less affected than those hit by the brush center.

8.9.2 Color Jitter

The Color Jitter mode, instead of applying a specific color, *jitters*—randomizes—the colors of the objects it touches. The force of the action determines the amount of randomization—that is, how far the colors deviate from their original values, as shown in Figure 8-10.

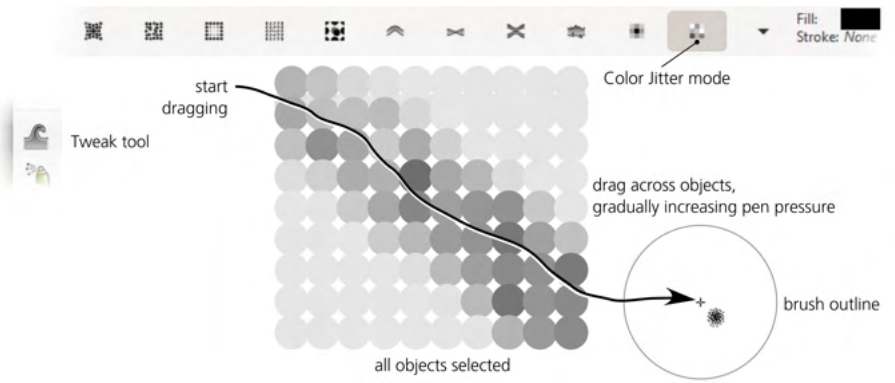


Figure 8-10: Using the Color Jitter mode of the Tweak tool

8.9.3 Channels

On the Tweak tool's controls bar, to the right of the Mode buttons, there are four Channels toggles: H, S, L, and O. This is where you turn on and off the tool's action on the object's hue, saturation, lightness, and opacity, correspondingly.

For example, if you want to raise the saturation of some part of your drawing without changing any hues, select some maximum-saturation color (such as pure red) and turn off all Channels buttons except S. Or you can replace the hues without affecting saturation or lightness (with only H turned on), or lighten/darken all colors without changing their hues and saturations (only L turned on). Enabling O applies the master opacity from the tool's style to the master opacity of objects (but not fill or stroke opacity).

8.9.4 Usage Notes

Color painting is similar to a soft brush in a raster editor (such as GIMP or Photoshop). Even though the tool works as a brush, it still applies its color to vector objects, which behave as vector objects usually do. For example, if you want to change the tint of a face in your drawing, and if a hand in the drawing is part of the same path object as the face, that hand's tint will change too, even if it's located far from the point you are painting. Still, even with this limitation, color painting allows you to quickly and intuitively make adjustments that would be awkward with traditional vector tools.

Drawings with scatterings of small independent objects are especially suitable for color painting with the Tweak tool. Examples include:

- Freehand drawings with the Calligraphic pen (14.2), consisting of many separate strokes.
- Object sprayings made with the Spray tool (4.7).
- Patterns made with the clone tiler (16.6). You will need to unset the fill and/or stroke on the original object and use the Color tab of the Create Tiled Clones dialog to assign some initial color to the clones—this will make them paintable with the Tweak tool without unlinking.
- Imported vector art with many small tiles, such as 3D-to-vector conversions where a smooth shape is approximated with polygons or gradient meshes imported from Adobe Illustrator's AI files (B.5), which Inkscape renders as lattices of small polygons. Although Inkscape now has its own Mesh tool (10.7), it cannot import AI meshes as meshes—but color painting on the imported tiles may sometimes be almost as good as editing the mesh.
- Text converted to paths, where each letter is a separate path (15.5).

Color tweaking can also be useful for compositions with just a few objects or even for single objects. Unlike all other color selection methods, painting with the Tweak tool implements the *color mixing* metaphor, which may be more natural for an artist than RGB sliders or even the HSL color wheel.

For example, start with a rectangle of pure blue color. Switch to the Color Paint mode, pick a number of different colors from the palette, and apply light touches with minimum force and minimum pen pressure. Add a little green, a

little brown, a little yellow, and so on—until you have the exact mixed tint you need. Similarly, you can whiten or blacken any object by adding white or black.

You can also use color tweaking to darken, lighten, saturate, desaturate, add a tint, or color-jitter your entire drawing. Just select all in all layers (Ctrl-Alt-A), zoom out, expand your brush so it covers all of the drawing, and apply a little color tweaking (with minimum Force) that will thus affect all visible objects.

8.10 Color Extensions and Filters

You've already seen a number of Inkscape's style setting commands and tools, each with its own approaches and capabilities. Naturally, each of them has downsides, too. In particular, the Fill and Stroke dialog and the selected style indicator cannot edit many different colors without unifying them. On the other hand, although the Tweak tool can adjust multiple colors independently, it actually requires you to paint on the canvas, so it can be slow and imprecise.

8.10.1 Color Extensions

A group of extensions in the Color submenu of the Extensions menu allows you to adjust all colors in the selection at once. These commands affect both fill and stroke colors, including colors of gradient stops, but excluding bitmaps or patterns. They include:

- A full set of *HSL adjustments* (increasing and decreasing hue, saturation, or lightness by 5 percent).
- Brighter and Darker (adjust brightness up or down by 10 percent).
- Desaturate (set HSL saturation to zero).
- Grayscale (equalize the three RGB channels; the result is largely similar but different from that of Desaturate).
- Black and white with adjustable threshold; colors brighter than the threshold become white, others become black.
- Negative (for example, convert black to white, yellow to blue, and so on).
- Commands for removing or swapping the Red, Green, and Blue channels.
- Replace color to replace any color you specify with another.
- A Custom command where you can provide your own formulas for modifying the color channels, using the values of other channels if necessary.

8.10.2 Color Filters

A better way to manipulate colors of objects is by using SVG filter effects (Chapter 17), specifically the preset filters from the Filters ▸ Color submenu. Compared to extensions, SVG filters are nondestructive (the original colors of objects are preserved and can be restored simply by removing the effect) and work on everything, including bitmaps and patterns. Filters in Inkscape can also be interactive: many of them have adjustable parameters, and if you enable the

Live Preview checkbox in the filter's dialog, you will be able to experiment with the parameters watching the result live on canvas before you commit.

In past versions of Inkscape, filters were much slower to render—but this is no longer the case, especially for the relatively simple color filters. You can create your own color filters (using the Color Matrix primitive, Chapter 17), but doing so is not for the faint of heart; fortunately, Inkscape has many ready-to-use preset filters to cover most needs—perhaps even too many for this book to cover every one. Still, here are the most universally useful filters from Filters ▸ Color:

- Color Shift rotates all the hues by a given amount; you can also increase or decrease saturation.
- Colorize shows your drawing as it would look through colored glass; you can select the color with a color chooser in the dialog.
- Duochrome re-paints your drawing in two tones—for example, with blue shades and yellow lights; you can select both colors.
- Extract Channel lets you suppress all color channels except one (R, G, B, or C, M, Y); the extracted channel can be converted to alpha.
- Fade to Black or White lets you naturally blend, by an adjustable amount, your drawing with a black or white background.
- Grayscale converts to grayscale; unlike the extension, you can interactively adjust how much each channel (out of R, G, B, and L) participates in the result.
- Lightness-Contrast works pretty much as you would expect.
- Invert lets you selectively invert some of the channels.
- Soft Colors creates a soft glow around the color boundaries; this filter is not interactive, but you can adjust the color of the glow (yellowish green by default) in the Filter Editor (17.5).

9

STYLE: STROKE AND MARKERS

While not very common in artistic drawings, stroked paths—outlines, frames, arrows, connectors, and so on—are very common in technical drawings such as plans or flowcharts. A sizable share of all SVG style properties control the appearance of stroke, so it deserves a chapter of its own.

While stroke in SVG is quite rich and can serve a lot of purposes, you may want some features that are not supported or are available only via workarounds. In particular, SVG stroke always has *constant width* (it cannot get wider or narrower along the path); to emulate variable-width stroke, use the Calligraphic pen (14.2) or path effects (Chapter 13). Also, while stroke can have a dash pattern (9.4) and markers attached to its nodes (9.5), you cannot stroke a path with a brush or repeated pattern that would follow the bends of the stroke—although, again, this is possible via path effects (13.3.2). You can always use a standard SVG rectangular pattern (10.8.1) to paint a stroke exactly as you would a fill, but the pattern in this case is simply superimposed—it is not distorted to follow the trajectory of the stroke.

9.1 Stroke Width

The most important property of the stroke is its *width*. Like any other length in Inkscape, width can be measured in a number of different units. There are currently two places in the UI where you can see and change the stroke width of the selected objects: the selected style indicator in the status bar (bottom left of the window, to the right end of the stroke swatch) and the Stroke style tab of the Fill and Stroke dialog (Figure 9-1).

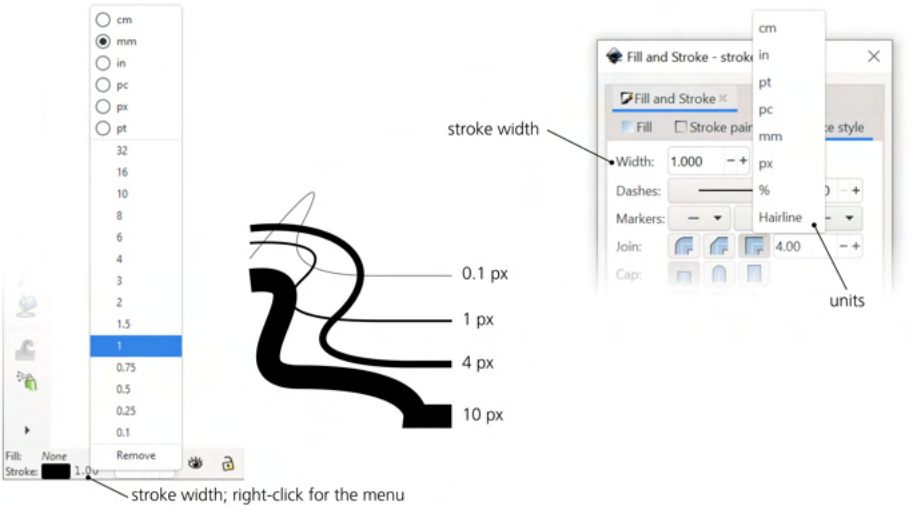


Figure 9-1: Stroke width in the selected style indicator's right-click menu (left) and the Fill and Stroke dialog (right)

In the selected style indicator, one way you can change the stroke width is by right-clicking directly on the number next to the stroke swatch and choosing a value from the menu that pops up. In the same menu, you can choose the unit to be used for stroke width; by default it's px (SVG pixel).

Similar to editing fill or stroke colors with color gestures (8.7), you can also drag off the stroke width value into the canvas to change the stroke width of the selection interactively. Dragging above and to the left of the 45-degree no-change axis (from the point where you started dragging) makes the stroke in selected objects wider (up to 50 times the original width); dragging below and to the right makes it narrower (down to zero). For example, if you start with a 1 px stroke, dragging up and to the left will give you 50 px, and dragging to the right and down will reduce it to zero. This way, you can adjust stroke width very quickly without opening any dialogs or menus.

In the Fill and Stroke dialog, there's a regular editable field where you can type any value, as well as a unit selector.

NOTE *If all of the Stroke style tab is disabled (grayed out), this means your selected object has no stroke paint. To enable it, in the same dialog, go to the* ***Stroke paint*** *tab and choose a solid color or any other paint for the stroke.*

When you choose the Hairline option in the units drop-down for the Width value, the Width value itself is disabled. The selected objects' strokes will always look like they're a constant 1 px wide *regardless of zoom*. The object itself will zoom in and out as usual, but its stroke will never appear thinner or thicker.

HAIRLINE WIDTH IN SVG

Hairline width is implemented via the `vector-effect:non-scaling-stroke` CSS property, which is a recent addition to SVG, so not all SVG viewers support it. At the time of writing, it's not supported when exporting to PDF either.

Outline mode (3.14) shows *all* objects in the document as if they had a hairline stroke that remains 1 px wide regardless of zoom. Visible Hairlines mode (3.14) allows you to scale the stroke up but not down, so it is always *at least* one screen pixel wide no matter how far you zoom out. Unlike the Hairline setting in the Fill and Stroke dialog, however, these modes are just an Inkscape-specific editing convenience and do not affect the SVG code of the document.

9.1.1 Stroke Width in Multiple Objects

When the selection contains multiple objects with different strokes, the selected style indicator *averages* the stroke widths of those objects that have any stroke. For example, if one of the two selected objects has 3 px stroke and the other has 1 px, the indicator will show *2* (and the tool tip will say that this is an averaged value).

Now, if you set any width via the right-click menu or by dragging, the same width will be assigned to all selected objects that had any stroke in the first place. For example, if one of the objects has 3 px stroke and the other has no stroke, it will display *3* as stroke width and *Different* in the stroke paint swatch, but any new width you set will be assigned *only* to the object that had 3 px stroke.

The Fill and Stroke dialog behaves differently. When different stroke widths are detected in the selection, the stroke width unit switches to the percent unit (%) and the displayed value is 100%. If you change it, for example, to 200% and press Enter, each stroke width in the selection will get two times wider than it was before. All stroke widths that were different will remain different, and the displayed value will again be 100%.

In the same dialog, you can just as well switch the unit selector from % to any absolute unit and assign the same stroke width to any number of selected objects. On the other hand, even if you have a single object selected, you can still switch to the percent unit and specify a new width as a percentage of the old.

9.2 Join

Stroke always follows a path, and a path can have sharp turns, called *cusps*. Typically, a cusp is a node (12.5.5) where two path segments join at an angle, but you don't need a node to make a cusp: a sharp cusp can be created even in the middle of a Bézier curve (see Figure 12-6 on page 227, bottom left).

The way the stroke behaves at the cusps is determined by the two style properties: *join type* and *miter limit*, editable on the Stroke style tab of the Fill and Stroke dialog, as shown in Figure 9-2. The three possible join types, represented by the three toggle buttons, are Round join, Bevel join, and Miter join (default).

Round join
: This simulates the effect of tracing the join with a perfectly round pen. The outer shape is a smooth circular arc whose center is on the path centerline at the cusp point. This option is the most natural for largely curvilinear paths where occasional cusps might look out of character if not rounded.

Bevel join
: This join is basically the same as a Miter join (see below) but with the miter limit set to 0. This means that for *any* angle, the corner is cut off by a *bevel*—a straight line perpendicular to the bisector of the cusp angle.

Miter join
: In this join type, the way it looks depends on how sharp the cusp angle is. For angles that are not too sharp, the outer outline of the stroke at the cusp point is continued by two straight line fragments, tangential to the stroke on both sides of the joint, until these straight lines cross. As a result, the joint is adorned by a sharp peak, called a *miter*, which becomes longer and sharper as the angle at the joint decreases, possibly reaching far beyond the position of the cusp node.

This leads to a problem, however. How long can the miter become? Obviously, when the angle at the joint becomes zero (which is perfectly legal), the miter will be *infinitely* long. To prevent this, the Miter limit control sets the *maximum length* of a miter in units of stroke width. For example, with the default miter limit of 4, any miter shorter than 4 stroke widths remains sharp-tipped, but as soon as you decrease the angle to make the miter longer than that, it will be cut—*beveled*—at the distance of 4 stroke widths from the joint.

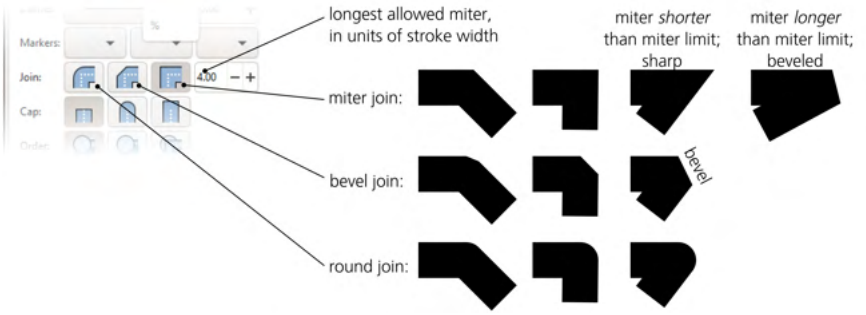


Figure 9-2: Stroke join controls in the Fill and Stroke dialog

Stroked text objects are where the artistic effect of different stroke join settings is perhaps the most obvious, as Figure 9-3 demonstrates.

Figure 9-3: The effect of join settings on stroked text

STROKE JOINS IN SVG

In SVG, the joins are controlled by the `stroke-linejoin` CSS property, which can take values of `miter`, `bevel`, or `round`. The miter limit is stored in a separate property, `stroke-miterlimit`.

9.3 Caps

An open path needs to know how to draw the ends of the stroke. Here, too, there are three options, somewhat similar to the three join types: stroke caps can be *butt*, *round*, or *square* (Figure 9-4).

Butt cap
: Bluntly cuts the stroke, perpendicular to the stroke direction, right at the end node of the path.

Round cap
: Adds a semicircular blob that smoothly rounds the end of the stroke.

Square cap
: Adds a half-square blob to the end of the stroke.

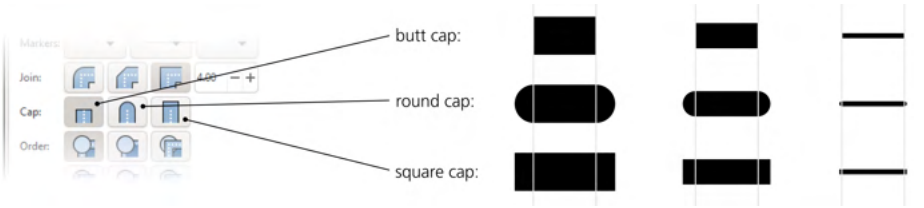


Figure 9-4: Stroke cap options in the Fill and Stroke dialog

Both round and square caps make an open path longer by its full width, with a half-stroke-width cap added to each end. Only with butt caps (which essentially means no caps), the path is exactly as long as the distance between its end nodes.

The effect of join and cap options is visible only on wide enough strokes or when zoomed in; for strokes that render at just two or three screen pixels, they make very little visual difference (except possibly for long miters).

STROKE CAPS IN SVG

In SVG, the stroke caps of a path are controlled by the `stroke-linecap` property, which can take values of `butt`, `round`, or `square`.

9.4 Dash Patterns

A stroke does not need to run solidly from end to end of a path. SVG allows you to stroke a path with a regular pattern of dashes separated by empty intervals of any length. If you know the corresponding CSS syntax, you can even create new dash patterns for your own use (use the **Selectors and CSS** dialog, 16.3). Otherwise, choose one of the many predefined patterns provided by the Fill and Stroke dialog, as shown in Figure 9-5.

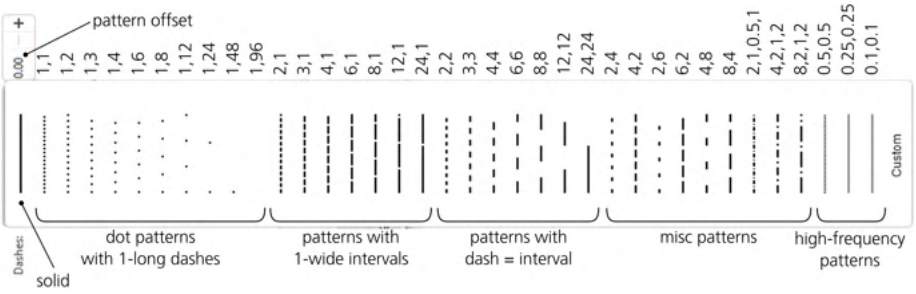


Figure 9-5: Dash patterns in the Fill and Stroke dialog

All dash patterns are defined in terms of stroke width, which means the pattern scales up or down proportionally when you make your stroke wider or narrower. Ordered roughly from the most common to the most exotic, the predefined patterns include:

- Dotted patterns that consist of dots (that is, stroked segments whose length is equal to the stroke width) with intervals of 1, 2, 3, 4, 6, and so on up to 48 stroke widths.
- Patterns with long dashes (1, 2, 3, and so on up to 24 stroke widths) and single-stroke width intervals.
- Patterns with equal dash and interval lengths, from 2 to 24 stroke widths.
- Patterns with varying dash and interval widths: 2 and 4, 4 and 2, 2 and 6, 6 and 2, 2 and 8, 8 and 2.
- Patterns where a long dash is followed by a short one.
- Patterns with equal dash and interval lengths that are shorter than the stroke width, from 0.5 down to 0.1 stroke widths (for example, a "square" path that is as long as it is wide will have five repetitions of the 0.1,0.1 pattern).

- Contrary to what you might expect, the Custom option does not allow you to edit a pattern in a nice visual way—it's just some seemingly random pattern that you can still only edit in the CSS editor.

The Pattern offset field next to the dash pattern selector allows you to *shift* the chosen pattern along the path by a given distance (again, in units of stroke width). For example, if you use a two-dash, two-interval pattern but want your path to start with an interval, shift it by 2.

DASHES IN SVG

In SVG, the dash pattern of a stroke is specified in the `stroke-dasharray` property. It can take values of `none` (solid line) or a comma-separated list of values where each odd value gives the length of a dash and each even value gives the length of an interval. For example, a specification of `stroke-dasharray:2,1,0.5,1` means that the dash of length 2 is followed by an interval of length 1, which is followed by a short dash of 0.5 and another interval of 1 (all lengths are in the units of stroke width).

The list of the predefined dash patterns in the Fill and Stroke dialog is stored in your *preferences.xml* file, inside the `group` element with `id="dashes"`. By editing the children of this element, you can add, remove, or change the patterns available in the Inkscape UI.

Note that the stroke caps (9.3) affect dashes, too. If you set a path to use round or square caps, they will be added to both ends of each dash. As a result, each dash will become longer by one full stroke width compared to its length when using the default butt caps. For example, the 1,1 dash pattern with round caps loses all its intervals; the round caps of adjacent dashes, each 0.5 stroke widths in length, now touch each other, as Figure 9-6 demonstrates.

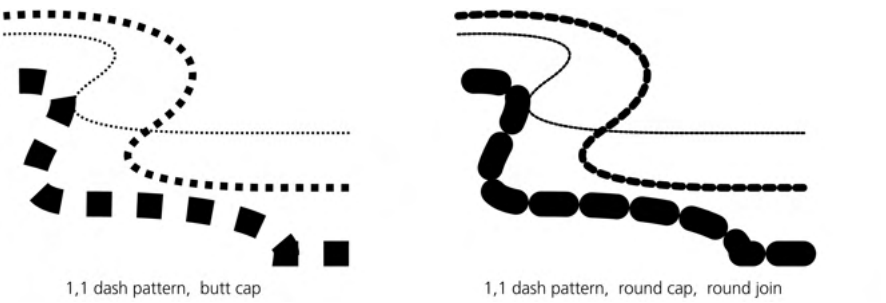


Figure 9-6: The effect of stroke caps on dashes

As a consequence, to create a pattern of round dots following a path, you need to use a dash pattern with *zero-length* dashes and round caps. Inkscape does not list such a pattern in the Fill and Stroke dialog, so you will need to create it manually. First, assign to your path a pattern with the *interval length* you need, and then edit the `stroke-dasharray` property of your path in the Selectors and CSS dialog to set its *dash length* (first value) to 0. Of course, without round or square caps, such a pattern will render the path completely invisible (which is the reason

it is not included in the standard patterns—remember that the default caps setting is butt).

One interesting use of dot patterns with very wide intervals (such as 1,48) is quickly creating a random scattering of dots. Draw a spiral-like doodle with the Pencil tool (14.1.2) and assign a 1,48 dash pattern to it to turn it into a cloud of seemingly unconnected dots, as shown in Figure 9-7.

Figure 9-7: Using a dash array with large intervals to imitate a random scattering of dots

Also, the regularity of dash patterns plays nicely with spirals—a shape that is also regular but consists of progressively longer and longer turns (11.6). An interplay of the equidistant dashes and gradually devolving paths can produce enchanting patterns (Figure 9-8).

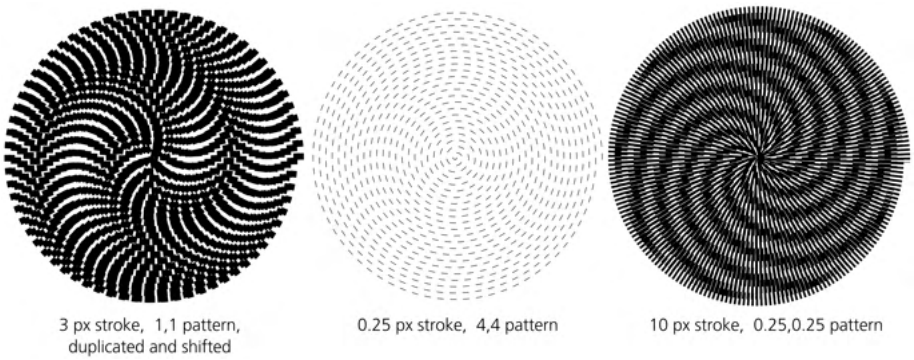


Figure 9-8: Dashed spirals are a form of art.

9.5 Markers

Markers are arbitrary objects (or even groups of objects) attached to a path and displayed as part of that path at (some of) its nodes (12.1). The most common use of markers is for creating arrowheads in diagrams and flowcharts.

A path can have three different types of markers: on its start node, intermediate (mid) nodes, and the end node. Each copy of the marker is positioned at the node and rotated so as to follow the direction of the path at this node. The size of markers is proportional to stroke width; simply make your stroke narrower or wider to scale all markers on a path.

The Stroke style tab in the Fill and Stroke dialog contains three drop-down lists where you can select start, mid, and end markers for the selected paths (Figure 9-9).

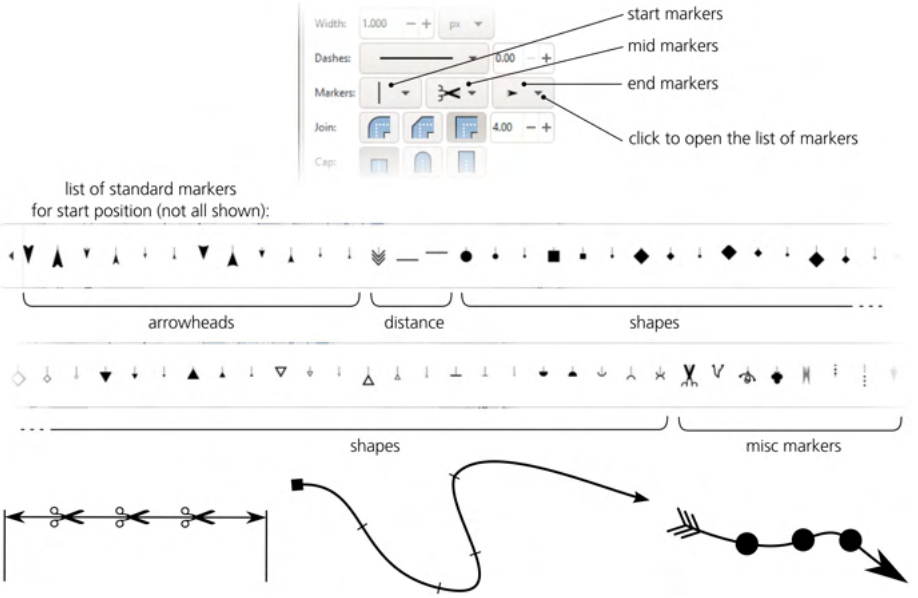


Figure 9-9: Choosing markers for a path

These three lists show all the same markers (in other words, you can use any marker in any position), but the preview thumbnails show them applied to the start, mid, and end of a horizontal straight line path, so you can get an idea of how the marker will look on the actual path.

Let's look at what markers are in Inkscape's stock list.

Arrowheads

There are two types of arrowheads: one is dart-like with two straight lines at the back; the other is delta-shaped with a concave circular arc at the back of the arrow. Each type of arrowhead comes in three *sizes*: large, medium, and small (these are not absolute sizes but relative to the stroke width). Also, each size has two *orientations*: start and end.

For example, if you want your arrowheads to point *outward* from the path, choose one with a start orientation as the start marker and one with end orientation as the end marker. If you mix this up, your arrowheads will point *inward* from the ends of the path. Or you can choose the same end-orientation arrowhead for all three positions (start, mid, and end) to have all arrowheads on your path point in one direction toward the end of the path. Use **Path ▸ Reverse** from the menu to reverse a path if necessary (see 12.1.1 for details).

Arrow tails

An arrow tail marker is available only in one size (matching the large arrowheads) and one orientation (making sense as an end marker—that is, oriented toward the path start).

Distance measurement markers
: Distance measurement markers are just arrowheads with added perpendicular straight lines at the tips. There are two orientations, one for a start marker and the other for an end marker.

Geometric shapes
: There is a collection of geometric shape markers: round dots, squares, diamonds (squares rotated by 45 degrees), equilateral triangles, straight line stops, and filled and empty half-circle marks. Most of them have solid black and hollow variants. Just as with arrowheads, these markers come in three sizes (large, medium, and small); some also have start and end orientations (although for symmetric markers such as diamonds, the orientations differ only in the position of the marker relative to its node). The start and end variants of triangle markers can be used as just another arrowhead shape.

Misc markers
: There are several fancy markers, notably Scissors (assign it to mid markers to create a typical "cut-off line") and the "infinite line" ellipsis endings.

If your document already uses some markers of its own, these markers will be added to the top of the drop-down marker menus, separated from the stock Inkscape markers below. To remove markers from a path, select the topmost empty line in the list.

MARKERS IN SVG

When you assign a marker to a path, Inkscape places a copy of the marker into the document's `defs` (A.4) and makes the path refer to the marker in `defs`. The list of the markers above the separator in the marker menus is simply that of the markers in `defs`; to remove markers you no longer use anywhere, use the **File ▸ Clean Up Document** command.

9.5.1 Mid Markers and Nodes

Start and end markers are simple in that their position on the path is never a surprise (although their orientation may sometimes be, if the end node has a very short Bézier handle that has little effect on the shape of the curve but may rotate the marker at this node in an unexpected direction). Mid markers are more interesting: they are located at the middle nodes (12.1) of a path, and the positions of these nodes may not be what you need or expect.

A simple use case is a path consisting of straight line segments (with no Bézier curves). On such a path, mid markers will be displayed at the corners. Use markers that don't have end or start variants and are thus positioned symmetrically around the node—for example, dots or squares (Figure 9-10).

You may want to fill a path evenly with mid markers, similar to the way the dash pattern is regularly repeated along the path. This is easy for a straight line path where you can add or remove as many mid nodes as needed without affecting the straight line shape. In a more complex path, however, some nodes may be necessary to give the path its shape, and you cannot move those nodes along

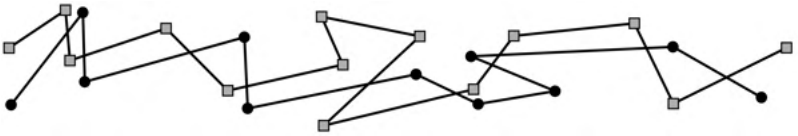


Figure 9-10: Mid markers at the path joints

the path without distorting that shape. You also cannot choose *not* to apply markers to these nodes: in SVG, mid markers apply to all mid nodes without exception. As a result, it may be impossible to distribute markers along a path perfectly evenly; however, the shorter your desired interval between markers and the simpler the path, the less noticeable this unevenness can be.

How can you add nodes to a path without changing its shape? A single new node can be added in the Node tool (12.5.3) by double-clicking or Ctrl-Alt-clicking anywhere on the path. For an even distribution of nodes, however, another shortcut is more useful: Insert creates a new node in the middle of each segment between selected nodes (and adds the new node to the node selection so you can keep multiplying nodes by pressing Insert repeatedly).

For example, selecting both nodes of a two-node path and pressing Insert adds one new node in the middle. Now, you have three nodes selected with two segments between them, so pressing Insert again adds two more nodes; another Insert adds four more nodes, and so on. In this simple case, all nodes will be distributed evenly at all times. However, if your path already had some mid nodes, selecting all nodes and pressing Insert repeatedly will add nodes more profusely in areas that had more nodes to begin with, as shown in Figure 9-11.

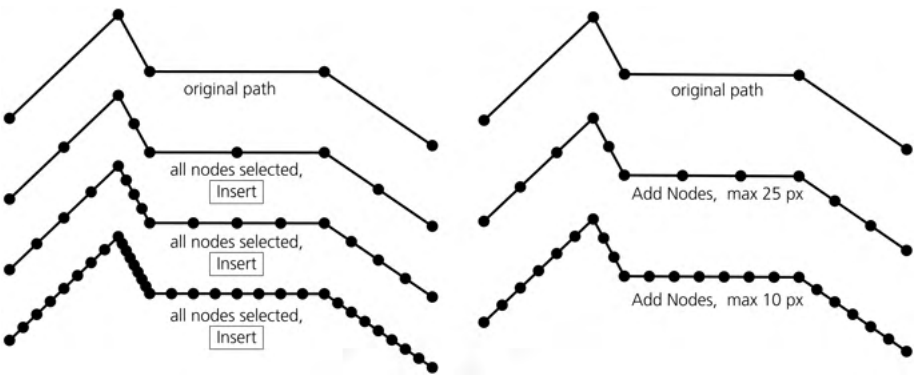


Figure 9-11: Adding mid nodes by pressing Insert repeatedly (left) and by using the Add Nodes extension (right)

A better approach to creating evenly distributed nodes is the Extensions ▸ Modify Path ▸ Add Nodes extension (13.4.2.1). In it, you can specify either the maximum segment length (in px units) or the exact number of segments you need, and the algorithm will add nodes (without moving or deleting any existing nodes) to produce an even node spacing satisfying these constraints.

To delete an individual node in the Node tool, Ctrl-Alt-click it, or select it and press Delete (12.5.3). A good method for deleting multiple nonessential nodes (those that were added only for the sake of markers and do not affect the

shape of the path) is the Simplify command (12.3). Of course, Simplify cannot really *know* which nodes are essential and which are not; it tries to guess—and usually performs acceptably, although undesired nodes and small shape distortions are likely to happen.

9.5.2 Coloring Markers

Typically, connector lines in diagrams and flowcharts are black, so the stock markers offered by Inkscape are either solid black or black with white filling. However, if you have a blue or red connector line, normally you would want the arrowhead on that line to be blue or red too. That's what Inkscape does by default: when assigning a marker to a path, it re-colors it to match the color of the path's stroke. When you change the stroke color of a path with markers, its markers get updated too.

This color-matching is not the standard behavior in SVG—it's just what Inkscape does to be helpful. Behind the scenes, whenever you change stroke color, Inkscape creates a copy of each assigned marker, re-paints it with the new color, and re-assigns it back to the path. That's why you will see the different-color copies of the markers you have used at the top of the lists in Fill and Stroke (which, as you remember, list the document's custom markers first before the standard ones). The good news is that if you don't like this behavior, you can disable it in Preferences ▸ Behavior ▸ Markers, as shown in Figure 9-12.

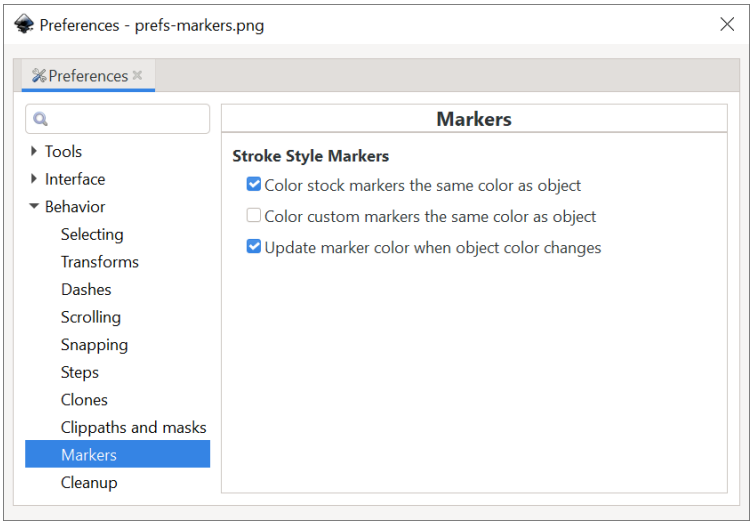


Figure 9-12: Marker behavior preferences

One reason to disable automatic marker coloring is if you want to use a path only as an invisible string for a rosary of markers—in other words, you want to see the markers but hide the stroke itself. For this, first assign the markers to a stroked path and then remove the stroke (for example, by middle-clicking the stroke swatch in the selected style indicator, 8.6). If you had set Update marker color when object color changes to off, this will give you what you need: visible markers on an invisible path.

When markers and their stroke have different colors, one more thing to take into account is the order of rendering (9.6). By default, stroke is painted first, so if it is different from its markers, it will only show through from underneath the markers if they are transparent.

[1.1] Automatic coloring of markers is a relatively recent Inkscape innovation; before it was available, the only way to do this was by Extensions ▸ Modify Path ▸ Color Markers. This extension is still available and is sometimes useful because it has some additional options. With it, you can not only paint markers to match the object but also swap fill and stroke colors, optionally include alpha, or just provide your own colors for fill and stroke. (If you want a more complex paint job for your markers, see the next section on how to turn a marker into a regular editable object and back.)

9.5.3 Creating New Markers

A marker does not have to be a single object painted with a solid color. It can consist of any number of objects, grouped or not, with any paint, opacity, or even blur properties—in other words, anything Inkscape can draw can be a marker on a path. Applying such complex markers to paths with many added nodes (9.5.1) can produce amazing compositions. Let's look at how to create markers out of arbitrary objects.

Generally, it's as easy as selecting the object or objects and choosing Objects to Marker from the Object menu. Selected objects disappear, but in the Stroke style tab of Fill and Stroke, you will see your new marker in the top part of the list of markers (just before the stock markers). You may need to close and reopen the Fill and Stroke dialog to refresh the marker list.

When creating a new marker, Inkscape assumes that the original objects are oriented as they should be on a horizontal path that goes from left to right. For example, if you have a new arrowhead that you want to use as an end marker, make it point horizontally to the right before converting it to a marker.

Similar to standard markers, the user-created marker will scale up and down as you change the stroke width. Its initial size (the size of the object that you have turned into a marker) will correspond to the stroke width of 1 px.

Each marker has an *anchor point*—the point that will be placed on the node to which that marker is attached. When you create a new marker, Inkscape uses the transformation fixed point (6.4) of the (first) selected object as that anchor point (Figure 9-13). By default, the fixed point is in the center of the object's bounding box (4.3), which means the newly created marker will be centered around its node. If you move the fixed point to one of the corners of the object, the new marker will touch its node by that corner.

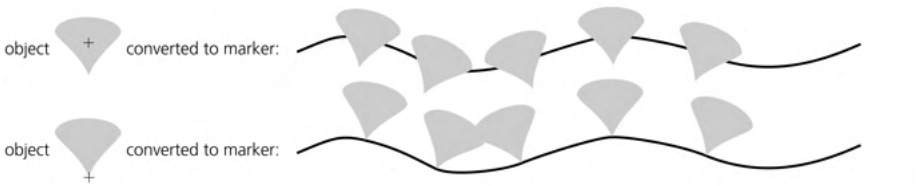


Figure 9-13: Creating a new marker from a selected object

Can you do the opposite and convert a path marker to an object (for example, to edit it and make it into a marker again)? Yes, although this involves destroying the path that the marker was applied to (so you may want to make a copy of the path first). Use the Path ▸ Stroke to Path command. It turns the stroke into a filled path (12.1.3), but also, if the original path had markers, it groups the converted path with objects that represent former markers. Ungroup, choose one of the former markers, rotate it into the default orientation, and edit as necessary.

NOTE *You can turn a clone (Chapter 16) into a marker and then continue to edit the original object, with its marker clones updated live.*

9.5.4 Advanced Markers

SVG markers have a few other useful options that are not yet available via the Inkscape UI. To change them, you need to use the Selectors and CSS dialog (16.3) and Inkscape's XML Editor (4.10) together. Open them both. Select an object with markers, in Selectors and CSS find the marker property you're interested in (`marker-start`, `marker-mid`, or `marker-end`), and click the green arrow at the right end of that line. This will scroll the XML Editor to the `svg:marker` element used by your object. Here's what you can change via this element's attributes:

- By default, markers rotate to orient themselves along the path direction. If you want your marker always to have the same orientation regardless of how the path goes at this point, remove the `orient="auto"` attribute.
- If you don't want the marker to scale up and down when the stroke width is changed, add the attribute `markerUnits` with the value `userSpaceOnUse`.

9.6 Rendering Order

[1.1] As you have seen, a single solid object—not a group—can still have up to three distinct components: its fill, its stroke, and its markers. The only unresolved question is in which order to render these intrinsic components. Since you cannot simply rearrange them (as you would with separate objects in a group), SVG governs this with a style property of its own. This property, editable in the Stroke style tab of Fill and Stroke (Figure 9-14), can have one of the six values that represent all possible orderings (bottom to top) of the three components:

- fill, stroke, markers (the default)
- stroke, fill, markers
- fill, markers, stroke
- markers, fill, stroke
- stroke, markers, fill
- markers, stroke, fill

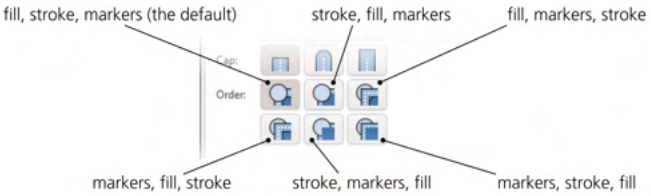


Figure 9-14: Rendering order buttons on the Stroke style tab of Fill and Stroke

STROKE ALWAYS CENTERED ON PATH

Adobe Illustrator has an option of painting stroke completely on the inside or outside of the path. This is not possible in SVG; stroke is always centered on the path (this may change in future versions of SVG, though).

Whenever fill is on top of stroke, it goes up to the line of the path on which the stroke is centered. Thus, if the fill is opaque (8.3), the visible result of putting it on top of the stroke is almost exactly as if the stroke were half its width.

10

GRADIENTS, MESHES, AND PATTERNS

Gradients are important in vector graphics because they are the cheapest and most universal way to depart from the lifeless flat look of solid color fills. Designers can create amazingly complex and photorealistic art using nothing but carefully laid out gradients. For all their versatility, gradients are relatively simple to create and edit, render quickly, and they are almost universally supported in SVG software—none of which is quite true, for example, for SVG filters such as blur (17.1).

Basically, a *gradient* is a smooth transition between two or more colors. *Color* also includes a level of opacity; this means that you can, for example, make a gradient from opaque red to transparent red, with semitransparent red shades in between. SVG supports two types of gradients: *linear* (along a line) and *elliptic*, or *radial* (away from a center, with possibly unequal *axes* and a noncentral *focus point*).

After the plain linear and elliptic gradients, this chapter describes the *mesh gradients* that are much more powerful and flexible—even though, unfortunately, they are not supported by other SVG software at the moment. Finally, I discuss *patterns* that paint an object with repeated copies of another object; patterns have many similarities to gradients in the user interface.

10.1 Creating Gradients

The Gradient tool (G or Ctrl-F1) is where you create new gradients on objects. It is also the most natural place to *edit* existing gradients—although, once created, an object's gradient handles are also visible (so they can be dragged and colored) in the Node tool (12.5), all shape tools (Chapter 11), and the Dropper tool (8.8).

Creating a new gradient is easy. Just make sure you have the object (or objects) selected and start dragging on the canvas—from the selected object or from any point nearby. You will see an actual gradient appearing and following your mouse as you drag, and you will also see a system of *handles* (small on-canvas controls) connected by lines.

You don't need to drag *on* the selected objects—gradient handles can very well lie completely outside the object they apply to. You can select any number of objects and paint them all with the same gradient (more precisely, with multiple but coinciding gradients) with a single drag. Another way to quickly create a gradient that spans the entire object is to double-click that object with the Gradient tool.

To use the most common object selection shortcuts, you don't need to switch to the Selector tool: click to select, Alt-click to select under, and Shift-click to add to selection all work in the Gradient tool as well.

You can apply a gradient to the *fill* of an object, its *stroke*, or both (8.2). In Inkscape, a stroke gradient shows a pink connecting line between the handles, while a fill gradient has a blue line, as shown in Figure 10-1.

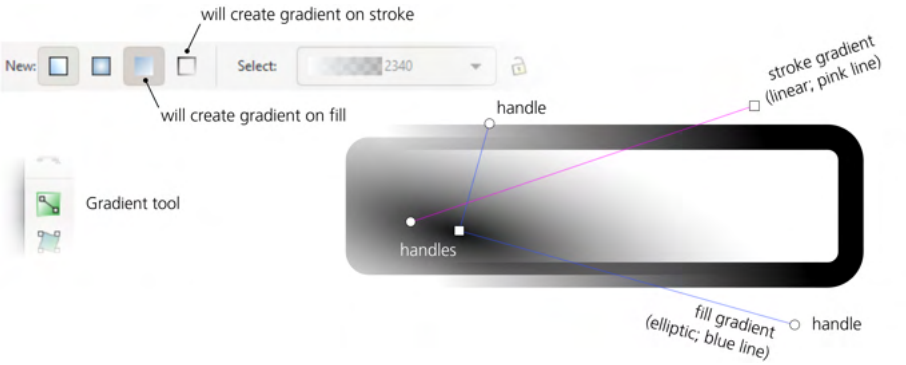


Figure 10-1: Different gradients on the fill and stroke of an object

If you are creating a gradient on an object that had flat color paint before, the initial gradient will go from the fully opaque to the fully transparent color of the object *on which you started the drag*. For example, if you have a yellow rectangle selected and start your drag on that rectangle, you will fill the object with a gradient going from opaque yellow to transparent yellow. You can, however, fill it with an opaque-blue-to-transparent-blue gradient if you have a blue-filled object somewhere and start dragging from it. To put it another way, the gradient you create applies to selected objects, but the color it uses may come from any object, selected or not, from which you start your drag. (If that's not what you wanted, don't worry—it's very easy to change gradient colors once it's created.)

Similarly, if the object where you started your drag already had some gradient, the tool will redraw it for the selected object, preserving the colors of that gradient (that is, preserving the *gradient definition*, 10.2). If you start dragging from an empty space, not from any object, the color or gradient of the *topmost selected* object will be used for the new gradient.

Of the many ways to *remove* gradients from an object, the easiest is to make sure none of the gradient handles are selected (all are white, not blue; if some are blue, press Escape once) and click any color on the palette (or Shift-click to remove gradient from stroke). The object will be filled or stroked by the flat color, replacing any gradient it had before.

When you transform (that is, move, scale, rotate, or skew, Chapter 6) an object that uses gradients on fill or stroke, all the gradient handles are transformed along with the object, so the gradient stays firmly put in place. However, the third Affect button on the Selector tool controls bar (6.11) can change that. If you uncheck this button (checked by default), gradients will remain fixed *relative to the canvas*, no matter how you transform the objects that use them. This can be useful, for example, if you want to scale an object up or down so that its size matches exactly the elliptic gradient it uses.

NOTE *Neither gradients nor any other paint (8.2) can be applied to bitmap objects (Chapter 18); if you want to apply a transparency gradient to a bitmap, use a mask (18.3) or shader overlays (10.6). Clones (Chapter 16) also cannot take gradients except when the paint of a clone's original is unset.*

Let's now look in more detail at the two types of gradients Inkscape can create: linear and elliptic.

10.1.1 Linear Gradients

A linear gradient goes along a straight line segment, and both ends of the segment (as well as, possibly, some points in the middle) have certain colors and opacities assigned to them. The transitions between color areas on the object are always perpendicular to the gradient line. This is the default kind of gradient that the Gradient tool creates; the linear gradient mode of the tool corresponds to the first toggle button on the tool's controls bar (Figure 10-2).

Figure 10-2: Creating linear gradients by dragging and double-clicking

NOTE *In the selected style indicator, linear gradients are marked with the letter* L *before the gradient swatch.*

A linear gradient has two freely draggable handles: a square at the start and a circle at the end of the gradient line. By dragging them around, you can change the direction and length of the gradient. The gradient repaints itself in real time as you drag a handle.

With Ctrl pressed, drawing a new gradient or dragging a handle of an existing one makes the gradient line *snap* to angle increments, by default every 15 degrees (you can change this value in the Behavior ▸ Steps page of the Preferences dialog, compare 6.3). Double-clicking an object in linear gradient mode creates a horizontal linear gradient that goes through the center of the object's bounding box.

10.1.2 Elliptic Gradients

To switch the Gradient tool to create *elliptic gradients* instead of linear, click the second toggle button in the tool's controls bar (above the canvas). Now, if you drag on canvas with some objects selected, an elliptic gradient will be created on those objects, as shown in Figure 10-3.

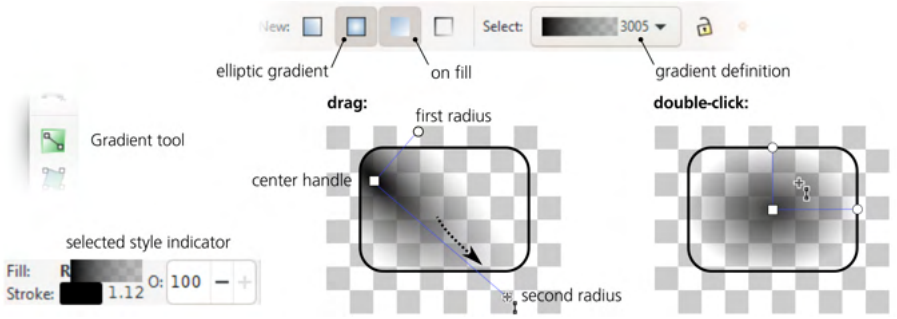


Figure 10-3: Creating elliptic gradients by dragging and double-clicking

NOTE *In the selected style indicator, elliptic gradients are marked with the letter* R *(for Radial) before the gradient swatch.*

An elliptic gradient has at least three draggable handles—the *center* (square handle) and the two perpendicular *radii* (circular handles). These handles let you move, stretch, squeeze, or rotate the gradient; you can make it into anything from a circle to a narrow ellipse, rotated at any angle. Moving the center handle moves the entire gradient (all handles); moving the radii stretches and rotates the gradient without moving the center. The two radii always remain perpendicular. As with a linear gradient, dragging a radius with Ctrl pressed *snaps* it to 15-degree angle increments.

When you start dragging to create a new gradient, you mark its center and are dragging out one of the radii. The other radius is created to be equal to half of the selected object's height—that is, if you start dragging from the center of the object and drag horizontally to the right edge of its bounding box, the ellipse

will be neatly inscribed into the bounding box. You can achieve the same effect simply by double-clicking an object.

NOTE *If you need a symmetric "bilinear" gradient with a mountain-like profile, an easy way to create it is via an elliptic gradient with one radius made very long, dragged far beyond the boundaries of the object. Then, the central part of the stretched ellipse will look almost indistinguishable from a couple counter-directed linear gradients.*

An elliptic gradient also has a fourth, normally hidden, handle: the *focus*. This is the point with the color and opacity of the gradient's central stop. Normally, it is merged together with the central handle and moves with it, keeping the gradient perfectly symmetric. To separate the focus—visualized by an X-shaped handle—from the center, drag away from the central handle with Shift. This creates an asymmetric, or *eccentric*, gradient (Figure 10-4). To merge the focus back, just drag it close enough to the central handle and it will snap.

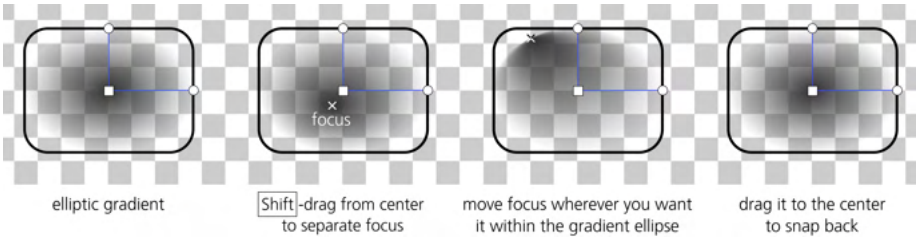


Figure 10-4: Creating an eccentric elliptic gradient by moving the focus

10.2 Gradient Definition

A gradient has a set of *stops*, each with its own color and transparency value. The minimum number of stops is the two *end stops*: the ends of the gradient line (linear) or the center and edge of an ellipse (elliptic). However, a gradient can also have any number of *middle stops* positioned somewhere between the end stops. For example, you may have a single middle stop at 0.5 of the gradient, which means it will always be exactly in the middle between the end stops. The complete list of a gradient's stops with their colors, transparencies, and relative positions is called the *gradient definition*.

Gradient definitions are document-wide resources. This means that any gradient you create or edit adds its definition to the list of all gradient definitions in your document. After that, you can assign that definition to any other gradient in this document, which means that gradient will take the colors and the relative positions of the middle stops from the definition (but the gradient's end handles will not move). The same gradient definition can be used by both linear and elliptic gradients. Use the drop-down list in the controls bar of the Gradient tool to view and select gradient definitions in the document (Figure 10-5).

The same list of gradient swatches is also available in the Fill and Stroke dialog as well as in the Paint Servers dialog (10.9). When you have one or more objects with gradients selected, choosing a definition from the list assigns it to all selected gradients.

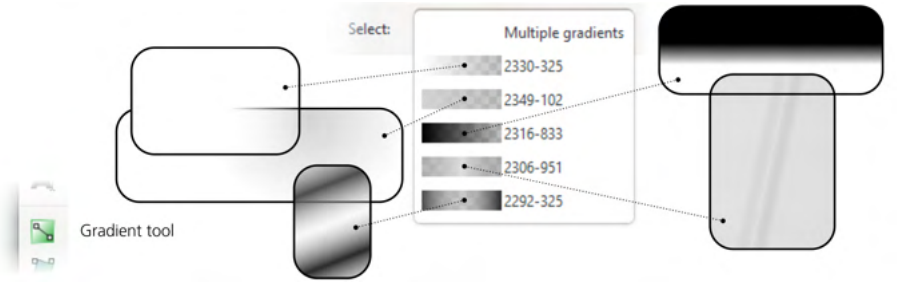


Figure 10-5: Choosing a gradient definition

You can *reverse* the color definition of the selected object's gradient by pressing Shift-R or clicking the Reverse direction button on the controls bar. For example, if you have an elliptic gradient with opaque blue in the center and transparent yellow at the edge, after pressing Shift-R you will have transparent yellow in the center and opaque blue at the edge. With a linear gradient, this is equivalent to rotating the gradient line by 180 degrees, but an elliptic gradient can't be reversed simply by moving its handles, which makes Shift-R indispensable.

10.2.1 Sharing Gradient Definitions

When copy-and-pasting or duplicating an object with a gradient, the object's copy automatically gets a *copy of the original gradient*, so that modifying it does not affect the source object's gradient definition. This behavior is controlled by the Prevent sharing of gradient definitions checkbox on the Tools ▸ Gradient page of the Preferences dialog. It is checked by default; if you uncheck it, copy-and-pasting, duplicating, pasting style, and explicit assignment of an existing gradient definition to an object will result in a *shared* gradient definition. When two objects share a gradient definition, changing the colors/opacities or middle stop positions of the gradient on one object affects *all* the objects that use the same definition. However, moving the end handles always affects only a single gradient because these nodes' positions are not part of a gradient definition.

Another setting that affects sharing of gradient definitions during object duplication is the lock button on the Gradient tool's controls bar. If it is pressed, the next duplication of an object with a gradient will result in a shared gradient, so that editing a gradient in one of the objects affects the other object as well. If you unpress the button, editing one of the shared gradients immediately breaks the link to the other (and pressing the button again will not restore it).

The Clean Up Document command in the File menu removes, among other things, any leftover unused gradient definitions that may be lingering in your document. It's a good way to tidy things up and reduce the document size (sometimes significantly).

10.3 Gradient Repeat

As you've seen, gradient handles don't need to coincide with the edges of the object they apply to; they can be positioned absolutely anywhere on the canvas, and the object will just display whatever part of the gradient happens to fall

within its limits. But what about those parts of the object that are not covered by the gradient—those that lie beyond the ends of a linear gradient or outside the edge of an elliptic one?

By default, these areas are painted by the flat color and transparency of the first or last gradient stop. For example, if you have a small elliptic gradient with semitransparent white at the radii, the rest of the object outside the gradient will be the same semitransparent white. However, this is not the only possibility.

Select an object with a gradient (but make sure none of the handles are selected) and in the **Repeat** list in the controls bar of the Gradient tool, select either **Reflected** or **Direct** instead of the default **None**. These options force the gradient to repeat itself indefinitely, either unchanged (direct) or with an inversion of every second copy (reflected). This is an easy way to create various striped patterns on objects, as shown in Figure 10-6.

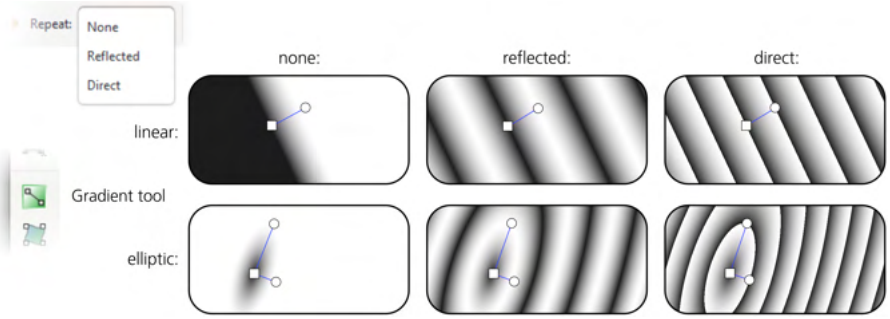


Figure 10-6: Using the gradient Repeat option

10.4 Gradient Handles

Handles are the on-canvas controls that correspond to the gradient stops. In the Gradient tool, they can be not only dragged around but also *selected* and *painted*, much the same as selected objects are painted.

10.4.1 Selecting

Selected handles are blue; unselected ones are white. The simplest way to select a handle is by clicking it; you can also Shift-click to add a handle to selection or remove it from selection, or Shift-drag around multiple nodes to select them with a rubber band (compare 5.7). After you have created a new gradient by dragging, the handle you've been dragging remains selected.

Similar to the Selector, in the Gradient tool you can also select handles with the keyboard. Tab and Shift-Tab select the *next* and *previous* handle (or first and last, if none were selected before); Ctrl-A selects *all* handles (in all selected objects); and Escape deselects any handles (but leaves the objects selected; a second Escape deselects objects).

NOTE *Watch the status bar (2.6) for useful information on the selected gradient handle(s) as well as the object(s) and gradient types they belong to.*

10.4.2 Painting Gradient Stops

You can assign any color or opacity level to the selected gradient handle or handles using all the same methods you would use to change the style of an object (8.1). When a handle is selected, the palette, the Fill and Stroke dialog, the color gestures and commands in the selected style indicator (8.2), the Paste Style command, and even the Dropper tool all work on the selected handle, not the selected object. Figure 10-7 shows an example.

Figure 10-7: Assigning color to selected gradient handles

Unlike an object, a gradient stop does not have fill or stroke (even though the gradient itself can apply to fill or stroke). Therefore, when you have one or more stops selected, the style indicator in the status bar shows the color of the stop in *both* the fill and stroke swatches; this may not be perfectly logical, but it is handy. The opacity of a gradient stop is indicated by the master opacity control (labeled O: in the status bar). The displayed stroke *width*, however, is always that of the object (if it has stroke), not the stop, since a stop cannot have a stroke.

It is convenient to use the Dropper tool if you want one end of the gradient to blend smoothly into other objects; make sure the corresponding handle is selected, switch to Dropper, and click the area into which you want it to blend. Note that by switching to the Dropper tool, you leave the Gradient tool, but this is not a problem; like many other tools, Dropper displays gradient handles and even preserves handle selection. You cannot select objects in Dropper, but you can switch gradient handle selection.

When *multiple* gradient handles are selected, the selected style indicator displays the *averaged* color and opacity of the selected stops, marked with a (this is the same behavior as for multiple selected objects, 8.4.1).

When at least one handle is selected, the Copy command (Ctrl-C) copies to the clipboard the style (both color and opacity) of the single selected handle, or the averaged style of several selected handles. This means you can copy and paste style between gradient stops—select a handle, copy, select some other handle(s), and paste style (Shift-Ctrl-V). If you select several handles, this allows you to quickly average their colors and opacities by copying and then pasting their averaged style back onto them.

By the way, if you want to change the opacity of *all* stops in a gradient by the same amount, you don't need to tweak the opacity of each stop separately. Instead, simply adjust the master opacity of the object that is using the gradient (8.3).

10.4.3 Moving, Merging, and Snapping

You can *move* the selected gradient handles by directly dragging them with the mouse or by using the arrow keys with all the regular modifiers (Shift for 10 times the standard 2 px displacement, Alt for pixel-size displacement, Shift-Alt for 10 pixels displacement; see 6.5.1). Naturally, the end stop handles (gradient ends in a linear, center and radii in an elliptic) can be moved arbitrarily, whereas the middle handles can be moved only along the gradient line. (Don't confuse the *middle* handles with the *center* handle of an elliptic gradient; the latter, despite being in the center of an ellipse, represents an *end stop* of the gradient definition.)

If you select multiple objects, all of them that have gradients (linear or elliptic, on fill or on stroke) will display their handles, and you can edit any of them simultaneously. This opens up interesting possibilities. For example, you can select the ends of all linear gradients and move them all in parallel. Or you can press Ctrl-A to select all stops in all gradients and use arrow keys to move the entire ensemble of gradients as a whole.

What's more, any number of end stop handles (but not middle stops) can be *merged*. Just move one handle close enough to another, and it will snap to it and merge. (The status bar will report the merged status of such a handle—for example, *Gradient point shared by 2 gradients*.) When you drag such a merged handle, that affects all the gradients to which it belongs. To separate a merged handle, drag away from it with Shift.

For example, you can merge the centers of two objects' elliptic gradients, merge an elliptic handle with another object's linear handle, merge all three elliptic handles of multiple objects, or merge the gradients on fill and stroke of the same object, as shown in Figure 10-8.

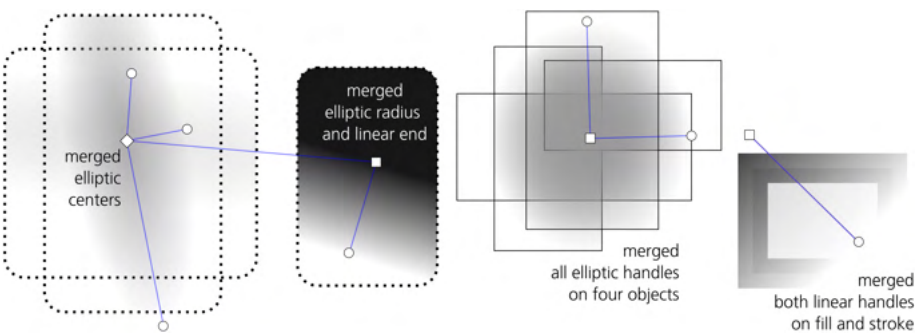


Figure 10-8: Merging gradient handles

When you drag to create a new gradient with several objects selected, you get what looks and acts like a single gradient applied to multiple objects—but it's actually many gradients with their handles merged together.

If two handles have different colors or opacities, they will remain different after you merge them. However, if you assign some color or opacity to a merged handle, it will apply to all the stops merged at this handle, equalizing their styles. If you want to change the style of one of the merged handles without unmerging it, this is possible if the merged handles belong to different objects: simply

deselect everything (both handles and objects) and select only the object whose handle you want to repaint.

When dragged, any handle of an object's gradient *snaps* obeying the snap bar toggles (7.3). In particular, the gradient handles as a snappable are controlled by the master toggle of the second family (Snap nodes, paths, and handles). The snap targets for gradient handles are controlled by the individual toggles in the first and second families; you can snap to edges, corners, centers, or midpoints of the bounding boxes, to paths and path intersections, and to cusp or smooth nodes on paths. Of course, you can also snap to grid or guides when they are present and enabled.

Also, when dragging a handle or when creating a gradient, holding Ctrl snaps the gradient angle to horizontal, vertical, and 15-degree increments in between (see Figure 10-9).

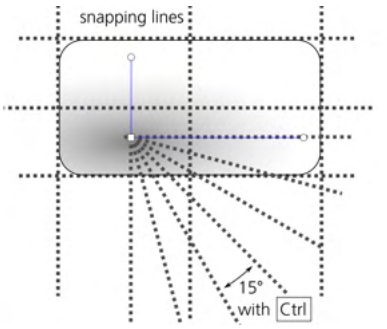


Figure 10-9: Snapping gradient handles

10.5 Multistage Gradients

A gradient that has at least one middle stop between the end stops is called *multistage* because it contains more than one color transition. In such a gradient, each middle stop has its own color and transparency, but its position is limited to somewhere in between the end stops. In Inkscape, a middle stop is represented by a diamond-shaped handle, as shown in Figure 7 in the color insert.

10.5.1 Creating Middle Stops

To add a middle stop in a gradient, double-click or Ctrl-Alt-click anywhere on the gradient line. The new stop will automatically get the color and transparency of the clicked point, so the appearance of the gradient does not change.

You can *drag and drop* a color from the palette onto the gradient line. Dropping a color on an existing handle changes the color of that stop; dropping it anywhere else on the gradient line *creates* a new stop with that color.

When two or more adjacent handles are selected, pressing Insert adds new stops in the middles of all selected stop intervals (much the same as it does for nodes in the Node tool, 12.5.3). The "plus" button on the Gradient tool's controls bar has the same effect. New handles added in this way are included in the

handle selection, so pressing Insert repeatedly adds more and more handles; if you start with two handles selected and press Insert *n* times, you will end up with 2^n handles.

To delete all selected stops, simply press Delete. You can also delete individual stops (selected or not) by Ctrl-Alt-clicking them. Or select a stop and click the "minus" button on the controls bar.

Deleting is not limited to middle stops; you can delete an end stop as well, so that the nearest intermediate stop becomes the new end stop of the gradient. When you delete an end handle of a linear gradient or a radii handle of an elliptic gradient, the behavior depends on how you did it; if you Ctrl-Alt-clicked the end node, the closest remaining handle moves up to its place so the gradient span stays the same, but if you selected the node and pressed Delete, the gradient shortens. If you delete the central handle of an elliptic, its nearest handle always moves in to become the new center. Finally, if you delete an end handle of a two-stop gradient, the gradient disappears and the object becomes painted with the flat color and opacity of the last remaining stop.

Pressing Ctrl-L with some intermediate stops selected *simplifies* the selected portion of the gradient, removing those stops that can be removed without too much change in the look of the gradient (compare 12.3). In particular, new stops created by double-clicking or pressing Insert initially do not change the appearance of the gradient, and simplifying will delete all redundant stops that weren't moved or repainted since creation. You may need to press Ctrl-L repeatedly to delete all unneeded stops.

10.5.2 Moving Middle Stops

Naturally, you can move a middle stop's handle along the gradient line (either by dragging or with arrow keys) no further than its neighboring handles. Dragging a middle handle with Ctrl snaps it to 1/10 fractions of the available range—that is, it will snap to 1/10, 2/10, 3/10, and so on of the span between its neighbors. The Offset field on the controls bar displays and lets you edit the position of the selected stop within the gradient (between 0 and 1).

Two or more middle stops may *coincide*. If they have different colors, the gradient in that point will have a sharp color boundary. For example, add two middle stops, paint one green and the other blue, and then drag the green one all the way to the blue one to create a sharp green-blue boundary in the gradient.

Dragging multiple selected handles with Alt moves them in a soft "rubbery" manner. The handle that you grab and drag moves all the way, but all other selected handles lag behind, the more so the farther they are from the handle you drag. The selected part of the gradient, instead of moving rigidly as a whole, smoothly reshapes as you drag (compare the node sculpting feature in the Node tool, 12.5.7.2).

Why is this useful? One gradient feature Inkscape (and SVG in general) lacks is *profiles*, which means any transition between two colors is always linear and cannot accelerate or decelerate (that is, it cannot shift more toward one of its end stops). However, Alt-dragging of middle stops makes it easy to *approximate*

such nonlinear profiles. If you have a two-stop gradient that you want to shape according to a curved profile, select both ends of the gradient, press Insert several times to add a number of intermediate handles, then Alt-drag a handle in the middle to smoothly reshape the gradient, as shown in Figure 10-10.

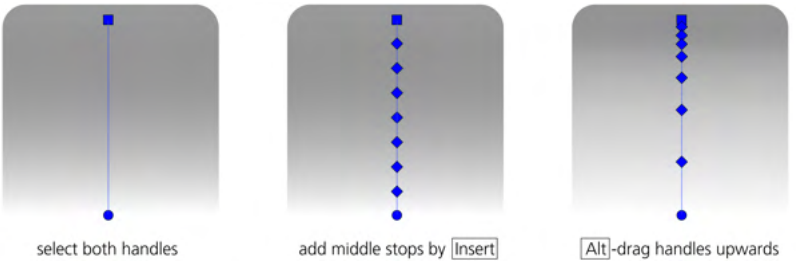


Figure 10-10: Approximating a profiled gradient by dragging middle stops with Alt

10.6 Gradient Tips and Examples

If you want to "fade out" or "feather" the edge of an object, you don't have to apply a color-to-transparent gradient to it (indeed, often you *can't*, for example, if it's a bitmap that can have no fill). You could use a mask (18.3); however, if the background under the object is a solid color, a collection of overlay objects (*shaders*) of the same color as the background with an opaque-to-transparent gradient may often be easier to create and maintain. This method works equally well for multiple objects that need to be feathered as a group, for different feathering on different sides, for objects that already have different gradient fills, or for bitmaps or pattern-filled objects.

In Figure 10-11, four gradient shaders are put over the edges of a bitmap to feather it out on a white background.

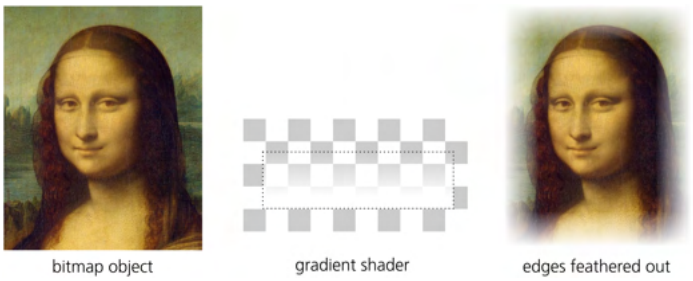


Figure 10-11: Feathering the edge of a rectangular object with linear gradient shaders

When using shaders, you don't need to move the gradient handles to put the gradient where you need it. Instead, just transform the entire shader object with that gradient. In Figure 10-12, the Gradient tool was used only twice: to create opaque-to-transparent elliptic gradients on two ellipses, one white and one black. Then, a total of 29 clones (linked copies) of these shader ellipses, variously scaled and rotated, all with various reduced master opacity levels, were used to add depth to the cartoon face.

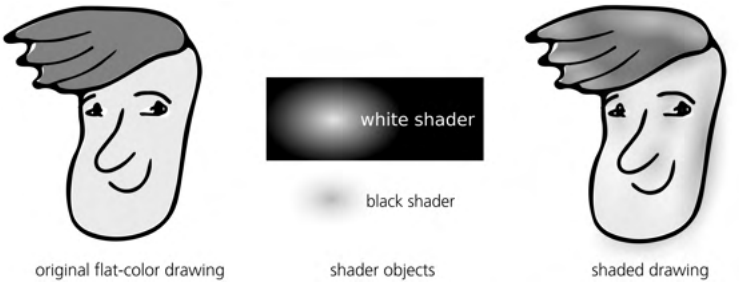


Figure 10-12: Adding depth to a cartoon with elliptic gradient shaders

One problem with black (or any other dark-colored) shaders is that they may not look soft enough—too blunt at the edges, especially when fully opaque. This is the result of the default gradient profile being linear. You can try to fix the problem by adding mid stops and "profiling" them by Alt-dragging (10.5.2). Another trick is to blur the shader a bit (17.1) so that its edges become less pronounced.

However, perhaps the simplest workaround for this problem is drawing a gradient not from opaque black to transparent black (which is the default), but from opaque black to transparent *white* (or, if you will be using the shader on some other light color, to the transparent version of that light color), as shown in Figure 10-13. This changes the perceived gradient profile drastically, making its edges a lot smoother and more natural. On the downside, this may make the center of the elliptic gradient a little too sharp, which you can often fix simply by scaling the shader object up somewhat.

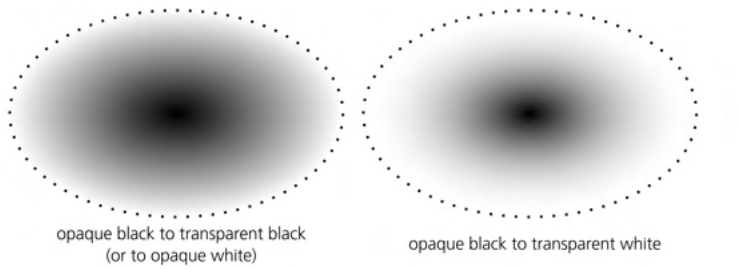


Figure 10-13: A better gradient profile for a dark shader by using a fully transparent light color at the edge

Using gradients in design often involves *overlaying* several objects with semitransparent gradients. Figure 8 in the color insert demonstrates a colored water drop (or a glass button), which is in fact a group of six objects with various elliptic gradients.

The Tweak tool's color modes (8.9) can paint or randomize colors not only in flat-color fills or strokes but in gradients, too. For gradients, the tool takes into account not only the position of the object with the gradient, but also the position of each gradient stop relative to the tool's brush.

For example, you can re-color only the blue end of an object with a blue-red gradient simply by painting over that blue end with a brush small enough not to touch the red. Color tweaking does not *create* gradients on objects that used

flat color before, nor does it add stops, but it paints the stops of the existing gradients in the drawing.

Let's give it a try. Take a simple opaque-black to transparent-black gradient, add a lot of middle stops by repeatedly pressing Insert, then randomize the lightness of the stops by stroking over the object with a Tweak brush in Jitter Colors mode, with only the L channel enabled. Then, repeat the same for two more copies of the smooth original gradient, and overlay them all on top of each other with a little rotation. The resulting texture is a quite believable misty seascape, as shown in Figure 10-14.

Figure 10-14: Randomizing a gradient with the Tweak tool

10.7 Mesh Gradients

[1.1] Similar to plain old-style gradients, *mesh gradients*, or simply *meshes*, also create smooth transitions between two or more colors. However, in a mesh, the points that carry these colors—analogous to gradient stops of a linear or elliptic gradient—are not positioned on the same straight line. Instead, the color *nodes* can be placed anywhere inside or outside the object. Also, the nodes are connected into a grid-like or conical *mesh*, whose segments are Bézier curves, similar to those used in paths (Chapter 12). The areas delineated by segments are called *mesh patches*.

Each mesh node has a color/opacity of its own; moving the nodes around, adding new nodes (and therefore multiplying segments and patches), as well as reshaping the connecting segments, affects how the color transitions between them stretch and bend. All this machinery allows you, with some skill, to create realistic approximations of various objects and effects quickly and efficiently. When tracing photos, you will even be able to do this, to an extent, automatically by picking the colors from under the meshed object (10.7.5).

MESHES IN ADOBE ILLUSTRATOR

Inkscape's Mesh Gradient tool works much like the Mesh tool in Adobe Illustrator. However, neither of the two applications can yet natively import each other's meshes. Inkscape imports meshes in AI and PDF files by approximating them with flat-color tiles with adjustable precision (B.5).

Mesh gradients are a relatively recent addition to Inkscape. I would be happy to say they are an addition to the SVG standard as well, but alas, this is not quite the case yet. Mesh gradients were in a draft of SVG 2.0 and extensively developed by the working group of the standard. Inkscape provided the first and by far the most advanced implementation of this new feature. However, browser vendors were reluctant to support it at all.

Understandably, Chrome or Firefox have rather more clout in the tech world than Inkscape; for SVG in particular, browser support is critical for survival. As a result, in order to speed up its adoption, the SVG working group decided to drop this feature from the upcoming SVG 2.0 standard. Meshes may be reinstated in a future version of SVG; meanwhile, Inkscape has no plans of discontinuing its support of mesh gradients even though this feature is currently not standard SVG.

10.7.1 When to Use Mesh Gradients?

A mesh gradient is a curious hybrid of bitmap and vector graphics. Like a bitmap, it is a grid of colored points—sometimes lots of them. Like a vector gradient, it allows you significant freedom in placing these points and smoothly interpolates colors between them. A perfect use case for mesh gradients is therefore something that is too organic, amorphous, or freeform for plain geometric gradients—but still smooth, without too much texture or fine details.

This class of mesh-friendly imagery includes all kinds of (mostly) smooth objects with soft, subtle color changes, curvilinear shades and highlights: clothes and drapery, faces and bodies, fruits and flowers, tree trunks and tree leaves, tableware, cars, ships, and sails. Also, as you'll see below, meshes are handy when you include a bitmap image in your drawing and need to retouch it to mask out some details.

While definitely more powerful than plain gradients, meshes in Inkscape are also much more complex and, as of yet, not as convenient to use. Many of the things that make working with regular gradients much easier (such as coloring with the Tweak tool or Alt-dragging multiple nodes) don't yet work in the Mesh Gradient tool. Hopefully the UI will keep improving in future versions.

Another thing to keep in mind is that mesh gradients, as of now, are an Inkscape-only feature not supported by other SVG software. If you plan to export your final artwork as bitmap or PDF, this shouldn't bother you much. If you plan to share your work in SVG format, your best bet is to convert the mesh to bitmap (18.6.4) inside the SVG document.

MESH GRADIENTS IN SVG

When saving an SVG file with mesh gradients, Inkscape automatically includes into it a polyfill script that ensures rendering of the mesh via JavaScript trickery when the SVG file is viewed in a web browser. In my testing, this polyfill works acceptably in Firefox and Chrome, but it does not work at all in Internet Explorer or Microsoft Edge.

10.7.2 Creating a Mesh

To create a mesh on a path (Chapter 12), a shape (Chapter 11), or even a text object (Chapter 15), switch to the Mesh Gradient tool via the toolbar button and double-click the object (Figure 10-15).

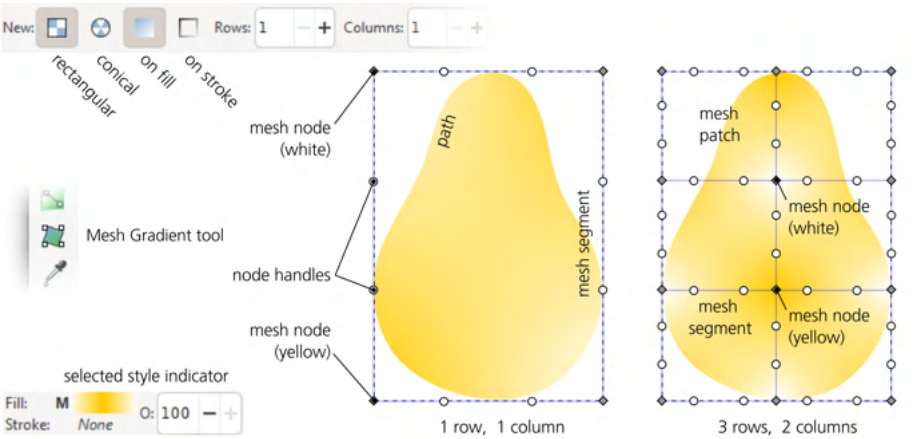


Figure 10-15: Creating a mesh gradient on a pear-shaped path

By default, the tool creates the simplest possible rectangular mesh with one patch and four nodes in the corners of the object's bounding box. You can change the initial number of patches using the Rows and Columns values on the controls bar (they affect only newly created meshes but not those that already exist). The nodes are initially colored, in checkerboard order, with the object's original color (for example, yellow) and opaque white.

Just as with linear or elliptic gradients, you can create a mesh gradient on an object's fill or stroke. This is controlled by a switch on the controls bar.

10.7.2.1 Rectangular or Conical?

Another switch on the controls bar offers a choice of a *rectangular* or *conical* mesh. A conical mesh, with a central point and radial segments, sounds like a great idea for (roughly) circular or otherwise centrally symmetric objects. In fact, when you create a mesh on a polygon or star (11.5), Inkscape will create a conical mesh, with the corresponding number of radii, regardless of the rectangular/conical switch on the controls bar.

However, a conical mesh is not a different mesh type; under the hood, it is still a rectangular mesh, only bent into a circle, with all nodes on one of the sides gathered into the same central point. One of the radii of a conical mesh is therefore a seam that can be opened up, as shown in Figure 10-16.

When editing a conical mesh, you will want to avoid disheveling the seam or the central point. That's not hard to do; just remember to select the nodes on the seam and the central node by dragging around them, not by clicking—so that you select *all* the overlapped nodes in that location (a simple click selects only one of the coinciding nodes).

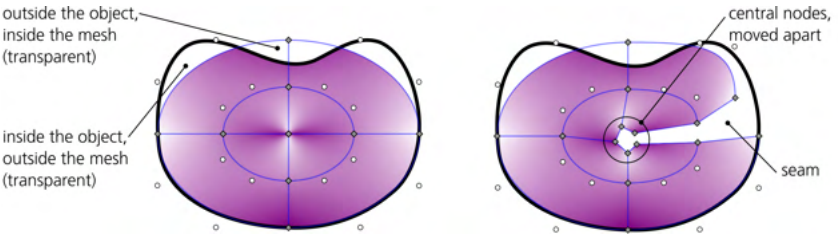


Figure 10-16: The anatomy of a conical mesh gradient

Note that in the initial conical mesh created by Inkscape, different nodes in the center have different (alternating) colors, which gives this area its characteristic conical appearance. If you want to edit the colors while keeping them different, you will have to move them apart, apply the colors, and then merge the nodes back. Otherwise, if you select all the central nodes by dragging around them and apply the color, you will get a plain symmetric color peak (similar to an elliptic gradient) instead of a cone (Figure 10-17).

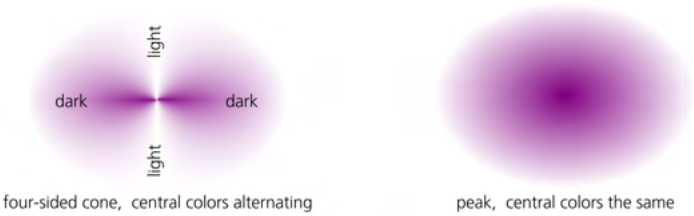


Figure 10-17: The center of a conical mesh: cone or peak

10.7.3 Shaping a Mesh

Mesh nodes are, in many ways, similar to path nodes (12.1). The biggest difference is that only the corner nodes of a mesh have two handles and two segments attached to them (like non-end nodes on a path); the edge nodes of a mesh have three handles/segments, and the inner nodes have four. This proliferation of controls makes editing complex meshes difficult. You can select and move multiple nodes, but you cannot elastically Alt-drag them (compare 12.5.7.2), which would be the most appropriate for the "mesh cloth," nor does the Tweak tool (12.6) work on mesh nodes.

In a mesh, you want the nodes—the carriers of the colors—to be positioned where you have your tinted spots, the peak highlights and shadows on your object. However, the shape of the segments connecting the nodes is no less important than the position of the nodes, because the segments form the "force field" that bends and stretches the tints. It is best to minimize mesh node editing by making sure that the new nodes are *created* more or less where you will need them and that the newly added segments *already* stick to the shape of the object you're modeling with your mesh.

When you are subdividing a mesh, new segments are created by *interpolating* between the adjacent existing segments. This gives us an optimal strategy: start

with a minimal four-node single-patch mesh (Figure 10-15, left), but before adding any more nodes, shape this outer frame to match the overall shape of the modeled object (in my case, a pear). Since the outside of a mesh is always fully transparent (see Figure 10-16, left), you need to place the outer nodes close enough to the object's path but on the outside, as shown in Figure 10-18.

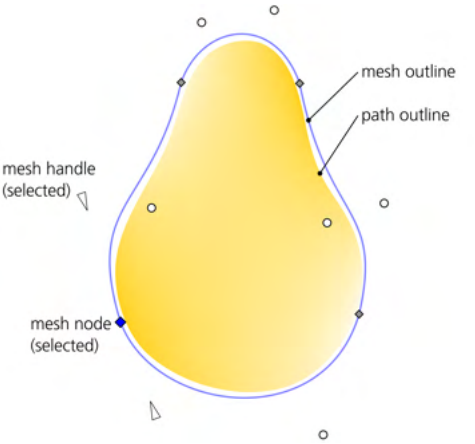


Figure 10-18: Shaping the outer edge of the mesh before adding any more nodes

Unlike paths (compare 12.5.5), there is no smooth node type for meshes, so you will need to drag the nodes and their handles manually to make them smooth and shape the segments. You can select multiple nodes and drag them at once; note that the handles of a selected node become oriented triangles pointing to their mother node (this makes disentangling a complex mesh a little easier, since there are no lines connecting a node and its handles). You can also switch a segment (after you select its end nodes) to a straight line and back to a curve using the two buttons on the controls bar.

Since reshaping even a simple mesh is such a hassle, you may be glad to learn than when you create a mesh on an ellipse (11.4), Inkscape automatically shapes it to match the elliptic shape (Figure 10-19, left; note that this is a rounded rectangular mesh, not a conical mesh). Thus, if your object is roughly circular or elliptic (such as a plate or a drop of liquid), it may make sense to create it as an ellipse first, make a mesh on it, and then, if necessary, convert it to a path and reshape it within its elliptical mesh. One problem with this approach is that the Node tool shows both path nodes and mesh nodes (that look quite similar) at the same time; to hide the mesh nodes, uncheck **Enable gradient editing** on the **Preferences** page for the Node tool (this affects both gradient and mesh nodes).

10.7.4 Subdividing a Mesh

Once you have shaped your object more or less as you want it to be *and* shaped the default single-patch mesh to cling to the object (Figure 10-18), you may start subdividing the mesh to add more points and patches. Now, the interpolated

segments and points will follow the mesh's outer shape—like meridians and parallels on a globe—so, in most cases, they will require only minimal editing. All that remains to do is assign colors to the mesh nodes to create the shading and tints you envision.

Subdividing the mesh is pretty easy. You can do it semi-automatically by selecting two or more nodes and pressing Insert or Shift-I: this adds a node (and corresponding segments) halfway between (each pair of) selected nodes. For example, if you select all four nodes (Ctrl-A) of a single-patch mesh and press Insert, you end up with a 2×2 patch mesh with nine nodes in total. Now, if you press Ctrl-A again (to add the newly created nodes to selection) and then Insert again, each segment will be bisected once more, producing 4×4 patches and 25 nodes (Figure 10-19).

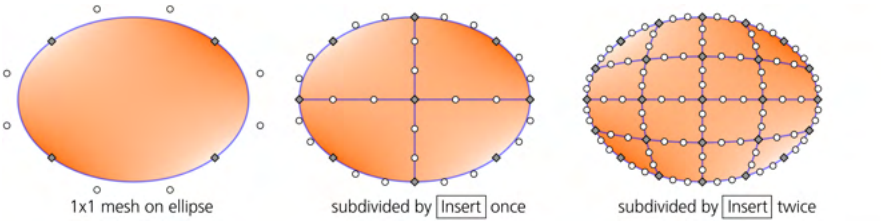


Figure 10-19: Subdividing a shaped mesh using the Insert key

Such indiscriminate doubling up of the mesh is a decent strategy if you don't have a good idea of the best locations for the nodes (for example, when creating a bitmap retouching patch, 10.7.5.1). If you do know where the peak highlights, shadows, and tinted spots are going to be, aim for them using manual subdivision. Double-clicking a segment in any point adds a node in that point and runs a new chain of segments through that point. For example, if you want a highlight in point A, first double-click the top segment in point B so you have a segment running through A, then double-click in A to create an intersection, as shown in Figure 10-20.

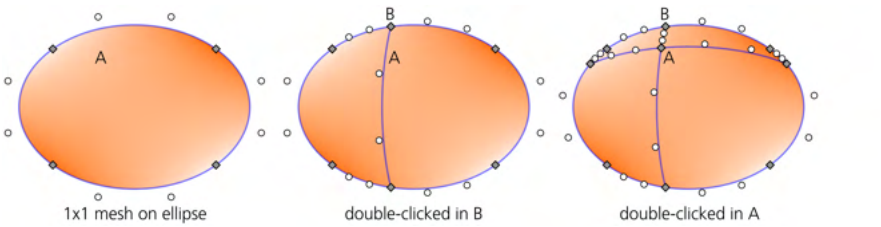


Figure 10-20: Subdividing a shaped mesh by manual creation of nodes

Note that adding nodes (with either method) does not change how the mesh looks: Inkscape interpolates not only positions and segment shapes but also colors—so that new nodes get exactly the color that the mesh had in this point anyway. It's now up to you to sculpt the color transitions by applying color to the new nodes.

NOTE *Currently there's no way to delete a mesh node once you add it (other than with Edit ▸ Undo).*

10.7.5 Coloring Mesh Nodes

Don't get overexcited with subdividing your mesh before you start actually coloring it; it is better to start minimally and add new nodes only when you see that you can't get the coloring right without it. Nothing beats seeing how a pretty minimal mesh can produce an amazingly lifelike interplay of colors and shades.

Before you get to coloring, however, I recommend that you switch the mesh for the default Coons interpolation to Bicubic on the controls bar. The Coons interpolation is linear, which makes the mesh look unnaturally angular; the Bicubic interpolation is much smoother.

Actual painting of the mesh nodes works the same as with gradient stops (10.4.2). Select one or mode nodes (click, Shift-click, rubberband drag) and assign a color from a palette, or adjust the existing color via the Fill and Stroke dialog (again, I recommend the Wheel tab as the most natural). You can also drag-and-drop color from the palette onto a node (selected or not).

NOTE *If you select several nodes, all you can do is assign the same color to all of them. There's currently no way to adjust them, for example, by making all selected nodes darker while keeping them different. The Tweak tool's coloring modes (8.9) do not yet work on mesh gradient nodes.*

From here on, working with a mesh is basically a repeated cycle of: select a node; adjust its color; deselect the node and the object (to hide the mesh, as it's hard to see how your colors look with the mesh sitting on top); select the object again; and repeat. Infrequently, you may also need to add a node by subdividing or move nodes by a little bit. Just invest enough time into this and you will be amazed by how lifelike your object emerges (Figure 9 in the color insert).

Several points are worth noting in Figure 10-20. Point A retains the original yellow color I assigned to all nodes at first; it is much easier to start with such an average color and then create slight deviations from it. Points B and C are the two main highlights; they are painted a very light (almost white) shade of the pear's yellow. Point D adds a green accent, and points E, F, and G contribute some redness. Naturally, the points on the right and bottom of the pear are painted darker to create the overall shading, but there are local exceptions to this; in particular, the edge nodes from E1 to E2 and from E3 to E4 (note that they lie outside the actual shape) are made pure white to create the "dissolving edge" effect as when the object is in front of a brightly lit background (such as a window).

10.7.5.1 Patching a Bitmap with a Mesh

Meshes are an easy way to create a photo retouching patch—an object that perfectly blends in with a bitmap but covers something on the bitmap image that you don't need. Let's suppose you want to mask out the craquelure on one of Mona Lisa's cheeks. Start by drawing a rounded object over an area. Create a rectangular mesh on it and subdivide it a few times with Insert (10.7.4). For best results, the mesh's outer nodes should be outside the object's area; for this, just select all the nodes of the path (not the mesh) in the Node tool and scale the path down by pressing < a few times (12.5.7.3), as shown in Figure 10 in the color insert.

Now, all you need to do is select all mesh nodes in the Mesh Gradient tool and click the Pick colors button on the controls bar. For each selected node, Inkscape looks up the background (that is, bitmap) color at this point and assigns that color to the node. The patch is still visible in close-up because its smoothness contrasts with the craquelure texture of the rest of the face—but if you zoom out a little, it blends in perfectly! While it may be possible to achieve a similar effect with other tools (plain gradients, Dropper tool, blur), only mesh gradients make retouching so delightfully easy.

10.8 Patterns

A *pattern* is a paint type (8.2) where an object's fill or stroke consists of repeated copies of a *tile*. The tile can be anything: a single object or a group of objects, using any style properties or Inkscape techniques (for example, a tile may itself be recursively painted with a pattern). This universality makes patterns a very rich and flexible feature. In another respect, however, patterns in SVG are limited: they can tile only in a simple rectangular grid with no rotations or reflections (compare the clone tiler, 16.6, which can arrange clones using 17 different planar symmetry types).

You cannot have both a gradient and a pattern on an object; they are two alternative paint types. If you want a transparency gradient on an object with a pattern, read 10.6 on shaders or 18.3 on masks.

10.8.1 Creating Patterns

Unlike gradients or meshes, patterns don't have a dedicated creation tool in Inkscape. Instead, just select the object or objects that you want to turn into a tile and choose **Object ▸ Pattern ▸ Objects to Pattern** from the menu, or press Alt-I.

The selected objects do not disappear and don't visibly change; however, you'll notice that instead of the original object(s), you now have a single *rectangle* selected. This rectangle is filled with the pattern made out of your object(s), and it has exactly the size and position of their bounding box, which means that exactly one copy of the pattern fits into the rectangle. If you now drag the rectangle sizing handle (using the Rectangle or Node tool), you will see the other tiles as well, as Figure 10-21 demonstrates.

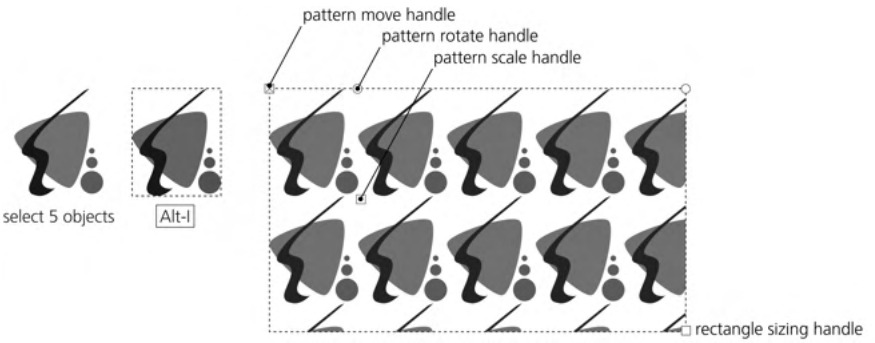


Figure 10-21: Creating a pattern from objects

Now you can easily assign the new pattern to any object by choosing Copy and Paste Style from this rectangle (which you can then delete if no longer needed), or by choosing it from the pattern list in the Fill and Stroke dialog, as shown in Figure 10-22.

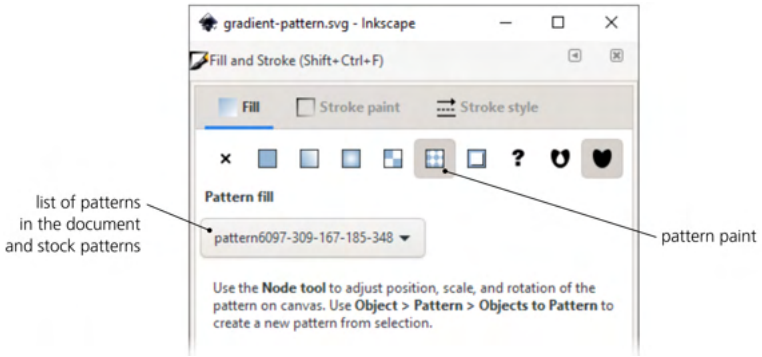


Figure 10-22: The pattern paint in the Fill and Stroke dialog

SVG allows you to set margins around tiles in a pattern to space them apart. Inkscape does not yet support doing this via the UI, but a simpler approach is to add to the original objects, before you turn them into a pattern, a transparent rectangle extending beyond the edges of the other objects and thus adding separation between the pattern's tiles. It is not possible to make pattern tiles overlap.

If you want to extract the tile objects back from a pattern in order to edit them, select an object with that pattern and use the **Object ▸ Pattern ▸ Pattern to Objects** command (but make a copy of the patterned object first if you don't want to lose it).

10.8.2 Editing Patterns

When you are in the Node tool or a shape tool, each object with a pattern fill displays three editing handles that allow you to move, scale, and rotate the pattern within the object (Figure 10-21). As with gradients, these handles can be anywhere on the canvas, not necessarily on the object itself. Clicking a scale or rotate handle (but, for some reason, not the move handle) selects it, and you can then move it with the arrow keys.

X-shaped handle
: Positioned in the top-left corner of one of the tiles (the "origin" tile). Dragging this handle *moves* the pattern in any direction.

Square handle
: Positioned in the bottom-right corner of the origin tile. Dragging this handle *scales* the pattern. To make the scaling uniform (that is, to preserve the width/height ratio of the tiles), drag with Ctrl.

Round handle

Positioned in the top-right corner of the origin tile. Dragging this handle *rotates* the pattern around the X-shaped handle as an axis. Drag with Ctrl to snap the rotation angle to 15-degree steps.

10.8.3 Patterns in Art

Aesthetically, the problem with patterns is the same as their main advantage: *repetitiveness*. Patterns may be necessary or at least acceptable in technical illustrations, but artistic drawings are rarely improved by an annoyingly repetitive rectangular pattern.

If you use a pattern in your artwork but don't want it to look too oppressively regular, you may want to scale the pattern up (so that fewer copies of it fit into the patterned object) and/or rotate it away from horizontal/vertical. A more radical approach is to tile not a single object but a random scattering of several copies (or clones) of it; this way, it may take the viewer some effort to notice any regularity at all (that's how the stock polka dot patterns were made, 10.8.4).

10.8.4 Stock Patterns

The Objects to Pattern command is not the only way to add patterns to your document. Inkscape comes with a selection of simple *stock patterns* that you can reuse in your documents.

Select an object and switch it to pattern paint by clicking the **Pattern** button on the corresponding tab of the **Fill and Stroke** dialog. You will see a drop-down list (Figure 10-22) that contains your document's custom patterns, if any, at the top, and then a set of stock Inkscape patterns after a separator. To use any of the patterns, simply select it from the list.

- There is a selection of plain Stripes with different ratios of stripe width to gap width, in the range from 4:1 to 1:64. For example, the Stripes 1:2 pattern has gaps twice as wide as the stripes. All stripes patterns exist in two versions: with black stripes and with white stripes (gaps are always transparent).
- There are two Checkerboard patterns with black and white odd squares (even squares are transparent).
- The Packed Circles option is a dense hexagonal pattern of black circles with transparent gaps.
- Polka Dots is a scattering of dots randomly but evenly distributed to mask the regularity of the repeating pattern. This pattern has three size variants (small, medium, and large dots) and two color variants (black and white dots).
- Wavy is a pattern of wavy lines with transparent gaps between them.
- Camouflage is a green-toned protective pattern such as that used by the military.
- Ermine is the traditional heraldic pattern, originating from a stylized representation of stoat furs with black tails.

- The three bitmap patterns, Sand, Cloth, and Old Paint, are based on seamless photographic tiles and allow you to add some natural texture to your drawing. All of them are grayscale, so you can make objects with these textures semi-transparent and overlay them on other colored objects to texturize them.

Stock patterns are stored in the file *inkscape/paint/patterns.svg* in Inkscape's data folder that you can look up in Preferences, System page. You can add your own patterns to this file or replace it with any other SVG file containing the stock patterns you need.

[1.1] *10.8.5 Hatches*

Hatches are an alternative mechanism that SVG offers for painting an object with repeated copies of something. Hatches are more limited than patterns in that they can't use *any* object as a tile but only a path fragment—which is used to, well, hatch the object that refers to this hatch as fill or stroke paint.

Hatches are a new feature in SVG 2.0 (still not finalized, as of this writing) and are not universally supported in browsers. Inkscape can render SVG hatches, but it doesn't offer on-canvas handles and other conveniences for editing them. It's not a feature I would recommend anyone use; I am mentioning hatches only because Inkscape comes with a number of stock hatches you can browse and apply from the Paint Servers dialog (10.9).

10.9 The Paint Servers Dialog

[1.1] Remember that paint (8.2) is anything that is used to draw an object's fill or stroke. Of all the paint types, some are simple (such as no paint or flat color paint) and some are complex (such as a gradient or pattern). In SVG, complex paints are defined not in the objects that use them but separately—which is why you can, for example, share gradient definitions between objects (10.2). Such complex paints are called *paint servers* because they provide their paint to any number of users (that is, objects).

Inkscape has a Paint Servers dialog (from the Object menu) that lets you review the paint servers and apply them to a selected object's fill or stroke. It's not something you couldn't do with other tools and dialogs; the primary value of this dialog is that it shows each paint as a thumbnail, making comparisons easier. You can view the paint servers (gradients, mesh gradients, patterns, and hatches) defined in the Current document, or you can review the stock Patterns and Hatches supplied with the program.

11

SHAPES

The freedom to do anything to anything, any time, is what makes vector graphics so attractive. With a small vocabulary of generic object types and tools for manipulating them, you can render, or at least approximate, any kind of graphic imaginable.

Absolute freedom is not always a good thing, however. For example, a path (Chapter 12) can represent any possible two-dimensional shape. Yet often, what you need is not "any shape" but a simple, well-defined geometric entity, such as a rectangle. Of course, a four-node path will give you a perfect rectangle—but isn't there a faster and more convenient way to create, specifically, rectangles?

Indeed, Inkscape has several object types for commonly used geometric shapes: rectangles, 3D boxes, ellipses, polygons, stars, and spirals. Each shape type has a corresponding creation tool, and Inkscape provides an array of numeric parameters, draggable handles, and shortcuts for manipulating those shapes. You cannot do *everything* to such a shape object, but what you *can* do to it makes perfect sense for its specific shape type.

INKSCAPE'S SHAPES VS. AI'S LIVE SHAPES

Shape tools in Inkscape are similar to the "live shapes" tools in the latest versions of Adobe Illustrator. It's interesting that AI didn't have this capability until 2015—a decade or more after Inkscape and most other vector editors had it!

You can always press Shift-Ctrl-C to convert a shape to a path (or, in the case of a 3D box, to a group). The reverse conversion, however, is not possible (at least not automatically). This means that a shape is a *higher level of abstraction* than a path: converting a shape to a path causes information loss and is therefore a one-way, destructive operation.

SHAPES IN SVG

SVG has its own shape elements for rectangles, ellipses, circles, and polygons. However, Inkscape's shapes are richer than what is defined in the SVG standard. As a result, Inkscape uses, as a fallback, the generic `path` element with some Inkscape-specific attributes whenever it cannot express the required shape using pure SVG. Details depend on the specific shape. Inkscape's rectangles are always represented by the `rect` SVG element; ellipses and circles start as `ellipse` and `circle` but are automatically switched to `path` when you turn them into arcs or segments (11.4.2); and stars, polygons, and spirals always use `path`.

11.1 Shape Tools

To create a new shape object, drag on the canvas with the corresponding tool—for example, dragging with the Rectangle tool creates a new rectangle. The newly created shape remains selected, and any selected shape displays its editing *handles* (similar in appearance to the gradient handles, 10.1). By dragging those handles, you can immediately edit what you have created.

Most handles work differently when you drag them with or without various keyboard modifiers (Ctrl, Shift, Alt). Hover your mouse over a handle to read a tip, in the status bar, on what this handle will do when dragged or clicked with different modifiers.

Like most tools, shape tools have certain object selection capabilities. In any shape tool, you can select an object (not necessarily a shape) by clicking—which works like Ctrl-clicking in the Selector tool (that is, ignores any grouping, 5.10). Alt-click (select under, 5.9) works but Alt-wheel does not; Shift-click (add to selection or remove from selection) works, but the rubberband does not (because dragging creates a new shape). Escape deselects.

To *create* a rectangle, you need to use the Rectangle tool—but *editing* a shape is another matter. *Any* shape object, when selected, lets you drag its shape-specific handles in *any* shape tool, as well as in the Node tool (F2). When you have *multiple* shapes selected, however, only the Node tool lets you edit any of their handles.

11.1.1 Shape Parameters

Each shape tool has its own controls bar (2.3) where you can view and edit parameters. Usually, there are a few numeric entry fields and toggle buttons, as well as a button to reset the values to the defaults (located to the right of all the other controls).

The leftmost label on the controls bar says either New: or Change: before all other controls. "New" means that no shape of the corresponding type is selected, so the parameters you're editing will be used for a new shape when you create it. "Change" indicates that the selection contains shapes of the current tool's type—to which your parameter change will therefore apply.

Any changes made to these controls are remembered and used for the next object you draw with that tool—just as the style properties you assign to an object are typically used for the next created object (11.1.2). For example, after you change the number of corners of a star, any new stars you draw will have the same number of corners.

Moreover, even simply selecting a shape sends its parameters to the controls bar, where those parameters are remembered and later used for newly created shapes of that type. This makes it easy to reuse shape parameters—similar to pasting a style from object to object (8.1) but without using copy and paste. For example, if you have a slightly rounded star with nine corners and no randomization somewhere in your drawing, simply select that star in the Star tool and subsequent stars will be created with exactly the same parameters.

11.1.2 The Style of New Shapes

What style is going to be used for the next shape you create? You can find the answer to this question by looking at the far right end of the controls bar. All shape tools—in fact, all object-creating tools—have a style swatch there displaying the fill, stroke, and opacity that will be used for the newly created objects (Figure 11-1).

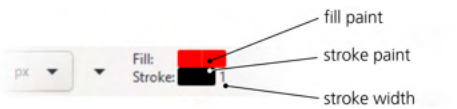


Figure 11-1: This style will be used for newly created shapes.

By default, all shape tools except Spiral use the *last set style* as the style of new objects they create. This means that every time you change some style property in an object, Inkscape remembers that property and will use it for new shapes. For example, after you paint something green, all new rectangles, ellipses, stars, and so on will have the same shade of green. Or if you assign a 3 px stroke width with a 1:1 dash pattern to some path or shape, new shapes will also have that stroke style. (This behavior becomes more complex for 3D boxes, which use not one but six different styles for the six sides of a box; see 11.3.5 for details.)

If you don't want the most recently set style to be used for new objects, you can change that. Double-click a shape tool's button in the toolbox on the left

to open that tool's page in the Preferences dialog (3.1.1), as shown in Figure 11-2. There, use the two-way switch to set the style of new objects either to the *last used style* or a fixed *tool style* specific to this tool. To change the tool style, click the **Take from selection** button to remember the style of the current selection as the tool style.

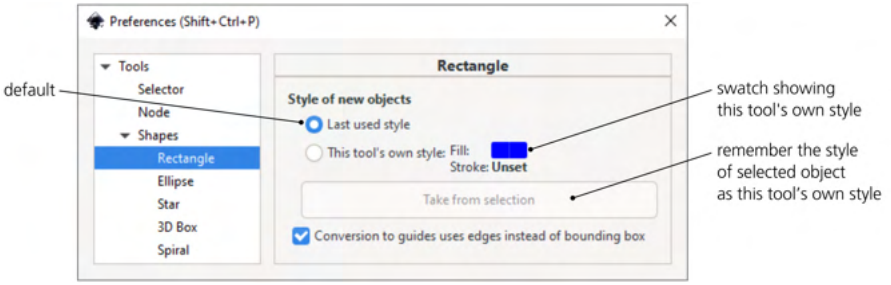


Figure 11-2: A shape tool's preferences page

For example, if you want all new rectangles to be black without stroke, select some black strokeless object, go to **Preferences ▸ Tools ▸ Shapes ▸ Rectangle**, switch to **This tool's own style**, and click **Take from selection**.

11.2 Rectangles

Rectangles, boring as they may sound, are the most commonly used type of shape. You'll be hard-pressed to find a design not dominated by rectangles. Inkscape makes creating and editing rectangles as easy and versatile as possible.

Switch to the Rectangle tool (by clicking the toolbar button on the left or by pressing R or F4) and drag anywhere on the canvas, as shown in Figure 11-3. Drag with Ctrl to get a square or an integer-ratio (2:1, 3:1, and so on) rectangle; drag with Shift to make the starting point the rectangle's center instead of one of the corners. You can combine Ctrl and Shift.

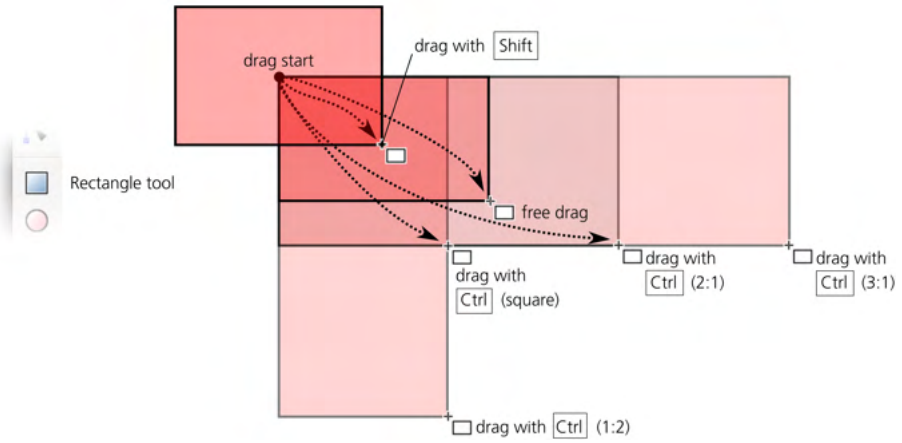


Figure 11-3: Drawing rectangles

The new rectangle displays four handles, as shown in Figure 11-4. Two are little squares in the top-left and bottom-right corners; these are the *sizing handles*. The two others, which appear as little circles, are *rounding handles*; they are both in the top-right corner and look like one handle until you drag one of them away.

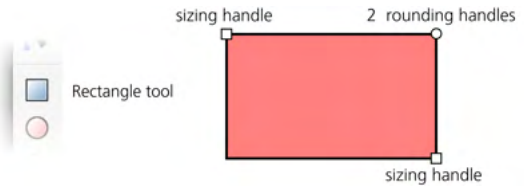


Figure 11-4: Rectangle handles

11.2.1 Sizing

Using the sizing handles, you can resize the rectangle simply by *dragging* any of the sides in any direction. Dragging with Ctrl locks the width, height, or width/height ratio by snapping the handle to the rectangle's sides or to the diagonal.

NOTE *Unlike the Selector tool, you cannot move, for example, the bottom-right sizing handle further up or further to the left than the top-left handle. The most you can do is make the rectangle invisible by reducing its width, its height, or both to zero.*

In the controls bar, two numeric controls labeled W and H also control the width and height of the selected rectangle. They use the measurement unit chosen in the unit selector on the right.

Why use the sizing handles when you can just as easily resize the rectangle with the Selector tool? The Selector tool always scales things horizontally or vertically in the document coordinate system—that is, along the edges of the page. In contrast, a rectangle's sizing handles scale it *along the sides* of that rectangle, even if the rectangle was rotated or skewed. The W and H values also always reflect a rectangle's intrinsic width and height, not necessarily the dimensions of its bounding box, which may be quite different if the rectangle was rotated or skewed (Figure 11-5).

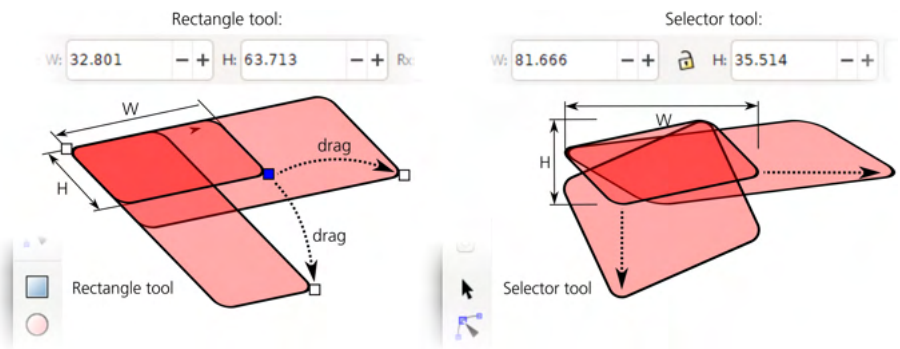


Figure 11-5: Rectangle sizing handles vs. scaling with the Selector tool

Another advantage of the sizing handles is that they always preserve the radii of the rectangle's rounded corners (although, as you'll see shortly, this is possible with the Selector as well).

As with any other shape type, you can make handles of rectangles snap (7.3) to grids, guides, and other objects. If you start from a particularly rotated and/or skewed rectangle and enable snapping of nodes to paths, nodes, and intersections, it's easy to use duplication (Ctrl-D) and sizing handles to create snugly fitting, gapless compositions of axonometric rectangles, as shown in Figure 11-6.

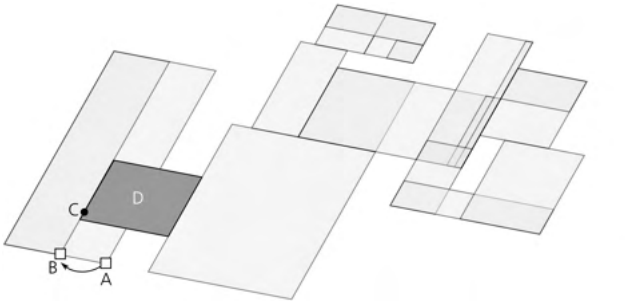


Figure 11-6: Snapping slanted rectangles to each other

What snaps is the *mouse point*, not necessarily the handle itself. These two may diverge quite far if, for example, you are resizing a rectangle with Ctrl to lock its width or height. In Figure 11-6, I wanted to make the leftmost rectangle narrower but keep its height, so I moved the handle with Ctrl from A to B. However, I also wanted this rectangle to abut the left edge (marked by a bold line) of rectangle D. So, while dragging and without letting go of Ctrl, I moved the mouse cursor to point C so it snapped to the edge I needed (the actual handle was then at B), and there I released the mouse. To make this possible, I enabled nodes and handles as snappables and paths as snap targets (7.3).

11.2.2 Rounding

To round corners of a rectangle, grab one of the circular rounding handles and drag it along the side of the rectangle. All four corners of the rectangle become rounded by circular arcs. Also, now you can see the second rounding handle—it remains in its original position in the corner (Figure 11-7). If *circular* rounded corners are what you need, you can leave it at that. If you want *elliptic* corners instead, move that other handle away from the corner along the other side of the rectangle.

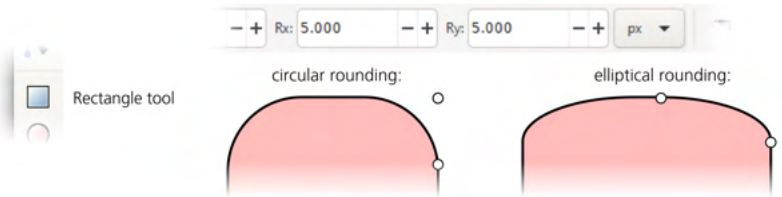


Figure 11-7: Rectangle rounding handles

NOTE *Even circular rounding can look elliptic if the rectangle is skewed (compare 24.3).*

Using the Rx and Ry numeric fields in the controls bar, you can explicitly set both rounding radii in absolute units (chosen with the unit selector to the right). If multiple rectangles are selected, the value you type will apply to all of them (if any non-rectangles are selected, they will be ignored). The button with a corner icon on the right removes any rounding from the selected rectangles.

The maximum distance you can move the rounding handles is half the length of the corresponding rectangle size. Reaching this maximum with both rounding handles effectively turns a square into a circle and a non-square into an ellipse.

In technical drawings such as schemes and diagrams, the size and shape of rounded corners often needs to be the same in the entire composition, even if the sizes of the rectangles are different. Inkscape makes this easy. The second of the four Affect buttons on the Selector toolbar (6.11), the one with two concentric rounded corners, controls whether the rounded corners are scaled when a rectangle is scaled. Figure 11-8 shows a bunch of rounded rectangles scaled with this button on and off.

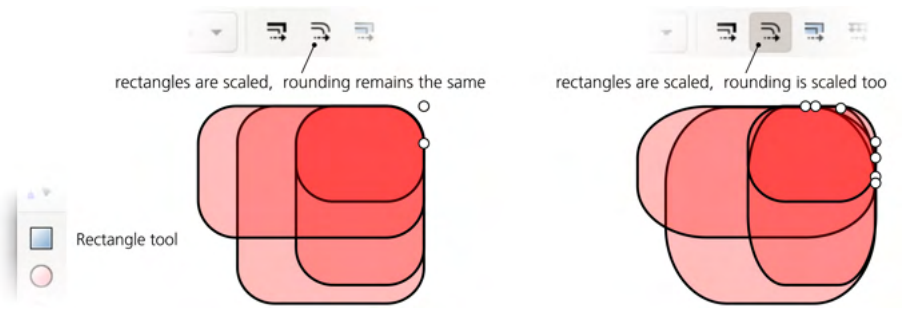


Figure 11-8: Scaling a rectangle may or may not affect its rounding radii.

Here are the shortcuts for the rounding handles of a rectangle:

- Drag with Ctrl to make the rounding circular (that is, make the other radius the same).
- Ctrl-click a handle to make the rounding circular without dragging.
- Shift-click a handle to remove rounding.

11.3 3D Boxes

A *3D box* is an object representing a projection of a three-dimensional box (rectangular prism) onto the plane of the drawing. It therefore consists of six *sides*, and from a pure SVG viewpoint, it is simply a group of six paths, each with four nodes and four linear segments. Inkscape, however, treats these objects in a special way—so you can, for example, resize and move them in their own 3D space or reposition the vanishing points of multiple boxes at once.

Apart from the 3D Box tool, most other tools and commands treat a 3D box as a group. In particular, in the Selector tool, you can Ctrl-click, Ctrl-Alt-click, or Alt-drag to select any side within the box (most often for changing its style), just as you would select objects inside a group. Selecting sides inside a box does not destroy the box; however, you can easily "unbox" it with the Path ▸ Object to Path (Shift-Ctrl-C) command—this removes any 3D-specific capabilities and leaves you with a regular group with paths inside. You can also simply Ungroup (Ctrl-U) a 3D box.

11.3.1 Why Use 3D Boxes?

What makes such a specialized construct as a 3D box useful?

Inkscape is not going to replace a full-featured 3D application—it does not model a "true" three-dimensional space in which you could place your 3D objects. Inkscape is and always will be a two-dimensional drawing tool. However, it is often used to draw three-dimensional objects. To assist in this, Inkscape implements a simple, 2D-oriented system of *perspective drawing*—whose principles have remained unchanged since being perfected by the Renaissance artists six centuries ago.

In Inkscape, you don't "build a 3D world"; you just create a flat drawing that represents a three-dimensional scene in a certain perspective. Inkscape's 3D Box tool is mostly a helper that makes creating such perspective drawings easier. In perspective drawing, a box is almost as important as a rectangle is for two-dimensional drawing and layout; drawing anything in correct perspective usually starts with drawing its enclosing box. So Inkscape's 3D boxes are often used not for their own sake but as quick and 3D-accurate guides for your perspective drawing—something to align your objects against and to inscribe them into (see Chapter 22 for an example).

On the other hand, the ease of creating and reshaping 3D boxes in Inkscape is an inspiration in itself. The About screen (Help ▸ About) of Inkscape 0.46 (in which this tool made its debut) is an example of a piece of art dominated by 3D boxes shown in Figure 11-9.

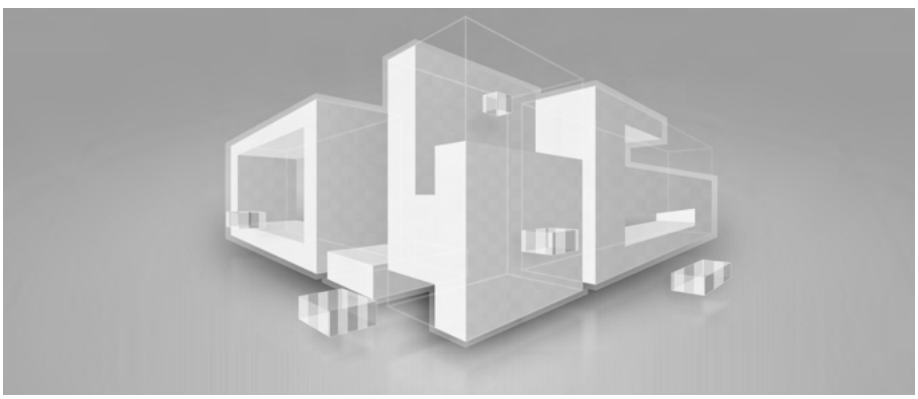


Figure 11-9: A composition of 3D boxes

11.3.2 Drawing

To draw a 3D box, switch to the 3D Box tool (the X key or Shift-F4) and drag somewhere within the page frame (because the default vanishing points are on the left and right edges of the page). This creates the front side of the box in the X/Y plane; the depth of the box along the Z axis remains fixed. To switch to the Z axis, drag with Shift; this freezes the X/Y side but lets you adjust the depth, as shown in Figure 11-10.

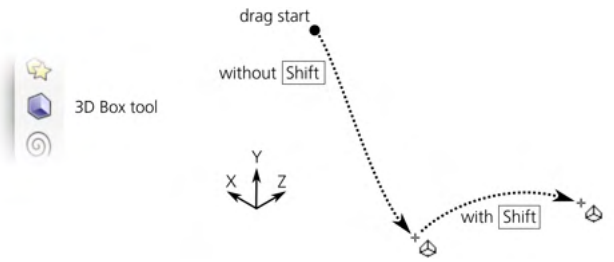


Figure 11-10: Drawing a 3D box

Once a box is created, it displays a diamond-shaped handle in each of the eight corners as well as an X-shaped handle in the center. Also, colored lines go from the edges of the box to the vanishing points. Let's look at these controls in detail.

11.3.3 Perspective and Vanishing Points

Each 3D box exists in a certain *perspective*, which is defined by three *vanishing points*, corresponding to the three spatial dimensions: X (red guides), Y (blue guides), and Z (yellow guides). Each vanishing point may be either *finite* (it is an actual handle that you can drag around) or *infinite* (it is just a direction toward a point at infinity; you can change the angle of this direction).

By default, the X and Z vanishing points are finite, located in the middle of the left and right edges of your document's page. The Y vanishing point is infinite, its direction being vertical, as shown in Figure 11-11.

You can toggle the finite/infinite status of each dimension's vanishing point using the three buttons in the tool's control bar. The finite vanishing points can be dragged freely on the canvas. The angles of the infinite ones can be adjusted numerically in the controls bar or by the shortcuts:

- [and] rotate X vanishing point directions.
- (and) rotate Y vanishing point directions.
- { and } rotate Z vanishing point directions.

Without modifiers, these keys rotate by the angle step (the default is 15 degrees, 6.3). With Alt, they rotate so that the perspective lines are moved by (at most) 1 screen pixel at the current zoom.

Shift-X, Shift-Y, and Shift-Z toggle the corresponding vanishing points from finite to infinite and back.

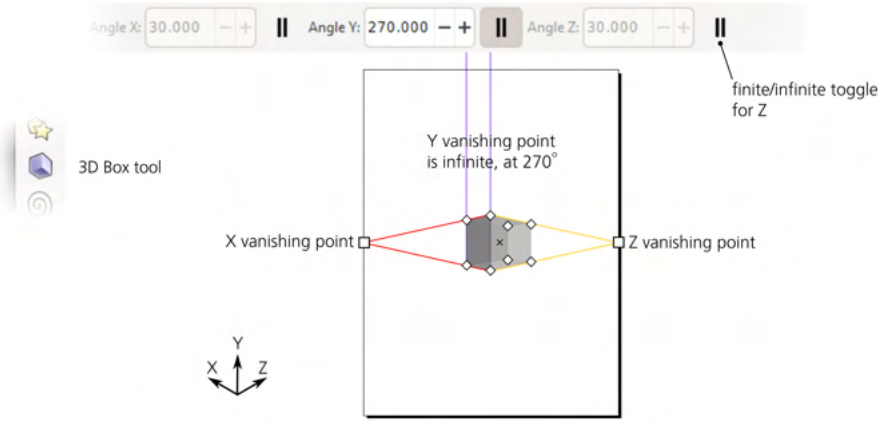


Figure 11-11: The default perspective

For example, you can make all three vanishing points infinite and rotate them to the angles of 150 (X), 90 (Y), and 30 (Z) degrees for drawing isometric boxes without any perspective foreshortening. Also, while boxes look most natural when they are located somewhere in between the three vanishing points, they are not obliged to be there; moving a box away from the sweet spot of a "natural" perspective makes it look curiously distorted—which, sometimes, may be exactly what you want.

11.3.3.1 Merging and Unmerging Perspectives

For a newly created 3D box, Inkscape reuses the perspective of the *last selected* 3D box. As a result, boxes in your drawing will normally *share* the same perspective. When you change a perspective, *all* boxes using this perspective—whether selected or not—respond to the change, as shown in Figure 11-12.

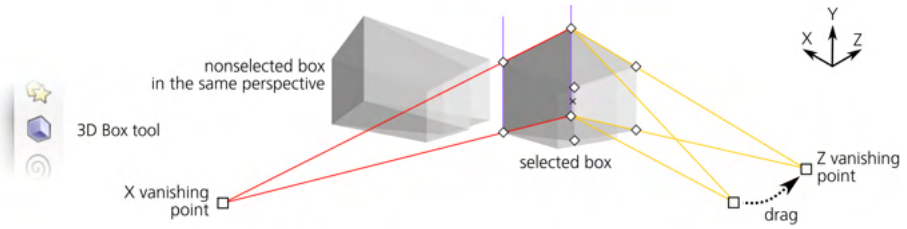


Figure 11-12: Changing a perspective by dragging a vanishing point affects all the boxes in this perspective.

If several boxes share a perspective, you can *unmerge* any one of them by dragging its finite vanishing point (with only that box selected) with Shift. Unlike dragging without Shift, this change affects only the selected box. Once unmerged, any change of the box's perspective (by dragging with or without Shift or via the controls bar) will affect only that box.

Also, when you move a 3D box object in the Selector tool, its own set of vanishing points moves along with the box, effectively unmerging its perspective if it was shared. This means you should avoid manipulating 3D boxes in the Selector tool if you want your 3D scene to keep a consistent perspective.

It is just as easy to *merge* two different perspectives together so they become one. To do this, select two boxes with different perspectives and move a finite vanishing point of one perspective to the same-dimension (same-color) point of the other—they will snap and join (similar to how the gradient handles join when dragged close enough, 10.4.3), as shown in Figure 11-13. The boxes will now have the same perspective. (If the perspectives had any infinite vanishing points, their direction angles must be the same for perspective merging to work.)

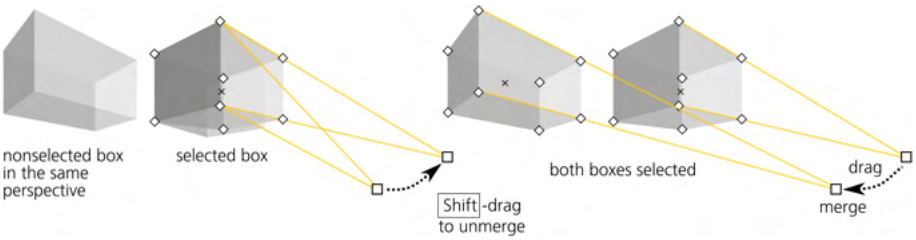


Figure 11-13: Merging and unmerging vanishing points

11.3.4 Handles

Each 3D box, when selected in any of the shape tools or the Node tool, displays eight diamond-shaped *corner handles* and an X-shaped *center handle*. Without modifiers, the four handles on the front X/Y side reshape that side, while the four others change the Z depth of the box. With Shift, however, their roles are reversed, as shown in Figure 11-14.

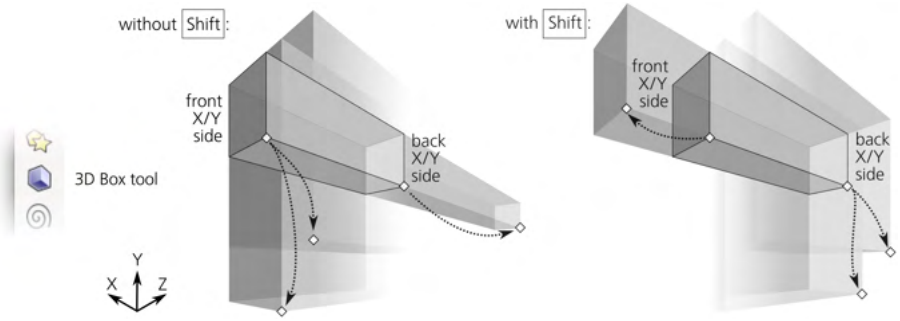


Figure 11-14: Dragging a box's corner handles with and without Shift

With Ctrl, the side-resizing handles snap to the continuations of that side's edges and its diagonal. For depth-changing handles, Ctrl has no effect (Figure 11-15).

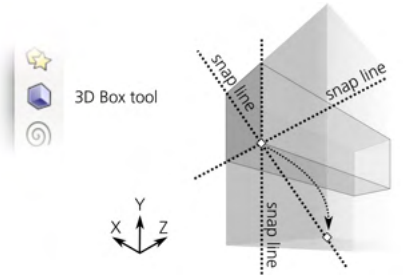


Figure 11-15: Dragging a box's corner handles with Ctrl

You can make any of the dimensions of the box negative by dragging a handle up to and beyond the other handle on the corresponding axis. This inverses the order of the sides (that is, the far side on that axis becomes the near side and vice versa). This is not recommended, though, because such an inside-out box does not always render its sides in the correct z-order.

Dragging the central handle without modifiers moves the box in the X/Y plane; with Ctrl, it snaps it to the directions of its X and Y vanishing points as well as the bisector between them. With Shift (with or without Ctrl), the central handle moves the box toward its Z vanishing point, as shown in Figure 11-16.

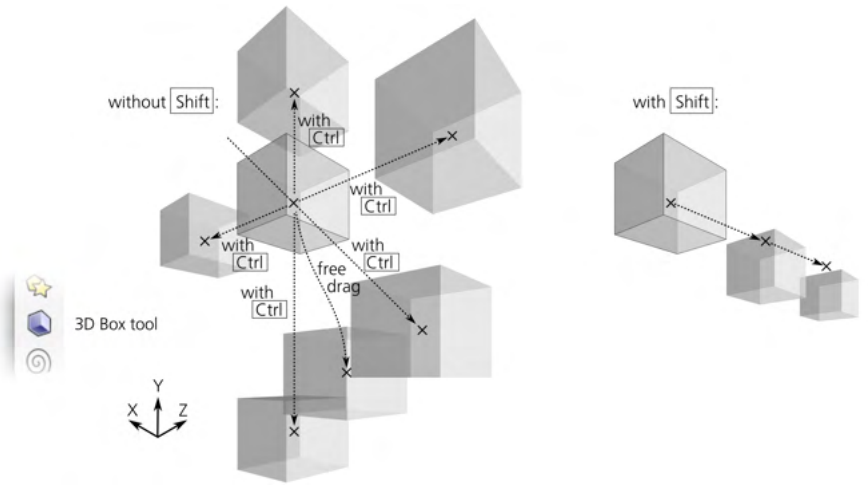


Figure 11-16: Dragging a box's center handle with and without Shift

As with rectangles, if you enable snapping of nodes to paths, nodes, and intersections (7.3.1.2), your 3D scene will feel very snappy—everything you draw or resize will be eager to join and line up, making it a pleasure to build solid gapless constructions out of boxes. As a simple example, to build a multi-story building, draw a single-story box, then duplicate it by Ctrl-D and Ctrl-drag the central handle of the copy upward to raise it to the next level until it snaps into place, as Figure 11-17 demonstrates.

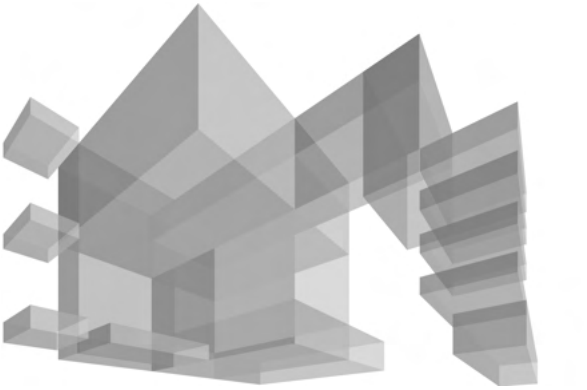


Figure 11-17: Using Ctrl and snapping handles to paths makes building complex scenes easy.

11.3.5 Styling

From the viewpoint of styling, a 3D box is no different from a group of paths. By default, all six sides of a new box have different shades of blue; you can paint the entire box with some color (removing any difference between the sides), or you can Ctrl-click and Ctrl-Alt-click to select any single side to paint it separately.

You can even *enter* the box just as you would enter a group (5.10) by selecting it and pressing Ctrl-Enter. After that, you can select single sides by simple clicking or using Tab and Shift-Tab. No matter how you style its sides, the box remains a box, so its styling is preserved in any 3D transformations. To remove the 3D box functionality, you need to ungroup it (Ctrl-U) or convert it to paths (Shift-Ctrl-C, which actually converts it to a *group* of paths).

Apart from changing fill or stroke paint, you can apply blur (17.1), clips, and masks (18.3) to an entire box or to any of its sides individually. If you lower the opacity of a 3D box, it works at the group level (4.8.2)—that is, you can see what's beneath the box, but you cannot see the hidden sides of the box itself. To be able to see the hidden sides, lower the opacity of individual sides by selecting them separately, as shown in Figure 11-18.

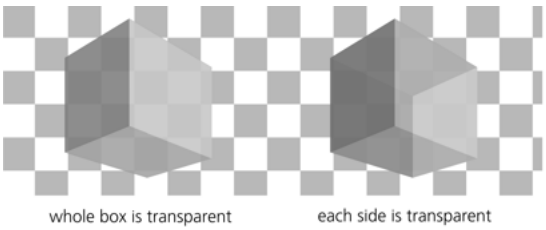


Figure 11-18: Opacity on the whole box vs. on individual sides

The *z-order* of sides within a box is automatically correct from the 3D viewpoint—that is, the sides that are supposed to be farther away from you are at the bottom, and the front sides are at the top of the z-order. However, the z-order

of different boxes is not enforced by Inkscape in any way; like any other shape tool, the 3D Box tool creates a new box on top of the current layer's z-order (4.4). Therefore, after drawing a box, you may need to move it in the z-order stack relative to other boxes for the composition to look correct.

For newly created boxes, the tool offers the same choice between the last set style and the tool's own style (11.1.2)—but with a twist. By default, the 3D Box tool uses the last set style, but unlike all other tools, it remembers the style last set on *the same side of a 3D box*, not on any object. For example, if you paint the top side of any box red, the top sides (but not the other sides) of all new boxes you create afterward will be red.

Since a 3D scene typically assumes a single light source, this behavior makes sense. Draw a single box and paint its sides as they would look when lit from, for example, the top-left corner. After that, all new boxes you create will be "lit" in a similar way.

NOTE *This tool remembers the last set style only when you assign it to individual sides, not the entire box. For example, if you just select a box and paint it all red, new boxes will not honor this. If you want all new boxes to have stroke but no fill ("wireframe"), the easiest way to achieve this is to enter a box as a group (Ctrl-Enter), select all its sides (Ctrl-A), and assign the stroke and remove the fill on all of them at once.*

The Tweak tool works on 3D boxes exactly as you would expect it to work on a bunch of paths (grouped or not). With path tweaking modes (12.6), once you distort the sides of a box, it ceases to be a 3D box and becomes a simple group. The Color Paint and Color Jitter modes, however, paint over boxes without destroying their 3D capability. It may make sense to enable only the hue and possibly saturation channels but not lightness (8.9.3), so that you can tweak the colors of boxes but preserve the relative lightness and darkness of their sides for the 3D effect, as shown in Figure 11 in the color insert.

11.4 Ellipses

An ellipse is a shape that can represent not only an *ellipse* or a *circle* but also an *arc* (a fragment of an ellipse or circle with a chord) or a *segment* (an arc plus two radii going from the ends of the arc and meeting in the center), as shown in Figure 11-19.

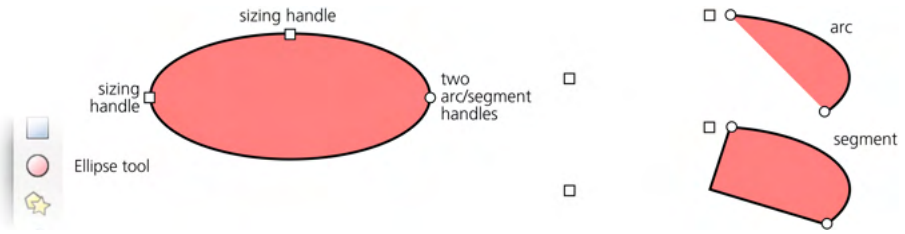


Figure 11-19: Handles on an ellipse, arc, and segment

11.4.1 Drawing

To switch to the Ellipse tool, press E or F5. Here are the ellipse drawing shortcuts:

- Dragging on the canvas creates an ellipse inscribed into the (imaginary) rectangle that your drag creates.
- With Ctrl, that imaginary rectangle is first restrained to a square or an integer ratio (1:2, 2:1, and so on), and then the ellipse is inscribed into it.
- With Shift, drawing starts from the center, so *one quarter* of an ellipse is inscribed into this rectangle instead of the whole ellipse. With both Ctrl and Shift pressed, you get a circle or an integer-ratio ellipse starting from the center (Figure 11-20).

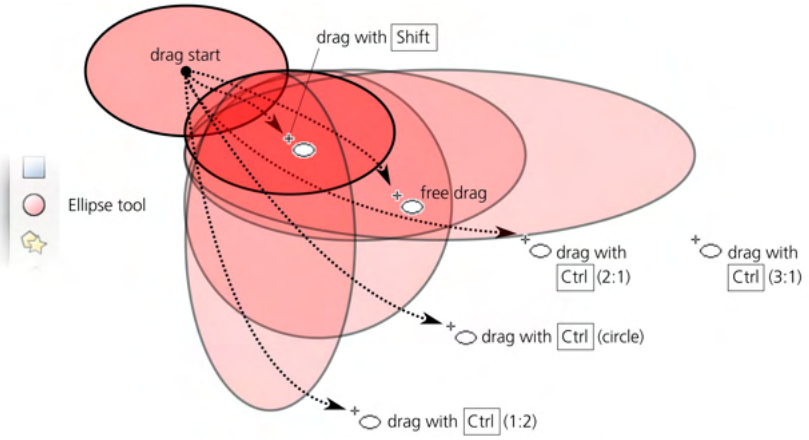


Figure 11-20: Drawing ellipses without Alt

- With Alt, the behavior of the tool changes (see Figure 11-21). Now it draws an ellipse whose *diagonal* goes from the start to the end points of your mouse drag. In other words, you start your drag in the 10:30 clock position on the ellipse and end at the 4:30 point, and the ellipse is squeezed as needed to fit this diagonal.

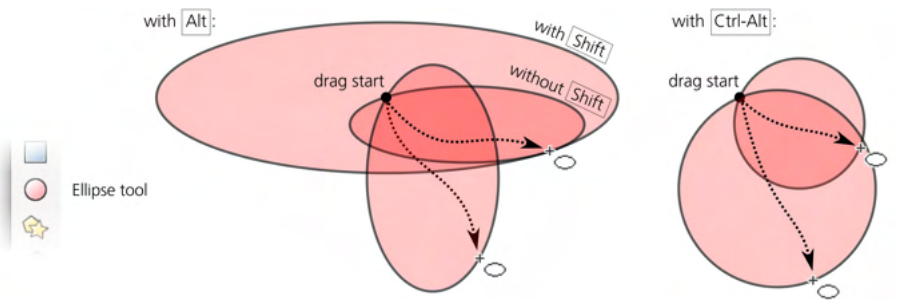


Figure 11-21: Drawing ellipses with Alt

- With Ctrl-Alt, the tool always draws a perfect circle, with its diameter stretching from the drag start to drag end. This is convenient if you need a circle with a given diameter.
- Adding Shift to Alt and Ctrl-Alt works the same as without Shift but begins drawing from the center—in other words, it creates a circle with a given *radius*.

One way to create circles of a fixed size quickly is by Ctrl-clicking in the Pen tool in the Straight lines or Paraxial mode (14.1.4). You can set the size of these circles on the Preferences page for the Pen tool. Add Shift to make them twice that size, or Alt to randomize their sizes, as shown in Figure 11-21.

11.4.2 Handles

Upon creation, an ellipse displays four handles: two square ones (top and left), a round one (right), and an X-shaped one (center). Just as with rectangles, the square handles are the sizing handles, and the round handle is in fact two handles sitting on top of one another. In ellipses, however, they're not for rounding but for turning an ellipse into an arc or segment, as you'll see shortly. The X-shaped handle moves the entire ellipse and is used for center snapping (7.3.1.3).

As with rectangles, the sizing handles of an ellipse change the ellipse's width and height *in the ellipse's own coordinates* instead of the document coordinates. No matter how you rotate or skew your ellipse, these handles remember the position of both axes of the ellipse and let you stretch or squeeze the ellipse with respect to those intrinsic axes, always preserving the position of the center, as shown in Figure 11-22.

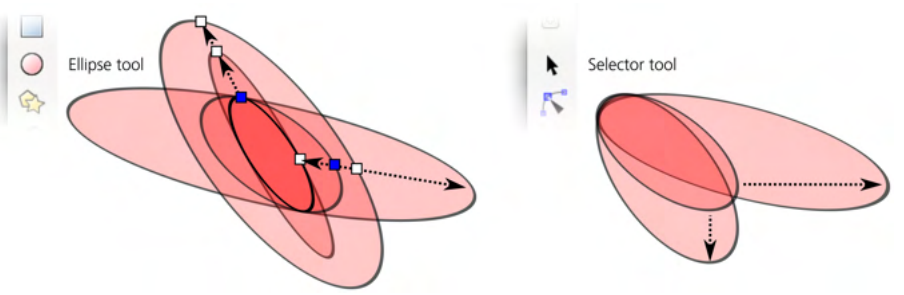


Figure 11-22: Transforming ellipses in the Selector and by the sizing handles

Here are the shortcuts of the ellipse sizing handles:

- Drag with Ctrl to turn an ellipse into a circle by making the other radius the same.
- Ctrl-click a handle to turn an ellipse into a circle without dragging. Note that the circle may still appear elliptic if skewing was applied to it.

Now let's look at the arc/segment handles—the round ones. To make a segment (an arc plus two radii), drag one or both of these handles *outside* the ellipse; to make an *arc*, drag while remaining *inside* it. Of course, the handle itself remains neither inside nor outside but exactly on the edge of an ellipse; the

phrase “drag inside” refers to where your mouse goes while you have that handle grabbed. Note that the sizing handles still work and remain in the same positions as for a whole ellipse (which may be outside your arc or segment).

The controls bar of the Ellipse tool lets you specify the exact angles of an arc, turn it into a segment, and make the ellipse whole (Figure 11-23).

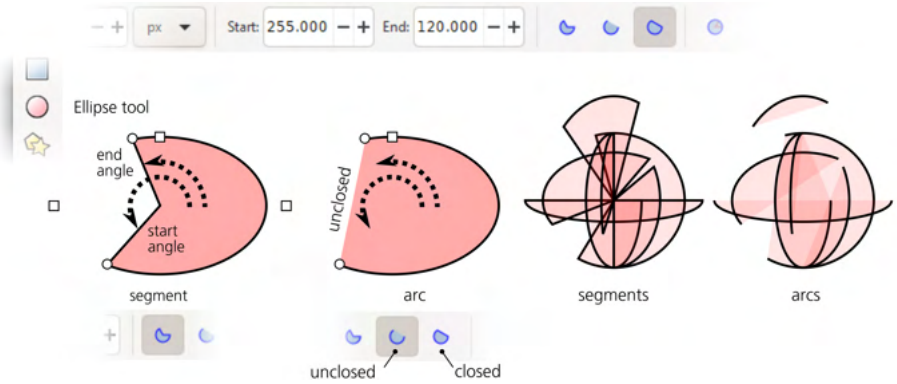


Figure 11-23: Segments and arcs

Note that unlike segments (which are always closed), arcs can be either *closed* or *unclosed*. An unclosed arc has the stroke going along the ellipse’s edge but does not connect the ends of the arc (you can make this obvious if you remove the fill, leaving only the stroke). Use one of the two arc buttons on the controls bar to switch an arc between closed and unclosed.

Here are the arc/segment handle shortcuts:

- With Ctrl pressed, snap the handle to angle increments (15 degrees by default, 6.3) when dragging.
- Shift-click a handle to turn an arc or segment into a whole ellipse.

Like all other shape parameters, start and end angles of an ellipse are remembered and reused for newly created shapes. It may be a surprise when you intend to draw an ellipse and get a narrow pie slice instead because you’ve recently turned some other ellipse into a segment.

11.5 Stars and Polygons

The Star tool (the * key or Shift-F9) creates two slightly different kinds of rotationally symmetric shapes: polygons and stars (Figure 11-24). Despite the simplicity of the idea, this is one of the most entertaining tools in Inkscape—perfect for wowing your friends!

A *polygon* is formed by a number of equidistant points lying on an imaginary circle and connected by segments. An Inkscape polygon has one diamond-shaped handle used to scale and rotate it, as well as control the shape of the connecting segments.

A *star* is a more interesting shape. It has two sets of equidistant points on two imaginary concentric circles, with the outline of the star zig-zagging back and forth between points on the inner and outer circles. Using a star’s two

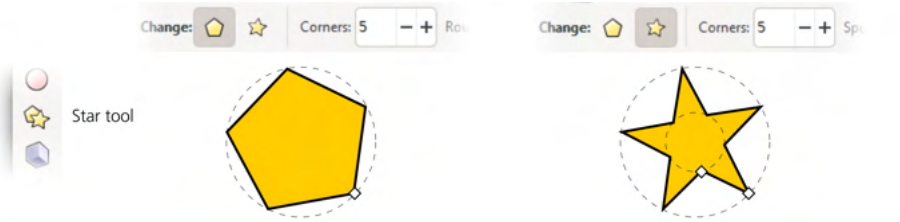


Figure 11-24: A polygon and a star

handles, you can vary the diameters of the circles and rotate the circles relative to one another around their common center, which produces a fascinating variety of symmetric shapes.

11.5.1 Drawing

Before you draw a new shape, decide whether you want a polygon or a star by pressing one of the toggle buttons on the controls bar, and choose the number of corners (convex vertices) in the numeric control. For example, a polygon with three corners is an equilateral triangle that will have three nodes if converted to path; a star with three corners, however, will have six nodes. You can always change the polygon/star type and the number of corners in an existing shape, too—just select it and edit the values.

The minimum Corners value is 2 for stars and 3 for polygons. A three-cornered polygon is, of course, a triangle, whereas a two-cornered star creates rhombuses or parallelograms.

NOTE *The maximum value for* Corners *is 1,000, but if you set it too high, it may slow down Inkscape unless you use the Outline mode (3.14).*

Drawing a shape is, again, as simple as dragging your mouse upon the canvas. A star or polygon is always drawn starting from its center. There's only one keyboard shortcut: dragging with Ctrl snaps the angle of one of the shape's corners relative to its center to 15-degree increments.

11.5.2 Handles

Dragging a polygon's diamond-shaped handle scales and rotates the polygon—something you can do just as well by transforming it in the Selector. The two handles of a star are more interesting.

First, by moving one of them closer or farther from the center of the star, you are changing the ratio of the diameters of the two circles on which the corners of the star lie. This ratio is called the *spoke ratio* and is also adjustable as a numeric parameter (disabled for polygons) in the controls bar (Figure 11-25).

You can even move the original inner handle farther away from the center than the outer one. The control will still show a ratio less than 1 because it always divides the smaller radius by the larger, regardless of which of them was initially inner or outer.

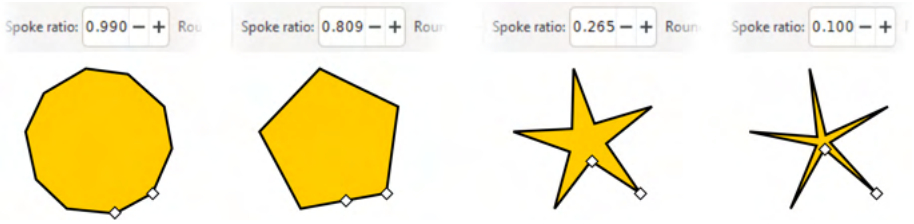


Figure 11-25: Adjusting the spoke ratio of a 5-vertex star

Then, the inner handle (initially on the inner circle) can be moved *tangentially*—along an arc around the star center—to skew the star's vertices, as shown in Figure 11-26. (Rotating the outer handle simply rotates the entire star.)

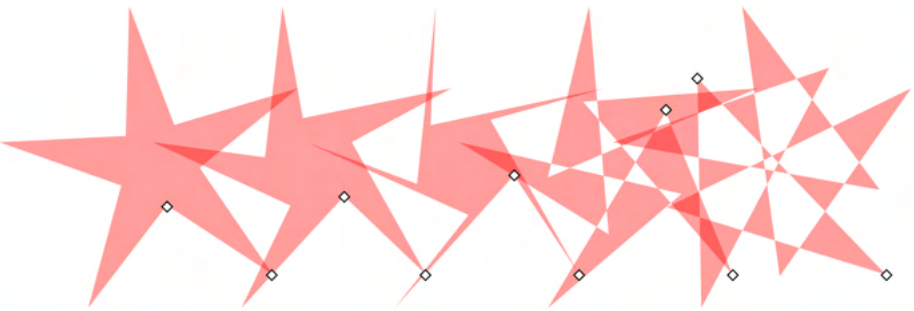


Figure 11-26: Moving the inner handle of a star tangentially

Drag the inner handle with Ctrl if you want to keep the vertices strictly radial (no tangential displacement), or Ctrl-click it to remove any existing tangential skew without dragging.

11.5.3 Rounding

You can create fascinating effects with a star or polygon by *rounding* it. This is different from rounding the corners of a rectangle; with a star or polygon, not only do the corners lose sharpness but the sides of the shape also bend into elegant Bézier curves (Figure 11-27).

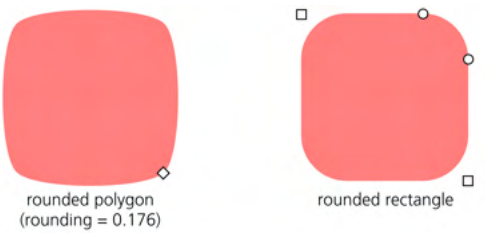


Figure 11-27: A rounded 4-sided polygon compared to a rounded rectangle

Note that a rounded square has straight line segments between circular or elliptic rounded corners. However, a rounded polygon or star has no straight lines at all—all of its segments become Bézier curves.

The Rounded numeric control of the Star tool is the ratio of the length of each Bézier handle (12.1.4) to the length of the polygon/star side that this handle affects. This parameter can be negative, which reverses the direction of tangents. Typically, values between 0.2 and 0.4 give the most natural-looking results. Values that are negative or too high positive may result in twisting, looping, and self-intersections. By varying this value, you can get an infinite variety of beautiful shapes, as shown in Figure 11-28.

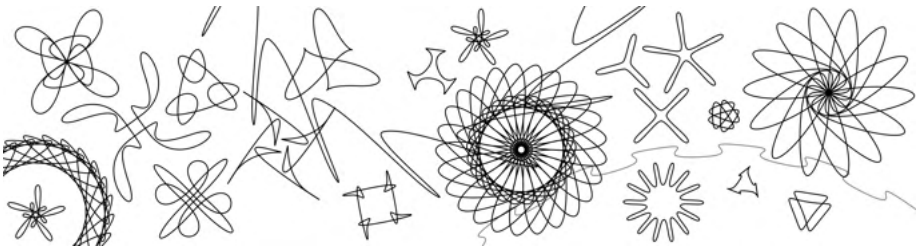


Figure 11-28: Fun with rounded stars

Apart from the numeric rounding parameter on the controls bar, you can round by Shift-dragging any of the handles tangentially. For positive rounding, rotate a handle counterclockwise around the star's center; rotate clockwise for negative rounding. Shift-clicking a handle removes any rounding.

If you want the outer corners of a star to be sharp but the inner ones smoothed, or vice versa, try creating an offset path (12.4) from the star.

11.5.4 Randomizing

Yet another way to make a star interesting is by *randomizing* it. Randomization moves all vertices of a star or polygon in random directions by random distances. If the star was rounded, randomization preserves the smoothness of all vertices.

The Randomized parameter in the controls bar controls the overall force of the effect. It can take positive and negative values. As you change this value—or simply Alt-drag a star's handle tangentially—the direction of random displacement for each node remains the same, and only the distance changes; a negative value simply moves the vertex in the opposite direction. In other words, the star remains randomized in the same way but to a varying extent.

By contrast, when you simply drag a randomized star's handle to scale or reshape it, or when you draw a new star with nonzero randomization, the shape trembles and jitters, abruptly changing the random displacements for all nodes (in mathematical terms, *reseeding* the randomization) with the slightest move of your mouse. So, if you want your star to become randomized differently but with the same overall amplitude, simply drag any of its handles slightly, as shown in Figure 11-29.

What are randomized stars good for? Randomness is a fundamental principle of nature, and randomness in design is a great way to make things look less rigid and more natural. Your artistic sense will tell you where and how much randomness is appropriate. Inkscape offers many sources of artistic randomness: you can shuffle the positions of objects (7.5.2), create patterns with random placement, scaling, and rotation (16.6), as well as randomly displace nodes in paths

Figure 11-29: Randomizing stars

to distort them (12.6.6). Still, randomized stars are unique in how easy it is to create a shape whose intrinsic symmetry emphasizes its randomness.

Slight randomization makes a star more lively—sometimes outright funny. Strong randomization produces a variety of wild and unpredictable shapes—messy hairballs, rounded amoeba-like ink blotches, fantastic landscapes at the edge of a large star with many vertices and spoke ratio close to 1. As an example, Figure 11-30 shows a star with 500 vertices, not randomized (left) and randomized by just 0.005 (right) to look more naturally hairy and to get rid of the moiré pattern.

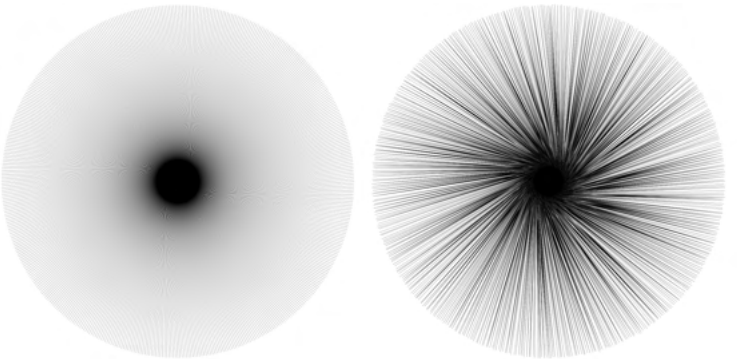


Figure 11-30: A 500-vertex star without randomization (left) and randomized by 0.005 (right)

As another example, here is a random but unclumped (7.5.2) scattering of pentagons, rounded at 0.28 and randomized at 0.15, looking very much like pebbles on a beach, as Figure 11-31 demonstrates.

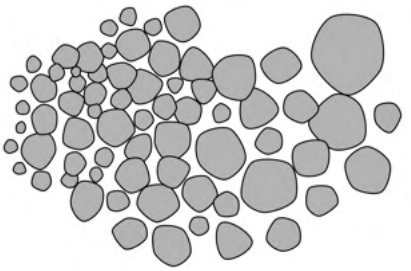


Figure 11-31: Pebbles: an unclumped scattering of rounded and randomized pentagons

11.6 Spirals

The Spiral tool (the I key or F9) creates another simple but versatile shape: a *concentric spiral*. Both as a drawing aid and as a shape in its own right, a spiral may be very useful—and no less exciting than a star.

A spiral, like a star, is drawn from the center. Dragging with Ctrl, as usual, snaps the drag point (the outer end of the spiral) to 15-degree increments (6.3). Unlike all other shape tools, the Spiral tool by default creates new spirals with its own style—no fill, black 1 px stroke—instead of the last used style (11.1.2).

Once drawn, a spiral displays two handles on both ends of the line. Simply dragging these handles circularly rolls the spiral in or out—from the inside or from the outside. In this way, you can, for example, create a spiral with only one turn.

The Turns parameter on the controls bar indicates how many full circles your spiral makes from the center to the outer end. The maximum number of spiral turns is 1000. The Inner radius value controls the inner end; it gives the percentage of the total turns where the spiral starts (Figure 11-32). For example, an Inner radius of 0 means that the spiral starts right in the center; 0.5 means that it starts half-way between the center and the outer end.

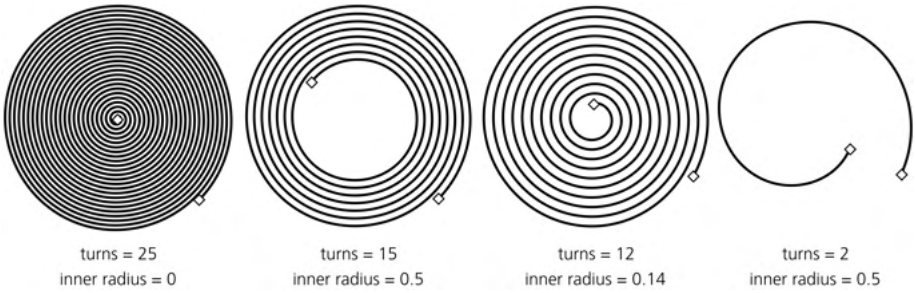


Figure 11-32: Adjusting the number of turns and the inner radius

The Divergence of a spiral controls whether a spiral's winding is equispaced throughout (divergence = 1), becomes denser toward its center (divergence > 1), or becomes denser toward the periphery (divergence < 1). You can change this parameter either numerically via the controls bar or by Alt-dragging the inner handle up or down, as shown in Figure 11-33.

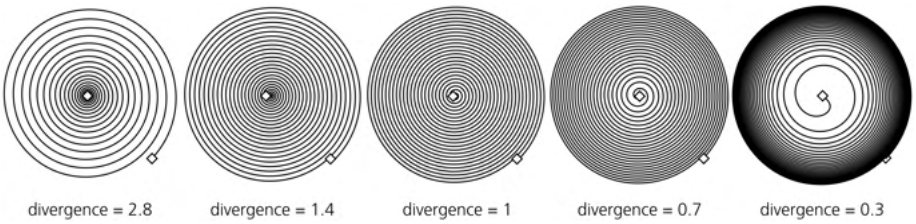


Figure 11-33: Adjusting divergence

Here are the other shortcuts for the outer handle:

- Shift-drag to scale or rotate around the center (no rolling on unrolling); this is the same behavior as when drawing the spiral.
- Alt-drag to lock the radius while rolling or unrolling. This way, the entire spiral becomes denser or sparser without changing its overall size.

Inner handle shortcuts include:

- Alt-drag vertically to adjust divergence.
- Alt-click to reset divergence to the default of 1.
- Shift-click to reset the inner radius to 0 (that is, move the inner handle all the way to the center).

When does it make sense to use spirals? A spiral fills a given space with a uniform concentric pattern—something that may not be easy to achieve manually or using any other tools. It may therefore serve as a drawing guide or carrier for other objects, such as markers (9.5.1), dash patterns (9.4), text on path (15.2.4), or live path effects (Chapter 13). Also, like the Ellipse tool, the Spiral tool can be convenient for creating curves with smoothly varying curvature; unlike with a plain Bézier curve, you can make an arc or a spiral shorter or longer by dragging a handle *along* the curve without affecting its shape.

12

EDITING PATHS

In a typical vector drawing, an overwhelming majority of objects are paths. That's why being familiar with paths is so important—otherwise, you cannot really say you know how to work in a vector editor. Inkscape provides a rich selection of tools, commands, and effects that work on paths.

I'll start with the basics of SVG paths and the traditional path tools whose analogs you may have already seen in other software. Then, in the second half of this chapter and in the next, I'll describe the advanced Inkscape techniques for path editing that are often more efficient—and almost always more fun to use. You may find, for example, that the Tweak tool (12.6) is so natural that you will rarely need to resort to the more technical and low-level Node tool (12.5). Still, it is important that you know what a path consists of and are able, when necessary, to manipulate path nodes directly, as this is one of the cornerstones of vector graphics of any kind.

12.1 The Anatomy of a Path

A path is a sequence of *nodes* (points) connected by straight or curved *segments* (Figure 12-1). Each node may have either one neighboring segment (if it's an end node) or two (if it's a middle node); SVG does not support branching paths with more than two segments joined at the same node.

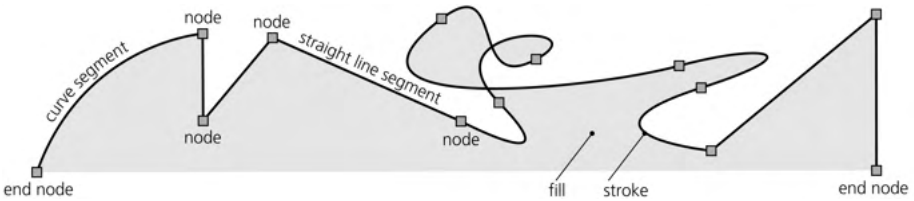


Figure 12-1: Path = nodes + segments

The length of a path is not limited; it may have anywhere from two to many thousands of nodes. It is also legal to have a path with one or even zero nodes, but such a path is invisible.

With a path, you can at least approximate (and in many cases, reproduce exactly) any conceivable shape, form, or figure. Depending on the shape you need and the required precision, in the worst case you'll just have to use many densely positioned nodes (Figure 12-2).

Figure 12-2: You can approximate the same figure roughly with a few nodes or reproduce it more precisely with more nodes.

12.1.1 Subpaths

In a path, a pair of nodes that are adjacent in sequence may not be connected by a segment. This produces a gap in the path, and each such gap divides the path into disconnected parts called *subpaths*, as shown in Figure 12-3.

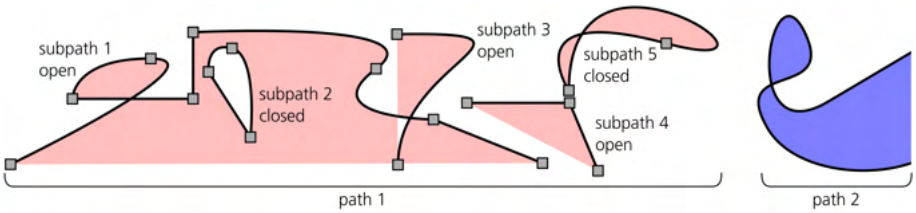


Figure 12-3: Subpaths are groups of connected nodes within a path.

A path without any gaps is said to consist of a single subpath. Any subpath, just like a piece of rope, can be *open* (with two loose ends, called *end nodes*) or *closed* (tied into a loop, so that its end nodes are one and the same node).

In many respects, subpaths look and behave just like separate paths. You can always convert the subpaths of a path into independent path objects by using the Break Apart command in the Path menu. The opposite command, Combine, converts several selected paths into subpaths of a single path.

When combining paths with different styles, you will lose the styles of all but the topmost (in z-order) selected path—because a single path, with no matter how many subpaths, can only have a single style. However, path data is never lost or added: if you combine and then break apart any number of paths, no nodes are changed in any way.

Every subpath has a *direction*—that is, its nodes are always ordered from start to end. In a closed subpath, the start node and end node are the same node; in an open subpath, the start and end nodes are different. Usually, the direction of a subpath does not matter; situations where it matters include using the start and end markers (9.5) and text on a path (15.2.4). The direction may also affect the fill of the path via the fill rule (12.1.2). You can visualize path direction in the Node tool if you enable path outline (Figure 12-17) and, in the tool's Preferences, turn on the Show path direction on outlines checkbox. Use **Path ▸ Reverse** (Shift-R) to flip the direction of the selected paths (or, when in the Node tool, of the subpaths with selected nodes).

12.1.2 Filling Paths

No matter what you use to fill your path—solid color, gradient, mesh, or pattern (8.2)—there are several important things to be aware of.

Fill always stops at the path itself—that is, at the centerline of the path's stroke, if it has any. Closed subpaths that do not intersect themselves or other subpaths are simply filled on the inside. Open subpaths are filled as if there were a straight line segment between the end nodes of the subpath, as shown in Figure 12-4.

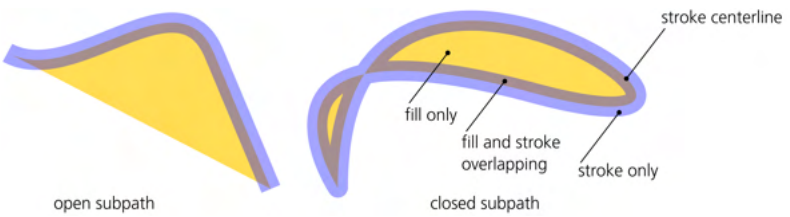


Figure 12-4: Filling open and closed subpaths

This straight line is not part of the path—it is not stroked, and you cannot, for example, bend it with the Node tool. If, when editing an open nonstroked path, you run into a straight line segment that refuses to be edited as you expect it to, most likely it's not really a segment but simply an edge of the fill. Close the subpath (12.5.4) to edit it without limitations.

When a path intersects itself or when one subpath is completely inside another, the decision on whether to fill some area depends on two factors: the *directions* of the subpaths surrounding the area and the *fill rule* of the path. The fill rule is a style property that can take one of two values, `nonzero` or `evenodd`, as set by one of the two toggle buttons in the Fill and Stroke dialog (Figure 8-3 on page 143):

- When the fill rule is `evenodd`, holes and loops are always left unfilled, except when you have a hole within a hole—then the inner hole becomes an island of fill.
- When the fill rule is `nonzero`, a loop or hole is filled only if its boundary is *counterdirected* relative to the outer path; it is empty if they go in the same direction.

With the fill rule of `nonzero`, those loops and holes that are filled are invisible unless the path is stroked. Usually they are not a problem, but sometimes you may want to get rid of them. The easiest way to do this is to select that one path and use the Path ▸ Union command (Ctrl-+). Unioning a path with itself removes all subpaths that do not affect its fill.

To reverse a path, use **Path ▸ Reverse** (Shift-R); if you want to reverse the direction of a subpath inside a path, use the Node tool to select one or more nodes in that subpath and then press Shift-R, which will apply only to the subpaths with selected nodes (if no nodes are selected, all subpaths are reversed).

12.1.3 Stroking Paths

The *stroke* of a path is a strip of paint that goes along the path itself, so that the path marks the *centerline* of the stroke. The stroke is normally painted on top of the fill, if there is any, but you can change that (9.6). The many style properties that affect the look of a stroke are covered in detail in Chapter 9.

The Path ▸ Stroke to Path command (Ctrl-Alt-C) converts the stroke of the selected path(s) to fill, as shown in Figure 12-5. In other words, it replaces a stroked path with a new path whose *fill* looks the same as the original path's *stroke*, honoring all the join, cap, miter, and dash properties of that stroke. The stroke paint of the original becomes the fill paint of the new path, and the original path's fill is discarded. If the original path had markers (9.5), the result will be a group where the stroke converted to path is grouped with markers that are now separate objects.

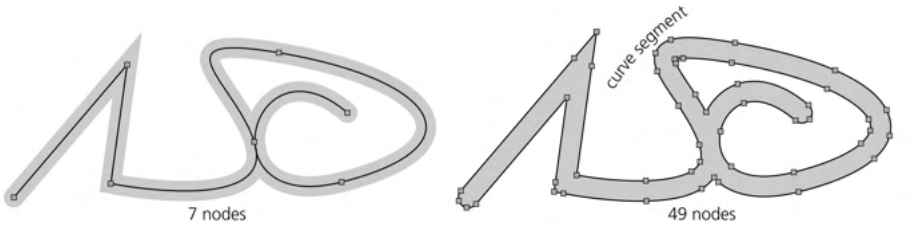


Figure 12-5: Converting stroke to path

12.1.4 Bézier Curves

As already mentioned, segments—parts of a path between the nodes—can be either straight or curved. Let's have a closer look at those curved segments—called *Bézier curves* after Pierre Bézier (1910–1999), a French engineer who was the first to use them in design.

A Bézier curve is completely determined by the position of four points, two of which are the *nodes* and two others are the *handles*, or controls. The curve itself is always completely inside the quadrilateral of these four points. In the Node tool (12.5), each handle is connected to its node by a straight line. These *handle lines* are always tangential to the curve at the corresponding node, as shown in Figure 12-6.

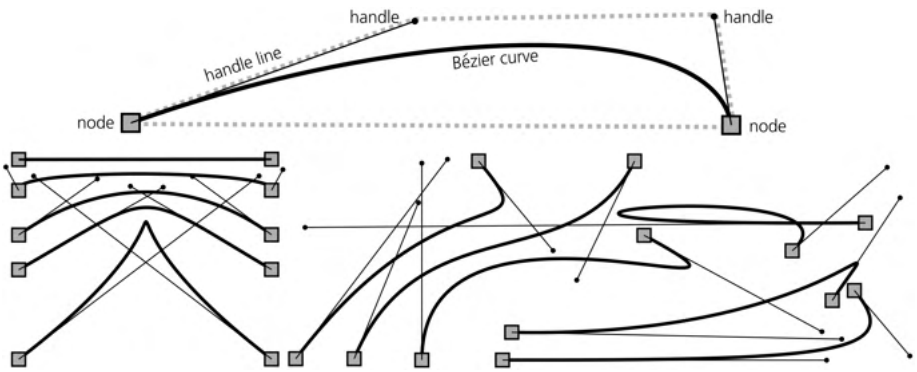


Figure 12-6: Bézier curves

If a path node is between two Bézier curve segments, it will have two handles connected to it, one for each adjacent segment.

The way a Bézier curve reacts to moving its handles is hard to describe, but you will quickly get a feel for it once you try. A Bézier curve may be indistinguishably close to a circular arc, but it may also have sharp bends, almost cusps; it may self-intersect or be perfectly straight when the handles are on the line between the nodes (or fully retracted).

Of course, for all its versatility, not many shapes are possible with a *single* Bézier curve. When building a path to approximate something (for example, when manually tracing over a bitmap, 18.5.2), experience will tell you how far you can reach with the current Bézier segment and where the best place for the next node is. Forcing a path to more closely approximate some real-life shape usually involves subdividing its Béziers by adding nodes and adjusting their handles. In contrast, simplifying a path, either manually or with the Simplify command (12.3), usually reduces the number of nodes and results in longer Béziers.

12.2 Boolean Operations

British mathematician George Boole (1815–1864) didn't specialize in geometry. He invented his *Boolean algebra* for dealing with the logical values of "true" and "false." Later, however, it was discovered that the same concepts make perfect sense for various other mathematical objects, such as sets or arbitrary geometric shapes.

Inkscape supports a number of Boolean operations on paths. They are accessed from the Path menu or by keyboard shortcuts derived from the symbols of the corresponding mathematical operations. Some of them require exactly two objects to be selected; others will work on any number of selected shapes. All of them will accept not only paths but also text objects and shapes (except 3D boxes), automatically converting them to paths. If a Boolean operation fails (for example, due to a wrong number or type of selected operands), it will explain the reason in the status bar.

12.2.1 Union

The *union* of two or more paths (Ctrl-+) creates a path that covers, with its fill, every point that any of the original paths covered. It thus joins any number of paths into a single path, giving it the style of the bottommost selected object, as shown in Figure 12-7.

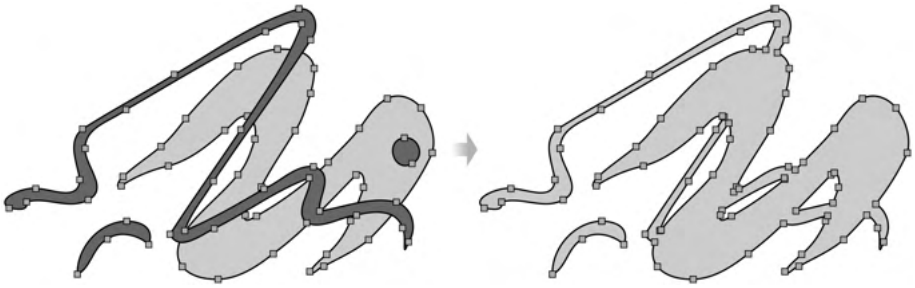


Figure 12-7: Unioning paths

If the paths do not overlap, the result will be exactly the same as for Path ▸ Combine (12.1.1). However, if the paths do overlap, this command, unlike Combine, will never create any holes; it will create new nodes where the paths intersect and remove any parts of the path that would end up inside the fill of the resulting path.

For example, if a small circle is completely inside a bigger circle, a Union of those two circles will simply remove the smaller inner circle. If you want the smaller circle to become a hole in the larger one, use Difference (Combine will also work unless the resulting path has the fill rule of `nonzero` and the two circles happen to be counterdirected, 12.1.2). Still, the Union command makes sense even for a single selected path, as it allows you to quickly clean up the path of any inner parts that do not affect the fill.

12.2.2 Difference

The *difference* of two paths (Ctrl-–) creates a path whose fill covers all points that were covered by the bottom (in z-order) path but not the top one—in other words, it *subtracts* the top from the bottom. The result has the style of the bottom object, as shown in Figure 12-8.

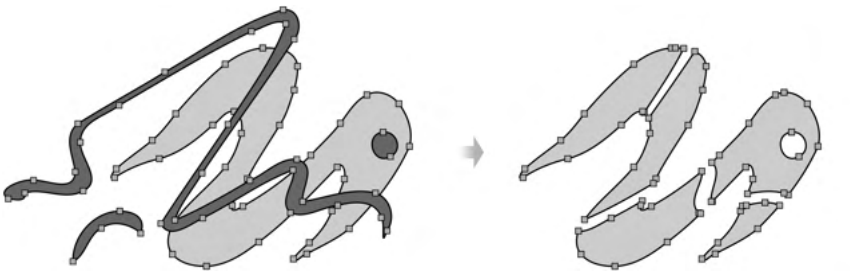


Figure 12-8: Subtracting a path from another path

If the paths do not overlap, Path ▸ Difference simply deletes the top path; if the top path completely overlays the bottom path, the result will be blank (both objects are deleted and nothing is selected). This command is the primary tool for creating holes and erasing the parts of paths you don't need.

The Calligraphic pen (14.2) *unions* the new path it creates with the selected one when you draw with Shift and *subtracts* from it when you draw with Alt. In the Eraser tool (14.4), one of the modes also subtracts what you draw from the selected paths or shapes.

12.2.3 Intersection

The *intersection* of two or more paths (Ctrl-*) creates a path whose fill covers only those points that were covered by *all* original paths. It uses the style of the bottommost selected object for the result, as shown in Figure 12-9.

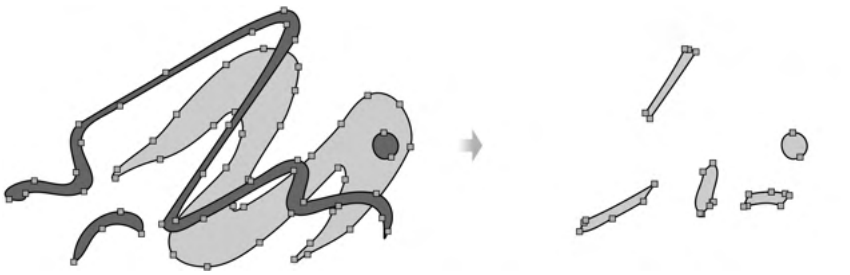


Figure 12-9: Intersecting paths

If at least two of the selected paths do not overlap (that is, their intersection is empty), Path ▸ Intersection deletes all paths without creating anything. This command is similar to clipping one path with another (18.3), except that a clipping path is nondestructive and works on any object, not just on paths; on the other hand, Intersection can intersect any number of paths at once.

12.2.4 Exclusion

The exclusion of two overlapping paths (Ctrl-^) creates a path whose fill covers the points that were covered by *only one* of the original paths. It uses the style of the bottom object for its resulting path, as Figure 12-10 demonstrates.

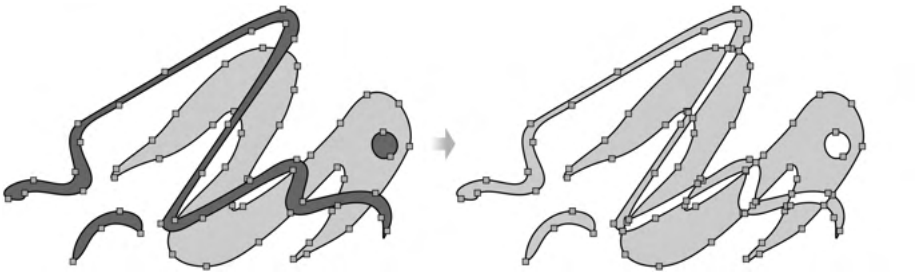


Figure 12-10: Excluding a path from another path

When the two paths do not overlap, the result of Path ▸ Exclusion is exactly the same as that of Path ▸ Combine. When they do overlap, the result *looks* exactly like a Combine, but the actual path is different: it has new nodes in the points where the outlines of the original paths intersect, whereas Combine creates no new nodes.

12.2.5 Division

A *division* of two paths (Ctrl-/) cuts the bottom path into pieces by the top path, deleting the top path, as shown in Figure 12-11.

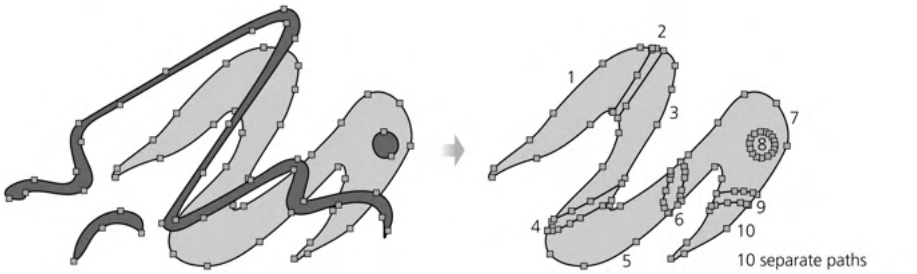


Figure 12-11: Dividing a path by another path

12.2.6 Cut Path

The Cut Path operation (Ctrl-Alt-/) is similar to Division. The main difference is that Cut Path does not create any new nodes or segments along the cut line, thus leaving the resulting paths unclosed. It also removes any fill of the path being cut. It is natural to use Division for slicing filled paths and Cut Path for cutting stroked paths without fill, as shown in Figure 12-12.

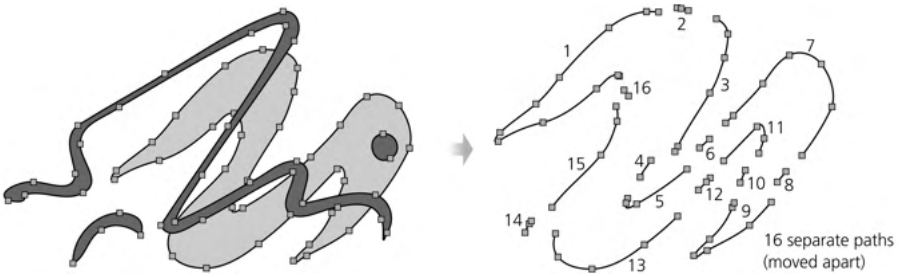


Figure 12-12: Cut Path is similar to Division.

12.3 Simplifying

An important operation on paths is *simplifying*. When you simplify a path (Path ▸ Simplify or Ctrl-L), Inkscape attempts to redraw that path using fewer nodes, ironing out smaller details but preserving the large-scale features of the path. If this description sounds a bit vague, it's because the operation itself is not entirely deterministic; usually, it is difficult to predict the result of simplifying accurately before you try it. Even the reduction of the number of nodes is not guaranteed (although common).

Despite that, simplifying is a very common operation and a true lifesaver for certain styles of artistic drawing. In technical drawing, on the other hand, it is rarely useful—if only because it considers any sharp corners in a path as "defects" to be smoothed out.

One effect you can almost always count on is that if your path has nodes that can be deleted without *any* change in the shape of the path, they *will* be deleted by simplifying. This includes any nodes you added with the Node tool (12.5.3) or with the Add Nodes extension (13.4.2.1) but never moved from their initial positions.

Path simplification is similar to gradient simplification in the Gradient tool, also invoked by Ctrl-L (10.5.1). Gradient simplification also removes any gradient stops that you added but didn't yet move from their initial positions.

Every simplification operation uses a certain *force*. Weak simplification changes the path just a little, removing only the most obviously redundant nodes. Stronger simplification changes the path more and smooths out larger bumps on it.

The default force used when you press Ctrl-L *once* can be set as the Simplification threshold value on the Behavior page of the Preferences dialog. The default is 0.002; anything greater than 0.01 is probably too strong for most cases. If you change it at all, consider lowering this value, because it is actually the *minimum* simplification force; you can always temporarily strengthen your Simplify command, raising this value without going into the Preferences dialog.

How do you make Simplify stronger? Just press Ctrl-L several times in quick succession (in Figure 12-13). Each invocation of the command increases the force a little, provided it happened less than half a second after the previous invocation. With such *accelerated simplification*, you can apply just the amount of simplification you need for each path. If the first keystroke didn't smooth the path enough, just keep pressing Ctrl-L, and it will gradually pick up. If you wait

more than half a second, though, the simplification force is reset back to the default value as set in Preferences.

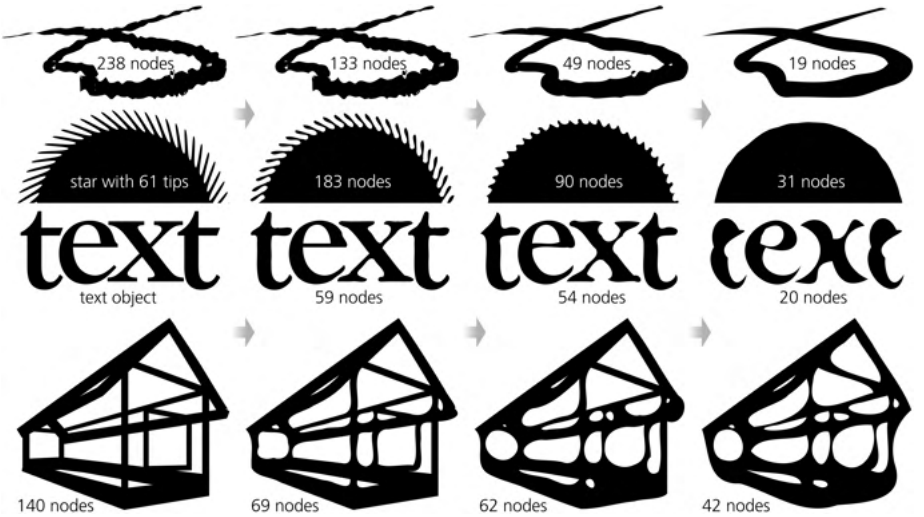


Figure 12-13: Simplifying paths

Figure 12-13 shows some examples of how simplification affects paths (see also Figure 12-2 on page 224, which was produced by gradual simplification of the most node-rich rose silhouette). As you can see, apart from reducing the number of nodes and ironing out small details, this operation melts sharp corners and curves straight lines, producing a natural—and often artistically engaging—kind of distortion.

12.4 Offsetting

Offsetting a path means expanding or contracting it in such a way that each point moves perpendicular to the path in that point. Offsetting inward is called *insetting*, and offsetting outward is *outsetting*. Imagine that your path is an island; outsetting would enlarge it by moving every point of the shoreline the same number of steps seaward, and insetting would make it smaller by allowing the sea to encroach inland by the same number of steps everywhere, as shown in Figure 12-14.

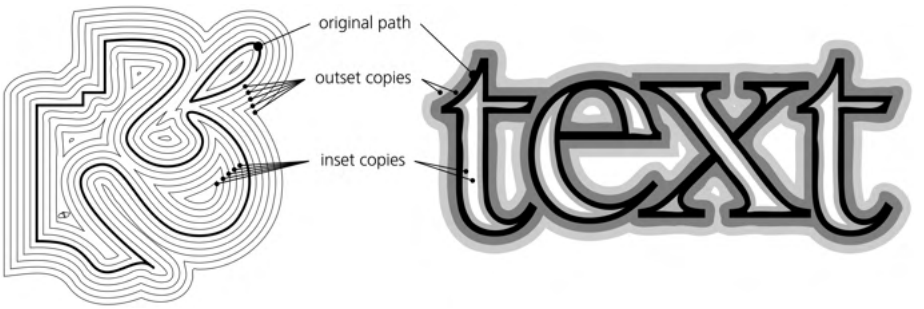


Figure 12-14: Offsetting paths

To inset the selected path or paths, press Ctrl-((inset by 2 px), or Alt-((inset by 1 screen pixel at the current zoom), or Shift-Alt-((inset by 10 screen pixels). To outset a path, use the) key with the same modifiers. On most keyboards, the 9 and 0 are on the same keys as (and), so the digits will work the same as the parentheses.

Offsetting is used in situations where you want to make a path "bolder" or "thinner" without changing its overall shape. This is useful for shadows, outlines, halos, bevels, and the like. (Sometimes, instead of outsetting you can simply set a wide enough stroke on the path, colored the same as its fill.)

If you outset and inset a path several times, it becomes distorted in a characteristic way, similar but different from the distortion of simplification (12.3). Such distortion welds together parts of a path—rounding corners, smoothing the intersections, and fusing together close subpaths within a path. For example, union all brush strokes in a Calligraphic pen drawing, and do a few inset/outset cycles on it to make it appear more natural and worn-down (Figure 12-15).

Figure 12-15: Melting complex paths with repeated offsetting

Offsetting, just like simplification, is a destructive operation: you cannot restore the exact original path except by undoing it. (For one thing, offsetting an open path always closes it.) However, Inkscape also has two dynamic object types, *linked offset* and *dynamic offset*, which store the exact original path and let you adjust the amount of offsetting without accumulating distortion (13.3.11).

The Inset and Outset commands, as well as the linked and dynamic offsets, apply the same offset distance to the entire path. It is possible, however, to inset or outset just a part of a path (for example, one end of a calligraphic brush stroke) using the Tweak tool's Grow and Shrink modes (12.6.4).

12.5 The Node Tool

Like everything else in Inkscape, the Node tool—the second button from the top in the main toolbar, accessible by pressing N or F2—strives to make simple things easy and hard things possible. This may be the most complex of all Inkscape tools—at least, the number of keyboard and mouse shortcuts in the Node tool is greater than in any other tool. You don't need to know all of its tricks in order to use Inkscape efficiently, but you should know the basics.

12.5.1 Path Display

After you switch to the Node tool, any selected path displays its *nodes* as little gray squares, diamonds, or circles (depending on the type of each node, 12.5.5). As a bonus, shapes (Chapter 11), gradients, meshes, and patterns (Chapter 10), as well as many of the path effects (Chapter 13), also display their editing handles in the Node tool.

You can *select* some or all of the path nodes; a selected node becomes blue and slightly larger. *Handles* of Bézier curves (12.1.4) are visible only for selected nodes and their neighbors. Even then, you can suppress these handles, if they get in the way, using a button on the controls bar, as shown in Figure 12-16.

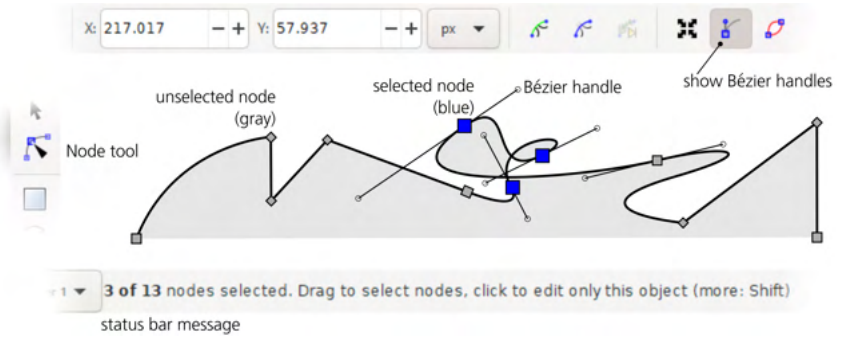


Figure 12-16: Nodes and Bézier handles in the Node tool

The selected paths are not, by default, visualized in the Node tool other than by showing their nodes. Normally, you would watch the path's stroke and/or fill update live in response to editing the nodes. Sometimes, however, your path may be too transparent or blurred, or may have some path effect (Chapter 13) applied; in that case, you can ask Inkscape to highlight the actual path with a red line by toggling another button on the controls bar (Figure 12-17).

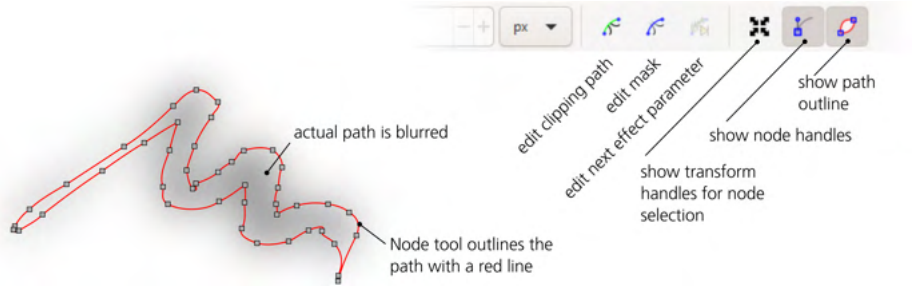


Figure 12-17: Path highlighting in the Node tool

NOTE *You can change the color of the highlight from the default red for each object individually, using the* Objects *dialog (4.9.5).*

Apart from the path itself, the Node tool allows you to edit some other paths associated with an object—they are invisible but affect the way the path looks: the clipping path (shown as light green, 18.3), the mask path (shown as blue, 18.3), and the linked path parameters of a path effect (shown as dark green, 13.2.1). These correspond to the three buttons on the control bar; the clipping path and mask buttons are on/off toggles, but the Show next editable path effect parameter button switches to the *next* linked path parameter (as there may be more than one, depending on the effect used).

Finally, as already mentioned, the Node tool displays all kinds of editing handles that objects (not necessarily paths) have. For example, in this tool you can edit your shapes (Chapter 11), adjust gradients, meshes, and patterns (Chapter 10), or change the dimensions of a flowed text object (15.2.2).

12.5.2 Selecting Nodes

Like so many other things in Inkscape, a path's nodes can be *selected* when in the Node tool. Not surprisingly, node selection methods are quite similar to those for object selection (Chapter 5).

NOTE *Unlike gradient handles (10.4.2), you can't style selected path nodes. Only the path as a whole can have a style, not its nodes.*

Along with selecting nodes in paths, the tool can also select objects (remember that object selection is common to all tools and commands in Inkscape). In the Node tool, you can use some of the shortcuts you know from the Selector tool: clicking selects an object (ignoring grouping), Shift-clicking adds to selection or removes from selection, and Alt-clicking selects under (5.9). The Node tool can edit nodes in multiple selected paths simultaneously (in many ways, it treats them as subpaths of a single path, 12.1.1).

This is similar with nodes. To select a single node in a selected path, just click it; the node becomes blue and slightly larger than a gray unselected node. Shift-click *adds* a node to the node selection; selected nodes need not be adjacent or even on the same path. Rubberband selection (dragging a rectangle around nodes, compare 5.7) also works; dragging with Shift adds nodes inside the rubberband to the selection.

If you click a path segment between two nodes, both nodes will be selected. Clicking an empty space (away from selected paths), just as pressing Escape, deselects any selected nodes—or, if there are no selected nodes, deselects any selected objects.

The Tab and Shift-Tab keys, which in the Selector tool go to the next or previous object, in the Node tool select the next or previous node on the selected path or paths. When the last node is reached, pressing Tab jumps to the first node; when the first node is reached, pressing Shift-Tab jumps to the last node. By the way, pressing Tab a couple times is a quick way to figure out the *direction* (12.1.1) of a (sub)path without changing the document in any way.

Also like in the Selector tool, Ctrl-A selects all nodes in all selected paths. However, if you already have some nodes selected, Ctrl-A selects only the nodes *in those paths and subpaths* that have selected nodes (this is similar to the way

Selector's Ctrl-A selects objects only within the current layer). To always select all nodes in all paths, use Ctrl-Alt-A. The ! key *inverts* selection by selecting what was not selected and vice versa within the (sub)paths that already have some nodes selected.

Yet another method of selecting nodes is unique to the Node tool. As you hover your mouse over a node (which is then highlighted red), you can *expand* or *contract* node selection by rotating your mouse wheel or pressing the Page Up and Page Down keys. Rotating the wheel *up* one notch or pressing Page Up adds the *closest* unselected node to the selection; rotating the wheel *down* one notch or pressing Page Down deselects the *farthest* selected node.

To determine the "closest" and "farthest" nodes, Inkscape measures the direct Euclidean distance from each node to the mouse pointer. However, if you hold Ctrl while rotating the wheel or pressing Page Up or Page Down, the distance will be calculated *along the path* and the selection will be limited to the (sub)path over whose node you are hovering.

12.5.3 Deleting and Creating Nodes

Deleting any number of selected nodes is as easy as pressing Delete or Backspace or clicking the "minus" button on the controls bar.

Deleting an end node of a subpath makes the subpath shorter, but you cannot open a closed subpath by deleting nodes—you'll need to *break* it as described in 12.5.4.

When deleting mid nodes (those between other nodes), Inkscape replaces each group of adjacent nodes being deleted with a single Bézier curve segment. In most cases, this introduces some distortion, but Inkscape tries to minimize it—it adjusts the handles on the remaining nodes so that the new Bézier segment runs as close as possible to the path fragment it replaces. In other words, deleting nodes on a path may work like a local Simplify command (12.3).

Sometimes, however, you don't need a new Bézier to bulge out all the way to ghost the nodes you're deleting, as you're trying to avoid changes to the handles of the remaining nodes. In that case, simply press Ctrl-Delete or Ctrl-Backspace, as shown in Figure 12-18.

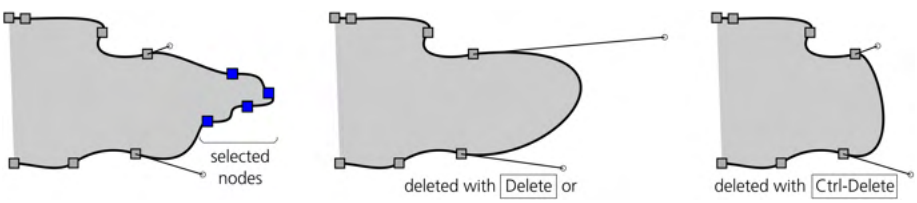


Figure 12-18: Deleting nodes

Unlike deletion, *inserting* new nodes is always possible in any point of a path without changing its shape. Simply double-click or Ctrl-Alt-click on the path (that is, on the center line of the stroke or the edge of the fill) where you want the new node to be. A new node is inserted, and the handles of its neighbor nodes are automatically adjusted so that the shape of the path remains unchanged (Figure 12-19).

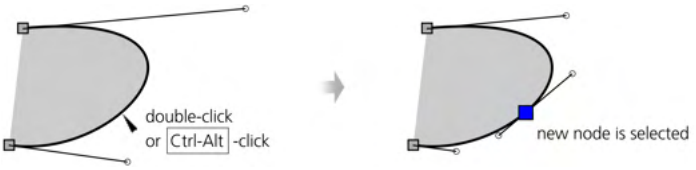


Figure 12-19: Creating a node by clicking

There is another node creation method that does not require clicking. Select two or more adjacent nodes and press Insert or Shift-I (or click the Insert node button on the controls bar) to insert a new node in the *middle* of each path segment (Figure 9-11 on page 169). Since the new nodes are then added to the selection, this is a quick way to multiply the number of nodes on a path—for example, if you start with two nodes and press Insert eight times, your path will have 257 nodes ($2^8 + 1$). This is similar to creating new gradient stops by pressing Insert (10.5.1).

[1.1] The Insert node button on the controls bar has a drop-down submenu of commands that insert new nodes at the *extremities* of the path or its selected part—but only if that extremity doesn't already have a node. For example, if you select two nodes with a curve between them that reaches lower (that is, further down on the Y axis) than either of the two nodes, you can use the Insert node at max Y command to insert a node at the lowest point of this segment (Figure 12-20). All nodes in extremities will be smooth nodes (12.5.5) with strictly horizontal (for min/max Y) or strictly vertical (for min/max X) handles.

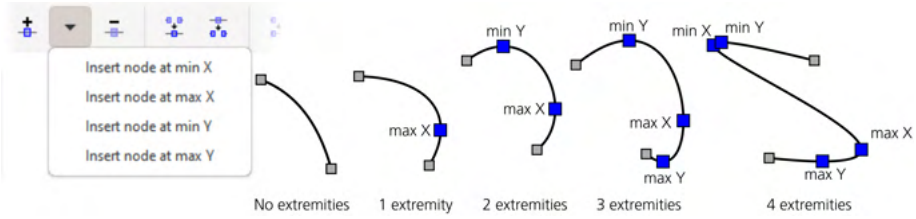


Figure 12-20: Inserting nodes at the extremities of a path

Yet another approach is *duplicating* nodes. With any number of nodes selected, press Shift-D. This "divides" each selected node into two, each inheriting one of the two handles of the original node. Figure 12-21 shows how it looks if you duplicate all nodes in a path and move away the new nodes (originally in the same locations as the old) by pressing →.

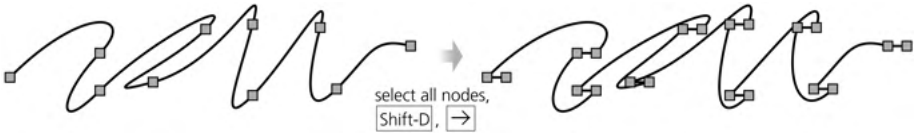


Figure 12-21: Duplicating nodes

The Shift-D method is especially useful for continuing an open (sub)path by duplicating and moving away its end node. For example, if you select an end node of a straight line segment (that is, an end node without a Bézier handle),

you can easily "draw" with line segments: just keep pressing Shift-D followed by arrow keys.

[1.1] 12.5.3.1 Copying, Cutting, and Pasting Nodes

The traditional Copy, Cut, and Paste commands and shortcuts work on nodes in the Node tool, although with limitations.

Copying a single node does nothing useful. Copying two or more selected nodes creates a path out of those nodes and places it on the clipboard. From there, if you're in the Selector or in any other tool, you can paste it back as a new path object created out of the copied subset of nodes. If you copy less than a complete closed subpath, the pasted path will be open.

If you paste a node selection copied from the Node tool in the same tool with a single path selected, you create a new subpath (12.1.1) in the selected path. It doesn't matter if any nodes are selected when you do the paste, because pasting doesn't insert the copied nodes between the existing nodes. It always creates a separate subpath, and depending on the fill rule (12.1.2), this action may produce holes in the fill. While that may be useful, incorporating the nodes into the path and not having a separate subpath sometimes makes more sense. To do that, exit the Node tool, paste the copied nodes as a path of their own, position that path as needed, and union it with your target path.

12.5.4 Joining and Breaking

To *join* two end nodes, first select them. These can be the end nodes of the same open subpath, in which case joining them will close that subpath. Or they can belong to different subpaths, in which case you will join these subpaths into a single subpath. They can also belong to different path objects, in which case these objects become one path (and take the style of the path that was selected first).

There are two ways to join, corresponding to the two join buttons on the Node controls bar. The first method—the Join nodes button or Shift-J—actually moves and joins the two end nodes into a single node. The second method—the Join with segment button—leaves the end nodes where they are but adds a new path segment between them. If you use the first method but don't want one of the end nodes to move, hover your mouse over it to lock its position while pressing Shift-J (Figure 12-22).

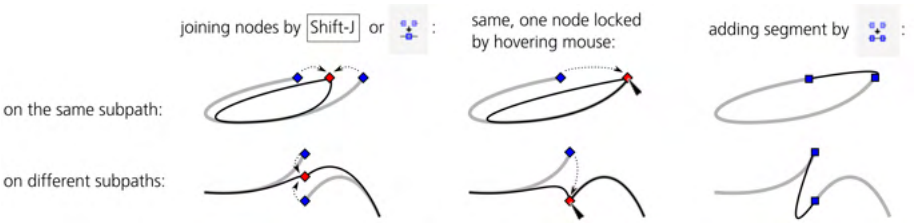


Figure 12-22: Joining nodes and inserting segments

Similarly, there are two ways to *break* a path. For the first method, select one or more non-end nodes and click the **Break nodes** button or press Shift-B. This will duplicate each selected node but without connecting it to the original node, so the path will be broken into subpaths at each selected node point. For the second method, select two adjacent nonend nodes, and click the **Delete segment** button to delete the segment between them, as shown in Figure 12-23.

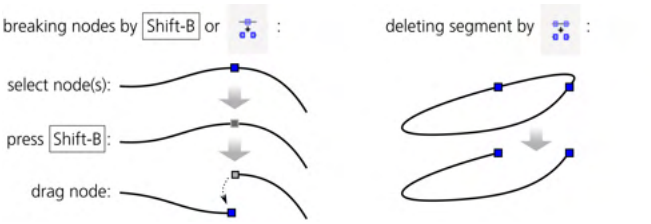


Figure 12-23: Breaking nodes and deleting segments

Cutting an open path with any of these methods produces new subpaths within a path (12.1.1). If you want to break a path into independent objects, first break it into subpaths and then use **Path ▸ Break Apart** to separate each subpath into a path object of its own.

12.5.5 Node Types

A mid node may have one or two handles attached to it, one for each side. In Inkscape, a node can belong to one of several *node types* that behave differently when you drag one of these handles or the node itself (Figure 12-24).

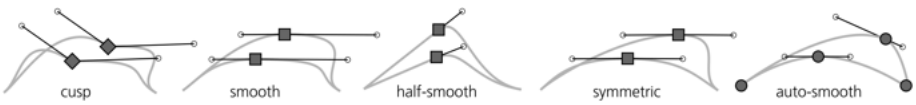


Figure 12-24: Node types: cusp, smooth, half-smooth, symmetric, and auto-smooth

- If a node has no handles (they are both *retracted*), or if one handle is non-collinear (not on the same straight line) with the opposite handle or segment, or if one handle remains unmoved when you drag the other one, such a node is called *cusp*—because when its two controls are at an angle, the node represents a sharp turn (cusp) in the path. Cusp nodes are shown as diamond shapes.
- If the other handle rotates so as to always be collinear with the handle you're moving, such a node is called *smooth* because it keeps the path flowing smoothly. Smooth nodes are shown as squares.
- The node may have only one handle, with a Bézier curve on one side but a straight line segment on the other, and the only handle of the node may be *locked* to be always collinear with the line segment—so that, when you drag the node, its handle automatically rotates to remain collinear. Such a node is called *half-smooth*; it is also shown as a square, but unlike other node types, it doesn't have its own button or shortcut. To make a node half-smooth, convert a segment between two smooth nodes to a straight line with the

Make selected segments lines button (12.5.6.1) or simply retract the inner handles on the segment's nodes (their outer handles will line up with the segment and the nodes will become half-smooth). You can also convert a cusp node with one handle, adjacent to a straight line, to smooth (Shift-S) *once* to make it half-smooth; another smoothing will make it fully smooth with two handles.

- The other handle of a smooth node may rotate and scale so as to always be both collinear and have the same length as the control you're moving. Such a node is called *symmetric* because its handles are always symmetric around it. Symmetric nodes are also shown as squares.
- An auto-smooth node, shown as a circle, is a smooth node that adjusts its handles automatically when you move it around. You should not manually edit the handles of an auto node; if you do, the node will downgrade itself from auto-smooth to smooth. It is best, if you use auto nodes, to hide the handles altogether using the controls bar button (Figure 12-16 on page 234) so they don't get in the way.

  An auto-smooth node adjusts the angle and length of its handles so as to make the adjacent path segments as smooth as possible. If the adjacent nodes are also auto-smooth, their handles will adjust, too. For example, when you move an auto-smooth node A closer to its neighboring auto-smooth node B, both nodes will make their handles progressively shorter and rotate them toward each other so as to keep the curvature of the segment between them, as well as of the adjacent segments on both sides, at a minimum. This behavior is reminiscent of the Spiro spline path effect (13.3.7.2).

To change the type of node in a cycle (cusp to smooth to symmetric to auto-smooth and back to cusp), Ctrl-click it. To change the type of several selected nodes, click one of the node type buttons on the controls bar or use the keyboard shortcuts:

- Press Shift-C to make the selected nodes cusp. The first Shift-C just changes the node type but does not move the handles; a second Shift-C retracts all handles of selected nodes.
- Press Shift-S to make the selected nodes smooth. If the previously cusp node had non-retracted handles, they will be rotated to be collinear; if its handles were retracted, they will be pulled out and made collinear at an angle that minimizes the curvature.

  If a node has a straight line segment (12.5.6.1) on one side but a handle on the other, the first Shift-S will make it half-smooth, locking the single handle to the direction of the line segment. Another Shift-S will extend the second handle, making the node fully smooth.
- Press Shift-Y to make the selected nodes symmetric.
- Press Shift-A to make the selected nodes auto-smooth.

Switching all nodes of a path from cusp to smooth or auto-smooth distorts the path in a characteristic way, removing straight lines and sharp corners, as Figure 12-25 demonstrates.

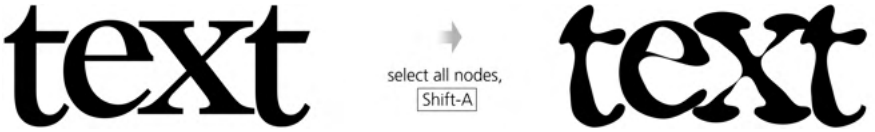


Figure 12-25: Converting node types in an entire path

12.5.6 Moving Handles

Perhaps the easiest way to edit a Bézier path segment is by dragging not a node or handle but the curve itself. This does not require any nodes to be selected nor does it move any nodes. Inkscape simply adjusts the Bézier handles of the two adjacent nodes so that the curve follows your mouse, as shown in Figure 12-26.

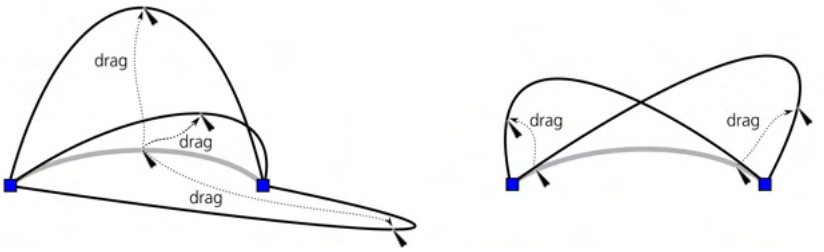


Figure 12-26: Curve dragging

If a node is smooth or symmetric, dragging the curve on one side of that node will also change the curve on its other side, because the movement of one of the node's handles will be mirrored by its other handle. Curve dragging next to an auto-smooth node converts the node to smooth.

Of course, you can also drag the Bézier handles of any selected node (if you don't see the handles, check that you have the Show handles button pressed on the controls bar, Figure 12-16 on page 234). Although they are shown only for selected nodes and their neighbors on the path, handles themselves (unlike nodes) are not selectable. As you drag a handle, Inkscape's status bar reports the total displacement as well as the current angle and length of the handle.

With Ctrl pressed, the handle you're rotating snaps to 15-degree increments. With Shift pressed, the other handle of the same node rotates by the same amount, preserving the angle between handles (as is always the case for smooth nodes; with Shift, however, this works for cusp nodes too). With Alt, you lock the length of the handle, so that only its angle changes. These modifiers work in any combination.

It is also possible to move node handles using keyboard shortcuts. In 12.5.7.3, you'll see that the < and > keys scale and that [and] rotate several selected nodes as if they were an object. Logically, when you have a *single* node selected, these same keys rotate and scale (that is, change the length of) the Bézier handles of that node without moving the node itself, as shown in Figure 12-27.

With Alt, you can even scale and rotate each of a node's handles separately. Just use the *left* one out of the two Alt keys on your keyboard to scale (with < and >) or rotate (with [and]) the *left* handle of the single selected node—that

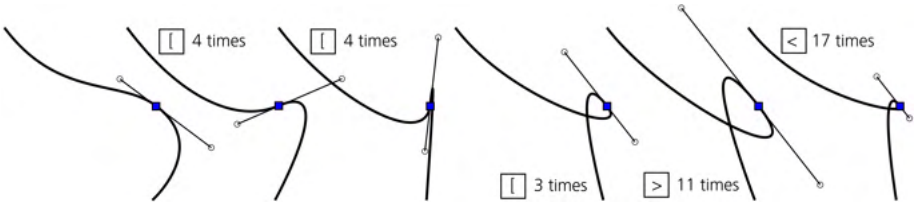


Figure 12-27: Adjusting node handles with keyboard shortcuts

is, the handle whose end is farther to the left than the other one. The right Alt will similarly affect the right handle.

12.5.6.1 Segment Types

What if you want to flatten a Bézier curve by converting it to a straight line segment? Actually, a straight line is just a Bézier curve with both its handles *retracted*—that is, coinciding with the corresponding nodes. To retract a handle, Ctrl-click it; to pull a retracted handle out of a node, Shift-drag it away from that node.

Another way to convert Bézier curves to lines—and back—is by using the segment type buttons on the controls bar. These buttons require that at least two adjacent nodes are selected, but they will work on any number of segments between selected nodes. The Make segment line button (or Shift-L) retracts any pulled-out handles; the Make segment curve button (or Shift-U) does not actually make the segment curvilinear, but it pulls out the handles and puts them along the segment, so you can move them manually, as shown in Figure 12-28.

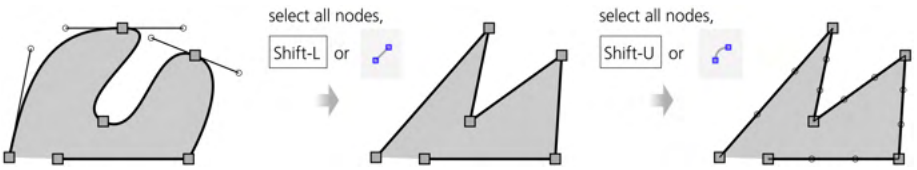


Figure 12-28: Changing the types of segments

12.5.7 Moving Nodes

Reshaping a path is as easy as selecting some of its nodes and dragging them around. The Bézier handles belonging to those nodes move parallel with them (except for half-smooth and auto-smooth nodes, which may rotate their handles as you drag them).

As in the Selector, simple click-and-drag works as expected for moving a single *unselected* node; if you drag a selected node, you're dragging *all* selected nodes with it. With Ctrl pressed, mouse dragging is restricted to moving horizontally and vertically.

Arrow keys move selected nodes in the same way and by the same distance as in the Selector tool (6.5.1): by 2 px (default value) without modifiers, by 10 times that distance with Shift, by 1 screen pixel with Alt, and by 10 screen pixels with Shift-Alt.

If you drag a node while pressing Ctrl-Alt, movement is restricted to the directions of the dragged node's Bézier handles and their perpendiculars. If a node has a straight line segment on one side, then the direction of that segment is taken instead of a handle. So, if the node's two handles or adjacent segments are collinear, you can Ctrl-Alt-drag it in one of four directions; otherwise, you can drag it in one of eight directions (Figure 12-29).

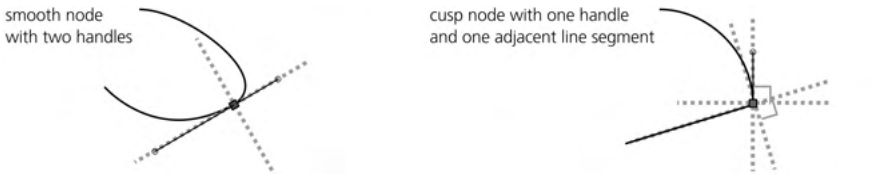


Figure 12-29: Directions of dragging a node with Ctrl-Alt

You can Ctrl-Alt-drag more than one selected node, but the movement will be restricted to the handles/segments of the node you actually drag with your mouse.

The nodes you drag with the mouse may snap (7.3.1.2) to guides, grids, and other objects or nodes. By default, snapping to guides and grid is enabled, but snapping to objects is not. You can temporarily disable snapping if you drag with Shift.

12.5.7.1 Moving Nodes Numerically

The coordinates of a single selected node are displayed in the X and Y fields in the Node tool's controls bar (Figure 12-30); editing those values moves the selected node to the new coordinates.

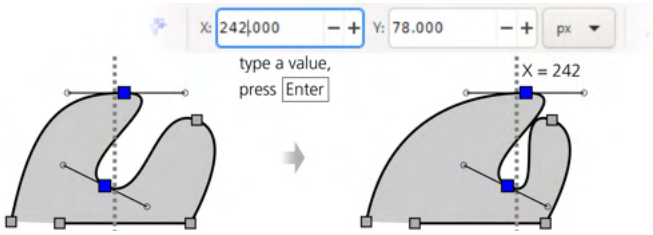


Figure 12-30: Using the X and Y editable fields to position nodes

If you have more than one node selected, these fields show their *average* coordinates—or, to put it another way, the coordinates of the geometric center of the selected nodes. In this case, typing a value in one of these fields moves the selected nodes as a whole so that their center takes the new coordinate value.

You can also line up selected nodes using a tool you have already seen used for objects: the Align and Distribute dialog (7.4). When you are in the Node tool, this dialog displays the Nodes section whose four buttons allow you to:

- Align the selected nodes horizontally.
- Align the selected nodes vertically.

- Distribute the selected nodes equispaced on X.
- Distribute the selected nodes equispaced on Y.

For aligning nodes, the Relative to drop-down list lets you choose the reference point of the alignment. For example, if you're aligning selected nodes vertically—in a column—you can choose whether that column will be in the Middle of selection (the default), at the Min value (at the left edge), at the Max value (at the right edge), or at the location of the First selected or Last selected node.

12.5.7.2 Node Sculpting

Simple dragging moves all selected nodes by the same distance, which often is what you need. For example, in a schematic face profile (Figure 12-31, left), you can easily make the nose longer by selecting two nodes and pulling them to the right—the result is acceptable for this style of drawing. However, what if you have a more complex and realistic drawing with lots of nodes (Figure 12-31, right)? No matter how many nodes you select, dragging them will introduce discontinuities and damage the natural silhouette of the face.

Figure 12-31: Pulling two noses

In such cases, it would be nice to be able to move different nodes by different distances so that the tip of the nose moves the farthest and the other nodes move less and less as you go along the path away from the tip. That is exactly what Inkscape does when you select all the nose's nodes and drag the tip node with Alt. This technique is called *node sculpting*.

In the simplest case, when all selected nodes are on the same straight line, Alt-dragging the middle selected node bends the path into a smooth bell-like curve. Farthest selected nodes stay put; the dragged node moves all the way; and all other selected nodes move by some intermediate distance. Now, if your selected nodes formed a wiggly line, text converted to path, or a realistic nose, Alt-dragging will smoothly bend them while preserving their features (Figure 12-32).

Figure 12-32: Sculpting nodes with Alt

When determining what nodes to move by what distances while Alt-dragging, the distance from the node being dragged is calculated along a straight line (spatially), not along the path.

If you don't have enough nodes on the part of a path you want to sculpt, just select the nodes you have and press Insert a few times to populate that part of the path with nodes. When sculpting complex shapes with many densely packed nodes, such as bitmap tracings (18.5), hide their Bézier handles (by unpressing the button on the controls bar) so they don't get in the way, and select nodes by expanding selection from a node (12.5.2).

Node sculpting is reminiscent of the Tweak tool (12.6) in that it makes path editing more natural and lets you develop complex shapes out of simple ones. However, unlike Tweak, this technique does not create or delete nodes—and it's more deterministic overall. Repeated Tweaking of a complex sprawling path will eventually simplify and degrade all of it, melting away small details even where you don't touch them with the tool. With node sculpting, on the other hand, only selected nodes are affected, and no degradation occurs no matter how many times you Alt-drag the selected nodes back and forth.

12.5.7.3 Transforming Nodes

What does *transforming nodes* mean? You already know many ways to move nodes around and even sculpt them. How is this different?

Remember that with the Selector tool, *transforming* includes not only moving but also scaling and rotating (Chapter 6). These kinds of transformations make perfect sense for a group of nodes on a path as well—if you think of that group as an "object."

[1.1] The Show transformation handles button on the controls bar does just that: displays the eight scaling handles—exactly the same as in the Selector—around the selected nodes on the path (Figure 12-33). Also like in the Selector, if you click any of the selected nodes, the scaling handles switch to rotation/skewing handles and back (compare 6.3). Most of the modifiers from the Selector tool also work: Ctrl-scaling to preserve ratio, Shift-scaling to scale around the center, and Ctrl-rotating to snap angle to 15-degree increments.

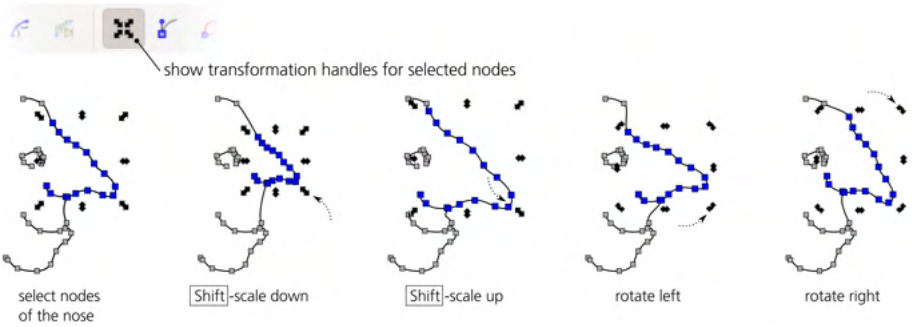


Figure 12-33: Transforming selected noses

Transforming nodes is also available via keyboard shortcuts. Just as in the Selector, the < and > keys scale the selected nodes, and the [and] keys rotate them as a whole. Without modifiers, rotation is by 15-degree increments and scaling is by 2 px; the same keys with Alt pressed rotate and scale by 1 screen pixel at the current zoom. The H and V keys for flipping (reflecting) horizontally and vertically also work.

By default, keyboard scaling, rotation, and flipping are performed around the geometric center of the selected nodes. However, if you hover your mouse cursor over one of the nodes, it will remain fixed while other selected nodes scale or rotate around it as a center. For example, you can select all nodes of an object by pressing Ctrl-A and then hover on a node and rotate the entire object around that node with [and].

12.6 Path Tweaking

You've already seen in 8.9 how the Tweak tool (W, Shift-F2) can be used to paint and jitter colors in objects, and in 6.10, how it can be used to move and transform objects. This versatile tool's remaining modes—*Push*, *Shrink/Grow*, *Attract/Repel*, and *Roughen*—are for editing paths.

TWEAK TOOL ANALOGS IN AI

Some of the Tweak tool's path editing modes are similar to the Pucker and Bloat tools in Adobe Illustrator.

The Tweak tool's approach to editing paths is fundamentally different from that of the Node tool. The Node tool, true to its name, enables you to edit nodes, and you need to have a working knowledge of how nodes define the shape of the path. With the Tweak tool, you can forget everything you ever knew about nodes; just interact with your path as a pliable body, like a lump of modeling clay, bending and sculpting it at any point in any direction. While hardly useful for technical drawing, tweaking paths is perfect for creating artistic images such as cartoons.

The Tweak tool works on any number of selected objects. For example, you can select all (Ctrl-A) and "smear" your entire drawing in Push mode. The tool will even go into groups and act on individual paths inside groups. If you're trying to use it without selecting something, it will remind you to select some objects first with a status bar message.

NOTE *The Tweak tool won't correctly work on open paths—an open path becomes closed as soon as you tweak it.*

12.6.1 Width and Force

In any of the Tweak tool's modes, what you work with is a circular *brush* (the orange-edged circle centered on the cursor) with which you "paint" over selected objects to change them. The Width parameter controls the brush size, and the amount of action it applies depends on the Force parameter as well as on pen pressure if you have a pressure-sensitive tablet. See 6.10 for more details on those parameters.

It takes practice to apply just the right amount of drag at the right place, with the proper force (including pen pressure), and with the correct brush size to get the result you want. However, this skill is very rewarding—what used to be awkward and time-consuming with the Node tool is often much quicker and more natural with the Tweak tool.

12.6.2 Fidelity

Any tweaking of a path slightly distorts—or rather *simplifies*, just like the Simplify command—the entire path, including even those parts you didn't touch with the brush.

The Fidelity value allows you to control the amount of this parasitic simplification. The tradeoff here is the number of nodes in the resulting path. With low fidelity, the resulting path will be node-poor but probably distorted more than you would find acceptable. High fidelity minimizes distortions, but the path may end up having a lot of nodes—which inflates the SVG size and slows down Inkscape.

The best value for fidelity depends on the nature of your artwork. If you're sculpting amorphous blobs, you can get away with a low fidelity of about 20. If, however, you are pushing or growing a text string converted to path and want the letters outside the distorted area to remain as clean and legible as possible, you will need to raise the fidelity to 80 or more.

Under the hood, the Tweak tool recasts the path into an "approximating polygon" with thousands of tiny straight line sides, tweaks the vertices of that polygon, and then recasts it back into a path with Bézier segments. The fidelity level controls the precision of this transformation and thus the number of nodes in the result, but no fidelity setting will give you the exact nodes of the original—some of the nodes will always end up displaced and changed, just like they do after a Simplify.

12.6.3 Push Mode

Push is the Tweak tool's default mode. To switch to Push mode from any other mode, press Shift-P, or click its button on the controls bar.

Push is a general-purpose sculpting mode. When you drag in this mode, the parts of the selected paths covered by the brush are shifted in the direction of your drag, for as long as the mouse button or pen is held down, as shown in Figure 12-34.

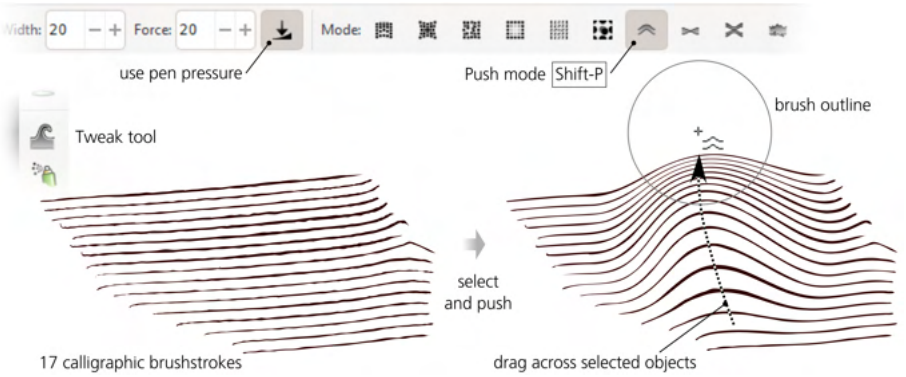


Figure 12-34: The Tweak tool's Push mode

Varying brush width as necessary, you can transmogrify by pushing any path into almost any other—but you can also use it for small tweaks, such as flattening a bump, bending an appendage, or curving an engraving grid. Thanks to the brush's bell-like profile, the paths you're pushing respond by curving softly and smoothly, as Figure 12-35 demonstrates.

Figure 12-35: Sculpting paths in Push mode

12.6.4 Shrink/Grow Mode

The Shrink/Grow mode (Shift-S) moves each point of a path in a direction perpendicular to the path's edge in that point, either inward (shrink, plain drag) or outward (grow, drag with Shift). To quickly access this mode from any other mode of the tool, Ctrl-drag to shrink and Shift-Ctrl-drag to grow.

The Shrink/Grow mode is very similar to the Inset and Outset commands (12.4), except that the Tweak tool, as always, acts softly on a part of a path instead of the whole path. You can use this mode to lighten or darken parts of drawings with of many details—such as engravings, freehand scribbles, or even text (converted to paths), as shown in Figure 12-36.

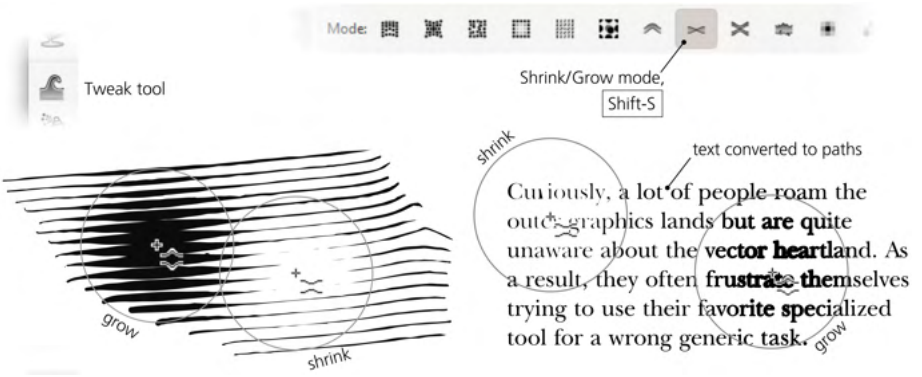


Figure 12-36: The Shrink/Grow mode of the Tweak tool

The Shrink/Grow mode allows you to easily Shift-drag to chase the moving edge of the path: this way, using a small-sized brush, you can grow appendages and branches of any length out of a path. (You can also do this in Push mode, but Grow is somewhat easier for this task.) Also, Shrink mode can act as a quick eraser—cutting through paths, evaporating small crumbs and bits, and trimming long brush strokes is easy and natural, as Figure 12-37 demonstrates.

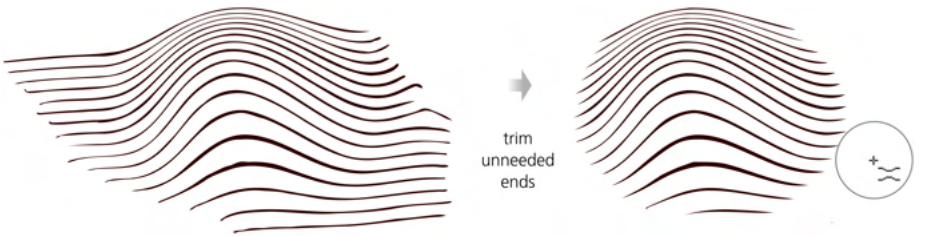


Figure 12-37: Using Shrink mode as an eraser

12.6.5 Attract/Repel Mode

The Attract/Repel mode (Shift-A) works by moving each affected point on a path toward (attract, plain drag) or from (repel, drag with Shift) the cursor point, pinching and exploding whatever paths fall under the brush. Sometimes, this may look similar to Shrink/Grow, but the difference is that Attract/Repel doesn't care about the direction of the path being tweaked; this mode moves everything symmetrically around the center of the brush (Figure 12-38).

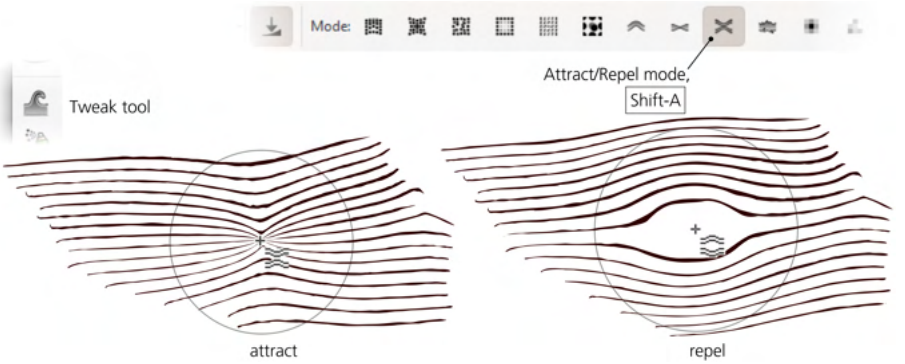


Figure 12-38: The Tweak tool's Attract/Repel mode

12.6.6 Roughen Mode

The Roughen mode (Shift-R) randomly distorts the edge of the path where you apply your brush, without changing the path's overall shape, as shown in Figure 12-39. Slight roughening simply makes the edge crooked and uneven, while strong roughening tears and explodes the edge into random blobs and splotches.

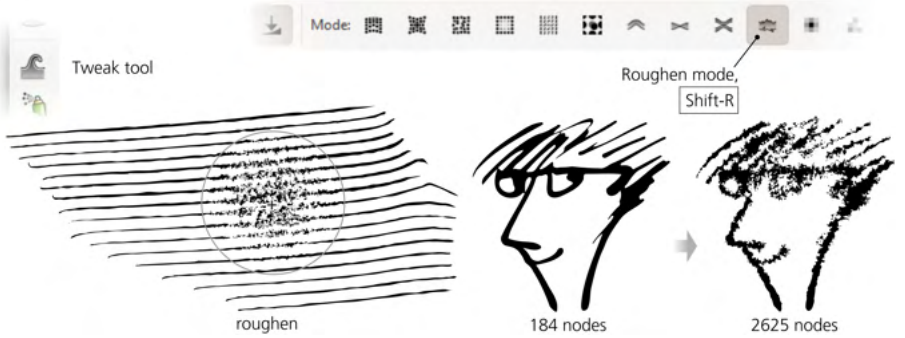


Figure 12-39: The Tweak tool's Roughen mode

This operation, especially with high fidelity, adds *a lot* of nodes. Such a roughened path may become hard to edit—it's awkward to handle with the Node tool and may be painfully slow with the Tweak tool. I recommend finalizing the overall shape of a path using pushing, growing, and shrinking as needed—and roughen it, if necessary, only as the final step.

13

PATH EFFECTS

Inkscape has three main extensibility mechanisms: filters, path effects, and extensions. If you're a developer eager to add a cool new feature that doesn't need to be in the core of the vector editor, you should look into one of these. Filters (Chapter 17), rooted in bitmap processing software, are mostly for altering colors and textures; in this chapter, continuing the topic of shapes and paths, I describe the interactive path effects that add a whole new level of complexity—and excitement—to plain SVG paths. Extensions, which are the most universal (but least interactive) way to provide new functionality for Inkscape users, are covered in Chapter 19, although we'll look at path-processing extensions in 13.4.

13.1 How Path Effects Work

Live path effects (LPEs), or simply *path effects*, are an easy-to-use (for the end user) yet powerful (for the developer) mechanism for modifying the visible shape of a path—for example, rounding the corners, roughening a smooth path, or distorting it by blowing or pinching. When you apply a path effect (or multiple path effects) to a path, the *original* before-the-effect path is still there—you can

always view and edit it. Every time you edit the original path, the *visible* after-the-effect path is recalculated from that original path and the effect parameters. That's what the *live* in "live path effects" means!

Inkscape's path effects are another application of the basic principle of vector graphics: instead of making some permanent and destructive change, leave the original object unchanged and record the way the change is to be applied to it. After that, both the original object and the parameters of the effect can be edited separately at any time.

Despite the name, path effects apply not only to paths but also to shapes (Chapter 11)—which remain shapes and are still editable as such, using the handles or numeric parameters in their shape tools (Figure 13-1). A path effect can also be applied to a group, which gives the same result as if the effect were applied to all the paths and shapes in the group. Path effects never apply to text objects, clones, or bitmaps.

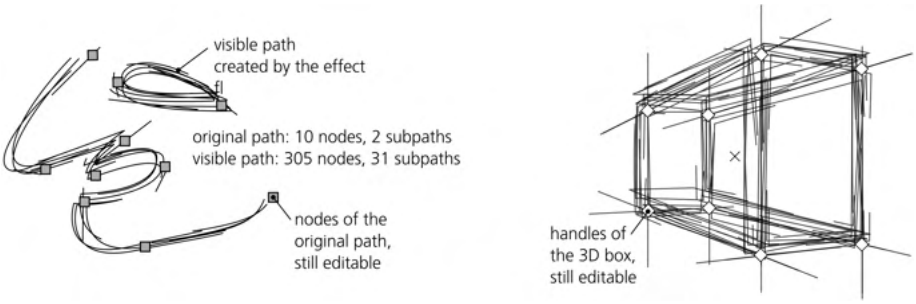


Figure 13-1: The Sketch effect (13.3.4.1) applied to a path (left) and to a 3D box (right)

When a path effect is applied to an object, the only aspect that changes is the *visible shape* of the object; if you want to change its *style* in a nondestructive way, try filters instead (Chapter 17). Path effects can be stacked on top of one another, so that the output of one effect is the input for the next one.

Path effects are an Inkscape-only feature; unlike, for example, filters, path effects are not part of the SVG standard. However, they are implemented in an SVG-compatible way. In other words, if you load an SVG file that uses a path effect into any SVG viewer (or into an old version of Inkscape that did not support this effect), you will see the same *visible* path that you see in latest Inkscape—just without access to the *original* path and the effect parameters.

PATH EFFECTS IN SVG

In a `path` element with a path effect applied, the original path data is stored in the `inkscape:original-d` attribute. The effect is given by the `inkscape:path-effect` attribute that refers to a specific `inkscape:path-effect` element in the `defs` of the document (A.4). The automatically calculated result of the effect is stored in the standard `d` attribute of the path object. All Inkscape tools and commands know that in a path with a path effect applied, they should ignore the `d` but work on the `inkscape:original-d` instead.

Unfortunately, *display* compatibility (your files are guaranteed to look the same in all SVG viewers) does not imply *editing* compatibility. If you edit an SVG document with path effects in another editor or in an old Inkscape version, you are editing the *visible* path, ignoring the effect. If you then reload the edited file into a version of Inkscape that does support the effect and try to edit it there, your changes will be lost because the new visible path will be regenerated from the unchanged original path and the effect parameters. What's worse, the list of path effects and their interpretations is not standardized; there is no guarantee that future versions of Inkscape will implement all the effects you're using in exactly the same way.

Despite these potential drawbacks, path effects remain a crucial Inkscape technology that, since its inception, has proliferated all across the application. Under the hood, many commands and features are implemented as path effects. Examples include pressure-sensitive (14.1.2.2) and shaped (14.1.5) strokes in the Pen and Pencil tools, the Spiro and BSpline modes in those tools (14.1.4), as well as Power Clip and Power Mask (13.3.12).

13.2 Managing Path Effects

Inkscape comes with a number of sample SVG files (in the *inkscape/examples* folder under the Inkscape data folder that you can look up in Preferences, System page). A few of these sample files (with names starting with *live-path-effects*) demonstrate and explain some Inkscape path effects. Let's load and explore one of them to get a quick idea of how a path effect may look and behave.

You will notice that a path object that has one or more path effects applied says so in the status bar when you select it—for example, *Path 328 nodes, path effect: Pattern Along Path, Roughen in layer Background*. You can use a copy/paste trick to reuse the same effect (or stack of effects) on any number of paths: copy the source object (Ctrl-C), select other path (or shape) object(s), and use the **Path ▸ Paste Path Effect** command (Ctrl-7).

To clear away the path effect and return to the original path, use the **Remove Path Effect** command in the **Path** menu. If, however, you want to preserve the result of the effect and forget the original path (this is sometimes called "flattening" or "baking in" the effect), use the **Object to Path** command (Shift-Ctrl-C); after it, your path will look exactly the same but the effect will be gone.

NOTE *The Object to Path command is a way to ensure that your file is not only correctly rendered but also editable in older versions of Inkscape and in SVG editors other than Inkscape.*

Boolean operations (12.2) on paths with path effects "flatten" them as well—that is, the result is the same as if you used Object to Path on all paths beforehand (however, there is a path effect reimplementation of Boolean operations that is nondestructive, 13.3.10). Unfortunately, this is also the case for the Combine and Break Apart operations (12.1.1), even though many effects work on subpaths, so a way to manipulate subpaths without losing the effect would be very useful. (In fact, that's how Combine and Break Apart used to work in previous versions of Inkscape.) A workaround is to group separate paths together and apply the

effects to the group. After that, you can enter the group (4.9.1) and combine or break apart the paths inside as needed, without losing the effect on the group.

The Node tool (12.5) edits the *original* path, not the *visible* path after the effect. Since the original path you're editing is not visible by itself, the Node tool can optionally highlight it with red if you turn on the Show path outline toolbar button (Figure 12-17 on page 234). Also, as you will see below, many effects have on-canvas editable handles that you can drag using the Node tool.

13.2.1 The Path Effect Editor Dialog

The Path Effect Editor dialog (Shift-Ctrl-7) is Inkscape's main control hub for path effects (Figure 13-2). This is where you apply individual effects to the selected object as well as edit parameters of applied effects. When an object with one or more path effects is selected, this is where you manage its stack of effects—you can add, rearrange, or remove effects in the stack.

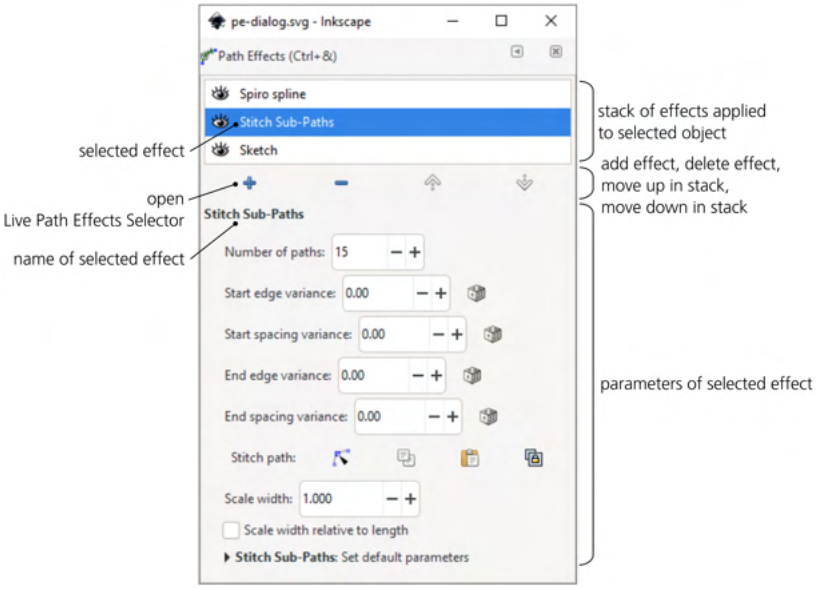


Figure 13-2: The Path Effect Editor dialog: a path with three effects is selected (Spiro spline, Stitch Sub-Paths, and Sketch).

The dialog works only when you have a single path, shape, or group selected—that is, you cannot edit effects on multiple paths simultaneously. Once again, the fact that path effects can apply to groups provides a workaround: if there are many paths or shapes that you want to have the same effect(s) with the same parameters, simply group them together, select the group, and use this dialog to apply the effect(s) to the group. After that, you will be able to access or edit these effects only if you select the parent group, not any of the individual paths inside the group (remember that, for example, the Node tool always selects paths ignoring grouping).

The list at the top of the dialog shows all the effects applied to the selected object. The effects are listed top to bottom—that is, the topmost effect in the list is the first to be applied to the source path. Its output is passed as input to the second one, and so on, until the last listed effect's output is displayed as the final result.

Any new effect that you add (using the plus-sign button under the list) is placed at the end of the stack. You can move any effect in the stack up or down with the arrow buttons. Clicking the eye icon before each effect's name disables that effect, forcing Inkscape to bypass it. To delete an effect from the stack, use the button with the minus sign.

The panel below the list is where you edit the parameters of the effect selected in the list. These may be run-of-the-mill checkboxes or drop-down lists, but some effect parameters belong to more interesting types:

- *Numbers* can be either integers or fractional, depending on the nature of the parameter. When a number denotes a distance, there is usually a unit selector as well.
- If an effect uses a random number, the corresponding *random number parameter* gives the range in which the random values must fall (see Figure 13-3). The dice button to the right re-randomizes (technically, *reseeds*) the random values controlled by this parameter.

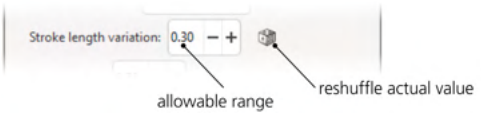


Figure 13-3: A random number parameter of a path effect

- *Link parameters* are used when one path's effect uses some other path as one of its parameters, as shown in Figure 13-4. That *linked path* can be a separate object located somewhere on the canvas (in the same document), or it can be a path stored entirely within the path effect and not visible in the document.

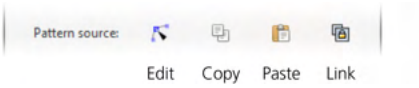


Figure 13-4: A link parameter of a path effect

Each link parameter displays a row of four buttons:

Edit
: Switches Inkscape to the Node tool and lets you edit that linked parameter path—whether it is a separate object or a path stored inside the effect (Figure 13-5). The parameter path is shown as a dark green outline.

  This is the same as if you manually switch to the Node tool and click the Show next editable path effect parameter button on the controls bar (see Figure 12-17 on page 234) to navigate to this specific linked parameter path (as there may be several, depending on the effect).

Figure 13-5: Editing a path effect's linked path

Copy
: Copies the linked path to the clipboard.

Paste
: Pastes the path from the clipboard into the effect, making a copy of the clipboard path and storing it in the effect.

Link
: Takes the path copied to the clipboard and links the effect to its original in the document. Now, editing the path you had copied will change this path's effect.

Some effects can only *link* to another path in the document, not store it inside, so they don't have the Copy and Paste buttons (for example, the Clone original effect, 13.3.5.1).

Apart from the numeric controls in the dialog, some path effects allow you to edit their parameters visually by on-canvas handles accessible in the Node tool and the shape tools (you'll see examples of this as I discuss specific effects). These handles are often white and diamond-shaped, but they can be of any color or shape so long as they do not get confused with the path nodes and node handles that are also editable in that tool. Hovering your mouse over a path effect handle will usually show some useful hint in the status bar.

At the bottom of the parameters panel, there is a folded section titled Set default parameters. In it, you can make the current value of any of the effect's parameters its default value. The default value will then be used when you apply this specific effect to new objects.

13.3 A Guide to Inkscape Path Effects

Path effects remain one of the active growth areas in Inkscape development, and the list of effects supported by the program expands with each new version. Unfortunately, this also means that path effects are a "work in progress" more so than the rest of Inkscape. Some of the effects are marked "experimental," but even among the non-experimental ones, not all effects are polished and reliable enough for a more than occasional use.

This section lists and explains, with examples, the most useful and usable path effects in the latest version of Inkscape, grouped by topic. Figure 13-6 shows what you see when you click the plus-sign button in the Path Effect Editor dialog to add a new effect.

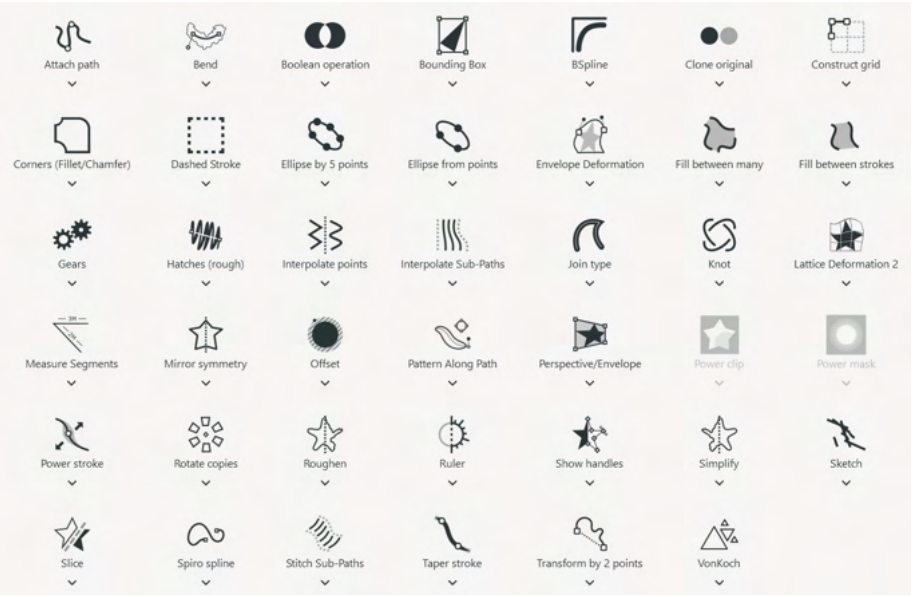


Figure 13-6: All path effects in Inkscape (excluding the experimental ones)

13.3.1 Stroke Shaping Effects

Historically, what later evolved into Inkscape path effects was first inspired by the desire to have paths stroked by something more interesting and expressive than the standard same-width SVG stroke—to have editable *vector brushes*. Now, Inkscape offers two different approaches to this, described in this and the following sections; each approach has at least two implementations as distinct path effects.

13.3.1.1 Power Stroke

Imagine that your stroked path has additional handles along its length, and you can drag any of them laterally to make the path wider or narrower in this area. That's exactly what the Power stroke effect offers: a variable-width stroke whose profile you can adjust interactively by dragging onscreen handles (in the Node tool). Just as with a regular stroke, you can also adjust cap (9.3) and join types (9.2), as well as play with various interpolation and smoothness settings.

Power stroke is what the Pencil tool applies to paths it creates in the pressure-sensitive mode (14.1.2.2). However, as with any other path effect, you can also assign it to any path manually. Just select a path, click the plus-sign button in the **Path Effect Editor** dialog, and select **Power stroke** in the list (Figure 13-6). This creates a power stroke with (initially) three purple-colored Power stroke handles along the path. Switch to the Node tool and drag any of them to see how the path responds (Figure 13-7).

Of the parameters of the effect, the most useful are the Width factor, which uniformly scales the stroke width at all handles, and the Cap type that you can select for the start and end nodes separately. The default Zero width cap smoothly tapers out the stroke from the first/last purple handle to the end of

the path, whereas the Butt, Round, and Peak caps keep it the same width from the ultimate handle to the end.

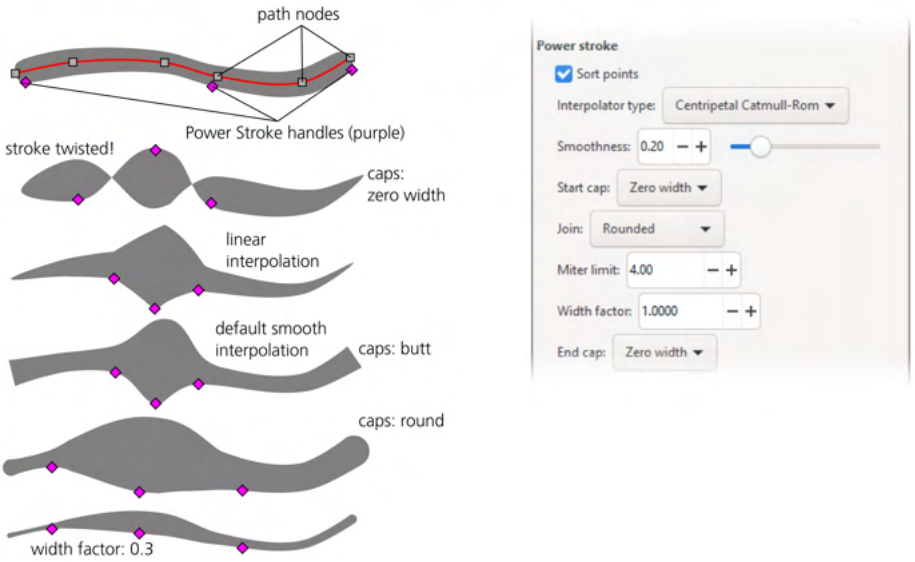


Figure 13-7: The Power stroke effect

If you need more purple handles to shape your stroke, Ctrl-click any of them and then drag away the new handle that gets created in the same spot. To delete a handle, Ctrl-Alt-click it.

13.3.1.2 Taper Stroke

A simpler variation of Power stroke is the Taper stroke effect shown in Figure 13-8. Instead of arbitrary shaping, all it does is taper off the ends of a stroke—but this is good enough for many use cases. This effect displays two round handles that can move only along the path's centerline, indicating the points where the path begins to narrow down. The Taper smoothing parameter lets you vary the tapering from triangular (0) to smoothly elliptic (1).

NOTE *Assigning Power stroke or Taper stroke to a plain stroked path removes stroke paint but adds fill paint of the same color. That's because from the SVG viewpoint, such a shaped stroke is a filled path; if you want to change its color, assign it to fill, not stroke. To preserve the original path's fill, Inkscape adds a fill-between path linked to the original path (13.3.5.5).*

The biggest problem with both Taper stroke and Power stroke is that they are tied to the positions of the nodes in a path, instead of the distance along that path. For instance, the numeric start/end offsets in Taper stroke are measured in the units of the first/last internode distance—so that when Start offset is 1, the start tapering handle coincides with the *second* node on the path. Many path operations, such as Simplify (12.3), do not preserve node positions, which makes Taper stroke and Power stroke behave unpredictably.

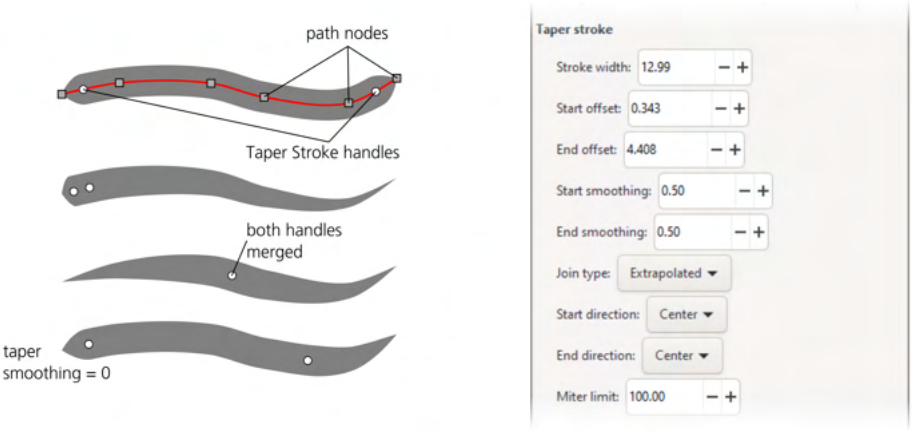


Figure 13-8: The Taper stroke effect

13.3.2 Path-Bending Effects

The Pattern Along Path and Bend effects implement a different approach to editable vector brushes (Figure 13-9). They both take one path (called the *pattern*) and bend and/or stretch it along another path (called the *skeleton*). As usual with path effects, both the skeleton path and the pattern remain editable at any time, with the result updated live.

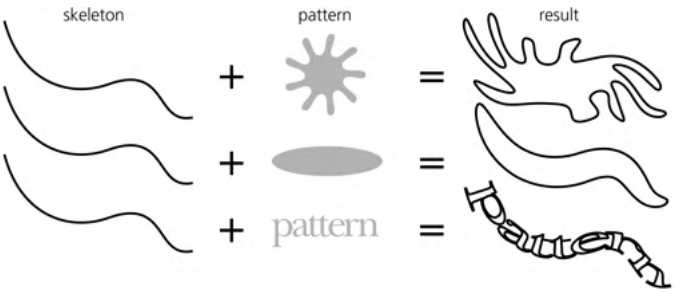


Figure 13-9: Bending a pattern along a skeleton

This is also used by the Pencil tool: instead of a Power stroke controlled by a pressure-sensitive tablet pen, you can assign a predefined fixed shape to your Pencil strokes (see Figure 14-12 on page 302 for an example). You can also take an existing drawing made with plain SVG strokes and try applying various patterns to its paths.

13.3.2.1 Pattern Along Path

The main difference between the Pattern Along Path and Bend effects is which path is the skeleton and which is the pattern. In Pattern Along Path, the path you're applying the effect to is the skeleton, and the pattern is linked up as a link parameter.

This effect is ideal for simple, possibly repeated, patterns applied to arbitrarily complex skeletons. The linked pattern path can be either an independent path object in the document or a path stored inside the effect itself. The result gets the style of the skeleton. This is the effect used by the Shape option in the Pen and Pencil tools (14.1.5).

Once you apply Pattern Along Path to a skeleton path, you need to supply the pattern using the Pattern source link parameter. The Edit button does not work unless you paste or link some pattern path first, so the usual sequence of operations is this: select a pattern path, copy it (Ctrl-C), select a skeleton path, assign **Pattern Along Path** to it, and paste or link the pattern to it, as shown in Figure 13-10. You can also draw paths with the copied pattern applied to them automatically if you choose From clipboard in the Shape list in the Pen or Pencil tools.

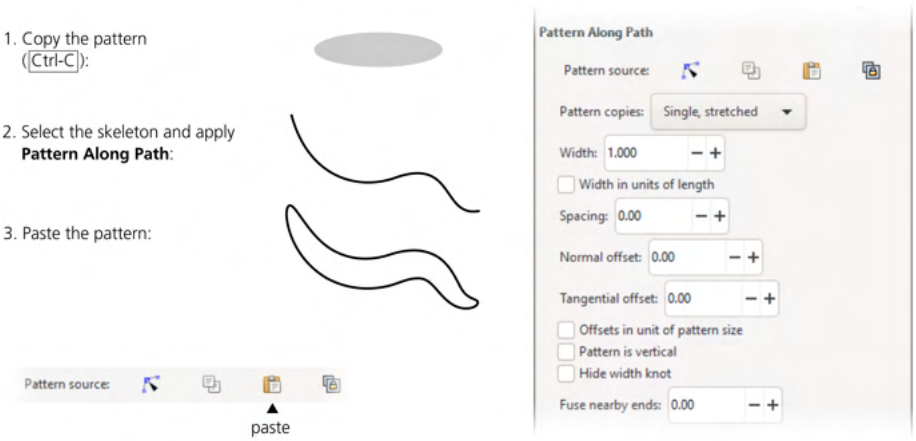


Figure 13-10: The Pattern Along Path effect

The Pattern Along Path effect can use one of the following *repeat modes*:

Single
: Places a single copy of the pattern along the skeleton, from the start node, without stretching it. If the pattern is shorter than the skeleton, it will only cover part of the skeleton's length; if the pattern is so long that it does not fit even once, it will not be applied.

Single, stretched (default)
: Also places a single copy of the pattern along the skeleton, but always stretches or squeezes it so it exactly fits the skeleton's length. The Bend effect always uses this mode; unlike Pattern Along Path, in Bend the repeat mode is not changeable.

Repeated
: Places as many copies of the pattern as will fit along the skeleton, but doesn't stretch them, so the remainder of the skeleton (less than one pattern length) remains unfilled. (This does not mean that the copies of the pattern are identical; the curvature of the skeleton may noticeably distort them, as Figure 13-11 demonstrates.)

Repeated, stretched
: Places as many copies of the pattern along the skeleton as would fit and stretches them evenly, so that they exactly fill the entire skeleton length.

The pattern always starts from the start of the path; if you want it to go the other way, use **Path ▸ Reverse**.

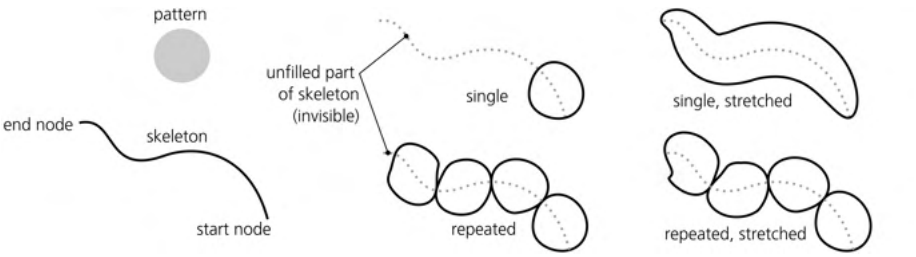


Figure 13-11: Repeat modes of Pattern Along Path

The Pattern Along Path effect also allows you to adjust some *distance parameters*:

Spacing (only for repeated modes)
: Sets the spacing between copies of the pattern on the path.

Normal offset
: Moves all copies of the pattern *perpendicular* to the skeleton path at each point.

Tangential offset
: Moves all copies of the pattern *along* the skeleton path, so that the first pattern starts not at the beginning of the skeleton but at this specified distance from it.

These offsets and spacing parameters are, by default, in absolute px units. By checking the Offset in units of pattern size checkbox, you can express them as multipliers of the pattern size—for example, a tangential offset of 0.5 will shift the pattern along the skeleton by half the pattern's width, as Figure 13-12 demonstrates.

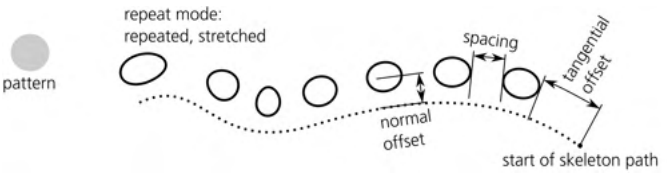


Figure 13-12: Spacing and offsets in Pattern Along Path

By default, the original pattern is considered to be horizontal—that is, the pattern is aligned on the skeleton by the pattern's horizontal axis. By checking Pattern is vertical (for Pattern Along Path) or Original path is vertical (for Bend), you can rotate the pattern by 90 degrees so it aligns its vertical axis to the path (see Figure 13-13).

Figure 13-13: Orientation of pattern in Pattern Along Path

Both effects let you change the width of the pattern. The Width parameter can be measured either in the units of the original width of the pattern or in the units of the skeleton length (Width in units of length). In both effects, you can also adjust width by an onscreen handle that is attached to one end of the skeleton path by a perpendicular segment (Figure 13-14).

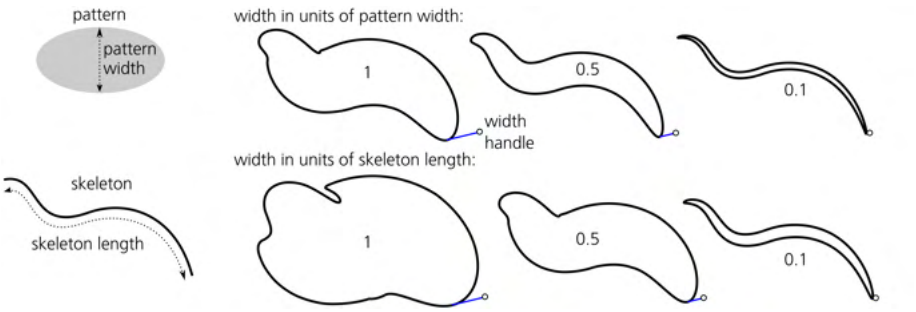


Figure 13-14: Adjusting the width of a pattern in Pattern Along Path

13.3.2.2 Bend

In Bend, the path you're applying the effect to is the pattern, whereas the skeleton is linked up using a link parameter. This is more convenient when you have a complex pattern that you want to curve lightly along a simple skeleton path that can be shared by several bent paths. Similarly, the linked skeleton path can be either an independent path in the document or a path stored inside the effect itself. The result gets the style of the pattern.

For this effect, you start with the pattern, apply the effect, and use the Bend link parameter to link to a skeleton. Unlike Pattern Along Path, however, Bend provides a default two-node skeleton path that stretches along the horizontal axis of your pattern—so you can at once use the Edit button to edit that skeleton (Figure 13-15). Or you can paste a skeleton (which may itself have some path effect applied to it) from the clipboard, or link to a copied path in the clipboard.

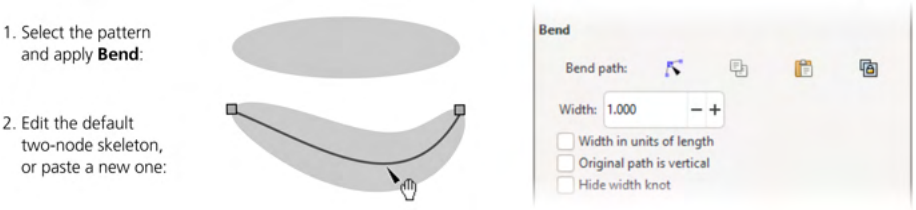


Figure 13-15: The Bend effect

13.3.3 Deformation Effects

Effects in this group create an editable external frame on your path, and you can then edit that frame to deform or distort the path in various ways. You will need this, for example, when inscribing a complex shape into a 3D scene (perspective distortion) or sticking a label onto a curvilinear bottle (lattice or envelope deformation). As with all other path effects, these deformations will work on a group as well as on an individual path, and the group can contain all kinds of differently styled objects (even though any non-paths and non-shapes in that group, such as bitmaps or text not converted to paths, will *not* be distorted by the effect).

13.3.3.1 Envelope

The simplest effect in this group, Envelope Deformation, creates a rectangular envelope whose four sides can each be edited as Bézier curves. The effect treats the sides as four individual helper paths ("bend paths"), which is slightly inconvenient: you need to click the Edit buttons for each side in turn to curve all four sides. The Copy and Paste buttons allow you to transfer the shape of the envelope from one object to another, but this also requires four separate copy/paste operations. On the other hand, the four sides being separate means they need not necessarily touch each other, which allows for some additional distortive flexibility, as shown in Figure 13-16.

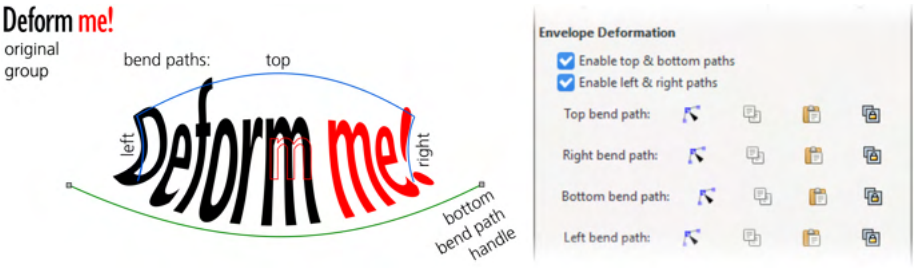


Figure 13-16: Envelope distortion on a group of paths (bottom bend path activated); the red outline (on mouse hover in the Node tool) shows the original undistorted path of one of the letters.

13.3.3.2 Lattice

The Lattice Deformation effect is a more complex variation of the same idea. It creates a five-by-five lattice of control points that you can drag around individually (also in the Node tool), which allows for more sophisticated and detailed distortions (Figure 13-17).

Handling that many points manually may be tricky, so the effect offers some ways to simplify the workflow: the Mirror movements in horizontal/vertical parameters make the lattice symmetric so you need to shape only one half of it, and Use only perimeter removes the inner points altogether, limiting the lattice to the 16 points on its perimeter. The Reset grid button lets you clear any botched edits of the lattice and start over.

Figure 13-17: Lattice distortion on a group of paths; the red outline (on mouse hover in the Node tool) shows the original undistorted path of one of the letters.

13.3.3.3 Perspective

Finally, the Perspective Transformation effect is specialized for 3D-like transformations of paths—for example, this is what you would use to place a word of text on the side of a 3D box (11.3). Here, the envelope has only four corner nodes and its sides cannot be curved, as Figure 13-18 demonstrates. The effect's parameters allow you to set the coordinates of the four points numerically or make the envelope symmetric.

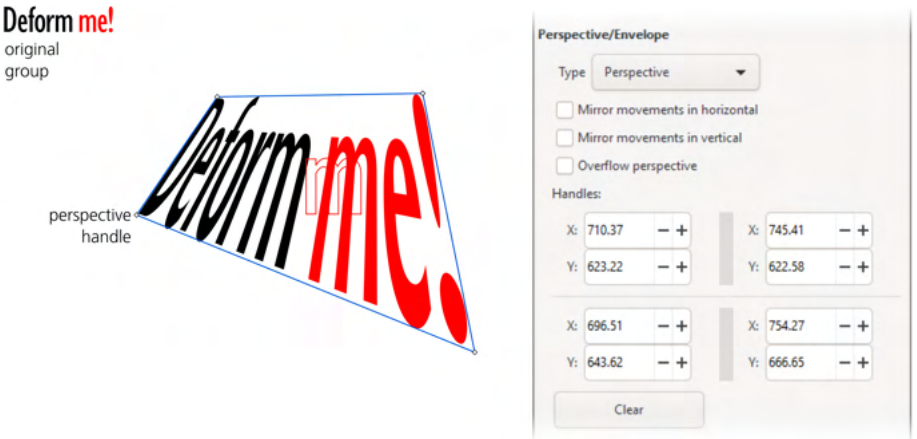


Figure 13-18: Perspective distortion on a group of paths; the red outline (on mouse hover in Node tool) shows the original undistorted path of one of the letters.

13.3.4 Artistic Effects

Effects in this group strive to make objects look less geometric and more organic or freehand-drawn. Usually, this involves an element of randomness.

13.3.4.1 Sketch

Sketch is an artistic effect that turns a path into a sketch-like drawing made of multiple strokes, as if drafted on paper by an artist trying to grope for the perfect shape (Figure 13-19).

To make sense of this complex effect's parameters, note that the sketch consists of two types of artifacts: approximating strokes and construction lines. The *approximating strokes* cover the entire path; they are typically curvilinear,

more or less parallel to the original path (with certain tremor), and travel at some distance from it. The *construction lines*, on the other hand, identify and emphasize the straight or almost straight segments of the path by drawing straight lines that extend each such segment on both sides.

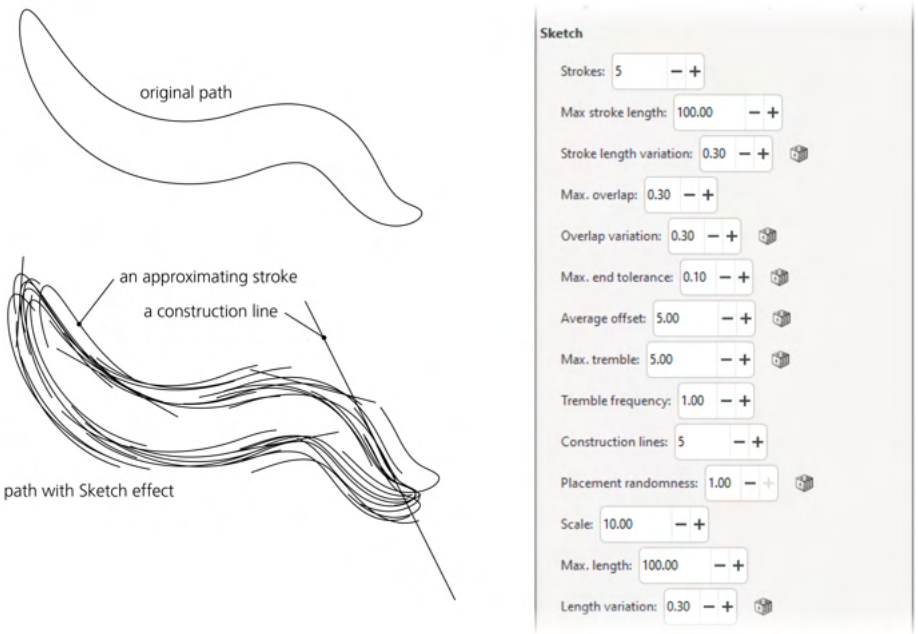


Figure 13-19: The Sketch effect

For the approximating strokes, you can change the following:

- The average *number* of parallel strokes at each point of the path (default is five). Set this parameter to 0 to hide approximating strokes (leaving only construction lines). Low values make the sketch airy and tentative; increasing the number makes it bolder and noisier, as shown in Figure 13-20.

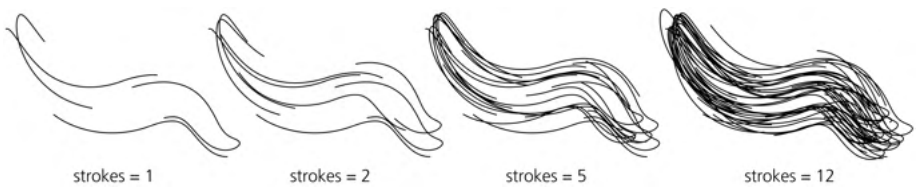


Figure 13-20: Changing the number of approximating strokes (construction lines are off)

- The maximum *length* of strokes (in px units) and the range of the random length variation (relative to the maximum length).
- The maximum *overlapping* of subsequent strokes (in px units) and the range of the random variation of this parameter (relative to the maximum overlap value).

- The *end tolerance*, which affects how close the approximating strokes follow the original path.
- The average *offset* of the approximating strokes from the original path; by varying this parameter, you can make the sketch either neat and tight or wide and ruffled.
- The maximum *tremble* and its *frequency*; these control how the strokes oscillate around the original path (Figure 13-21). Increasing the maximum tremble ruffles the sketch, similar to increasing offset but more randomly. Increasing the frequency makes the sketch lines look rougher by making them tremble on a smaller scale.

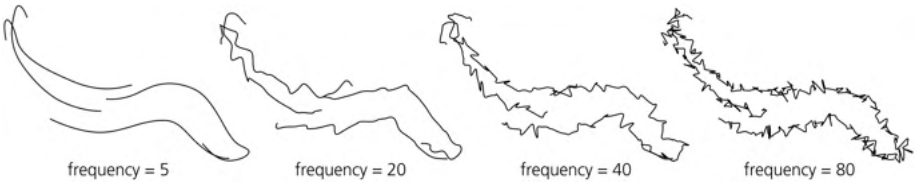


Figure 13-21: Changing the tremble frequency of approximating strokes

For the construction lines, you can change the following:

- The total (not average) *number* of lines in the sketch (the default is five). Set this to 0 to suppress construction lines, leaving only approximating strokes.
- The *scale* parameter, which tells how far the ends of the construction lines can go beyond the ends of the straight (or approximately straight) segments of the path.
- The maximum *length*, and its random *variation*, which set the upper limit on the length of construction lines (Figure 13-22).

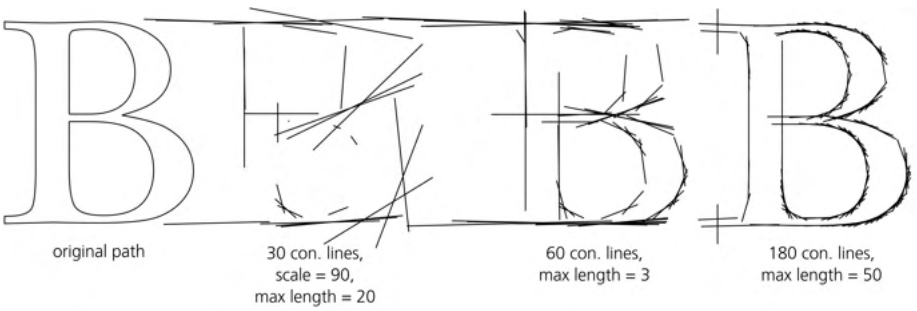


Figure 13-22: Playing with construction lines (approximating strokes are off)

13.3.4.2 Hatches

The Hatches effect, unlike everything else you've seen so far, focuses on the fill of the path, not its stroke. It creates a wiggly line that fills the path to create an impression of freehand artistic hatching.

The basic parameters of the hatching are controlled by onscreen handles, editable in the Node tool. There are two pairs of handles: the green pair controls the density (frequency) of the hatching and its slant, while the yellow pair bends the hatching lines into arcs. In each pair, the round handle is the base (which can be placed anywhere), and the diamond handle controls its parameter by its position relative to the base, as shown in Figure 13-23.

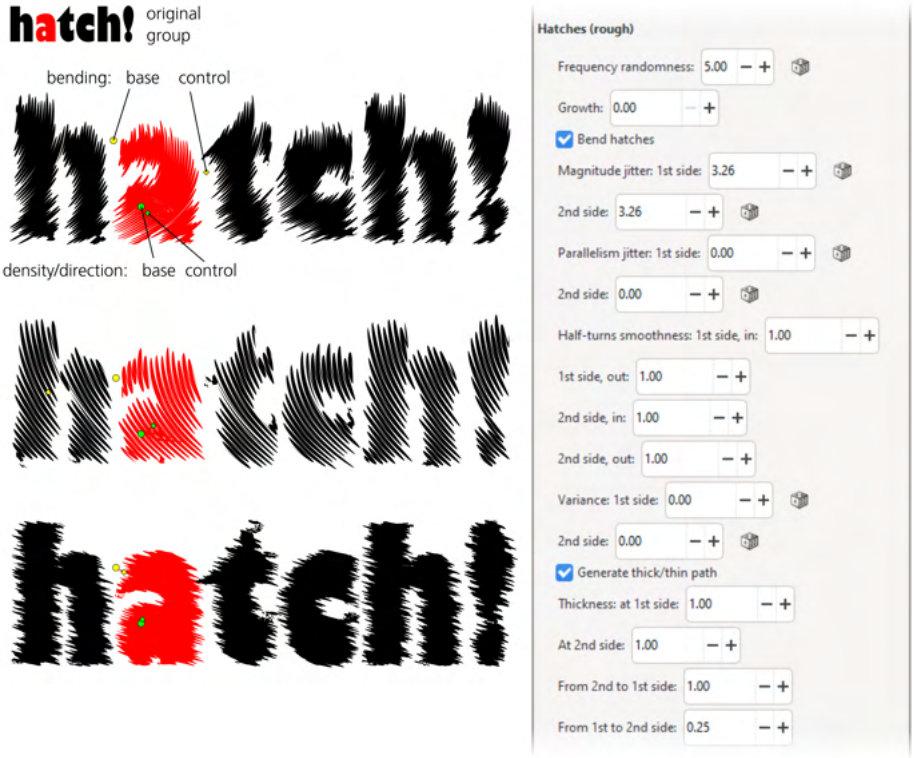


Figure 13-23: The Hatches effect and its control handles

The rest of the parameters of this complex effect control various aspects of the hatching shape. Frequency randomness controls how much random variation is added to the density of the strokes; the smaller this value, the more uniform is the density of the hatching.

The 1st side and 2nd side in the parameter labels refer to the two opposite series of bends of the hatching wiggle—for example, bottom and top, if hatching is oriented vertically. Thus, Magnitude jitter, 1st side controls how random the position of each bend is at the bottom of the hatching—that is, how closely the hatching tracks the edge of the original path at the bottom.

Parallelism jitter adds randomness to the direction of the wiggle strokes, and Half-turn smoothness allows you to vary the bends from sharp angles to circular arcs. Finally, Thickness values change how narrow or wide the hatching strokes are in various parts. Figure 13-24 shows some examples.

Figure 13-24: Playing with the Hatches effect's parameters

13.3.4.3 Roughen

The Roughen effect is the general-purpose noise component in the path effects ecosystem. It subdivides the Bézier curves of a path into shorter segments and randomly displaces the nodes and node handles (Figure 13-25).

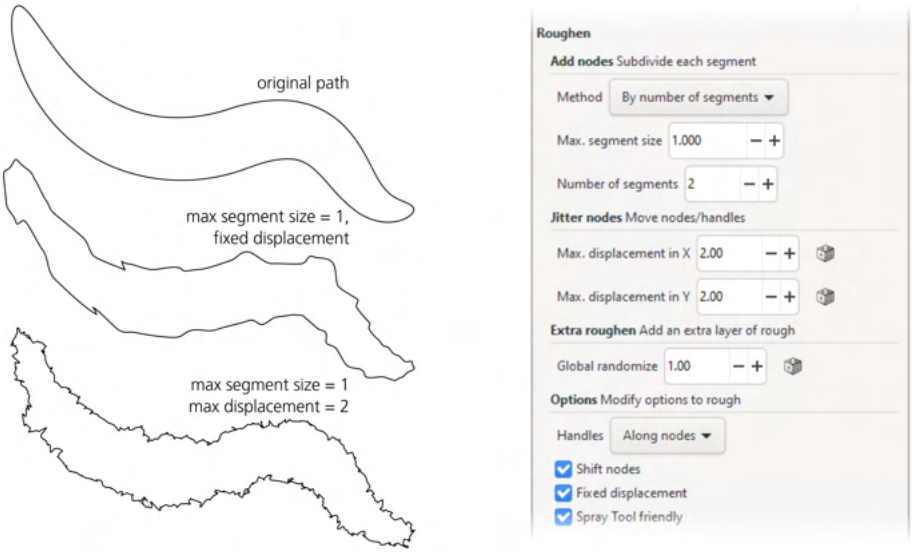


Figure 13-25: The Roughen effect

The subdividing is done in one of two ways: either specify the maximum segment size or give the number of segments into which to break each Bézier of the original path. The first approach (the default) is better in that it does not depend on the nodes of the original path and produces segments of approximately equal size.

Next, you get to choose by how much to jitter the nodes. Maximum displacement parameters limit this from above—but if you tick the Fixed displacement checkbox, it also limits the movement of the nodes by 1/3 of the (subdivided) segment length. That is, if you want to jitter nodes by 10 px, you need to put 10 into the Maximum displacement fields and also uncheck Fixed displacement. With Shift nodes unchecked, the effect only randomly rotates the handles of the Bézier segments but does not move the nodes themselves.

13.3.4.4 Simplify

You've already met Simplify (12.3) as a one-time destructive command. Now, meet its tamer, more cooperative, and fully reversible cousin: the Simplify path effect.

Simplifying is a powerful concept. You take a path and ask Inkscape to redraw it from scratch, without looking at the original nodes but trying to preserve the overall shape—with some simplification. The result may be more or less useful, but it rarely fails to amuse. It makes sense to group Simplify together with Roughen because they act in directly opposite ways—although, of course, simplifying a roughened path will not give you the exact original, but just a different (and subtler) kind of artistic distortion, as shown in Figure 13-26.

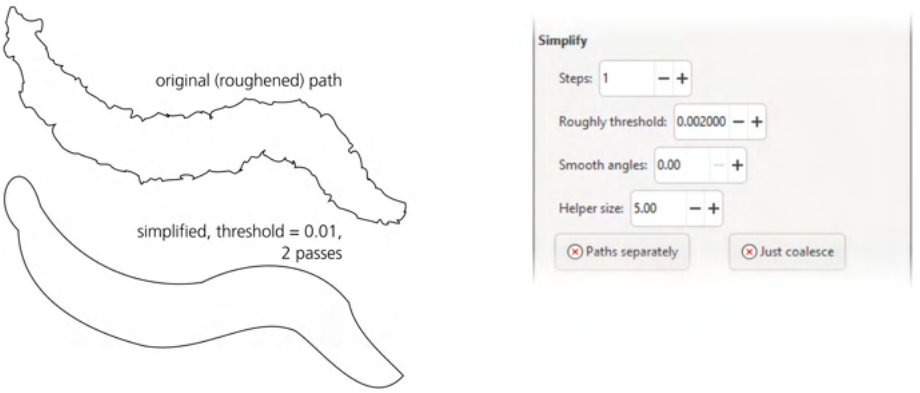


Figure 13-26: The Simplify effect

The effect's parameters worth fiddling with include the simplification threshold and the number of passes. (The Helper size is best set to 0; I could not find any use for the helper lines that this effect shows on canvas.)

Sometimes, it makes just as much sense to overlay Roughen on top of Simplify, so that Simplify creates a characteristic organic shape to which Roughen adds texture, as Figure 13-27 demonstrates.

Figure 13-27: Roughen on top of Simplify

13.3.5 Repeaters and Fractals

A number of effects, instead of distorting the source path, simply copy or multiply it in various useful ways.

13.3.5.1 Clone Original

This is a generic clone effect for copying one path's nodes (more precisely, *path data*, which includes positions and types of all nodes and handles) to another path. Unlike the standard SVG clones (Chapter 16), this effect lets you copy only the path data, with or without its path effects. The cloned path is still a path, so you can apply some other effects to it on top of the Clone original. For example, you can have an original path that uses Spiro spline and a number of its clones that have varying levels of Roughen or Simplify applied on top. This effect also allows you to specify precisely which attributes or CSS properties (8.1) the clone object gets from the original.

You can even link a path to a text object without converting it to a path. This way, the text remains editable as text, but you also have an automatically identical path with the Clone original effect that can have any other path effects stacked on top.

To use it, you first need to have a path object that will become a clone (its own path data will be discarded). Assign the **Clone original** effect to it. Then, select and copy (Ctrl-C) the path or text you want to clone. Then, select the path with the **Clone original** effect again and click the **Link to item** button in its parameters panel. (The Select original button next to it selects the object that this one clones from.) Then, choose one of the **Shape** options that control exactly what path data will be copied:

- With LPE's (default): the visible path with all its path effects.
- Without LPE's: the path before any path effects.
- Spiro or BSpline Only: the path with only its spline effects (13.3.7) applied, if any.
- No Shape: nothing is copied.

Finally, you can specify (as comma-separated lists) the attributes and CSS properties that the clone will get from the original path. A separate checkbox controls the copying of transformations (Chapter 6); if it is unchecked, the clone will always be positioned exactly over (or under, depending on z-order) the original.

[1.1] 13.3.5.2 Slice

The Slice effect cuts a path (or shape or group) into two halves with a straight cut. This "sawing a person in half" trick produces two separate paths that you can move, transform, and style independently (Figure 13-28). Of the two resulting paths, one (with the Slice effect applied) stores and lets you edit the entire path but shows only one half of it. The other path has no path effects of its own, but when you edit the first path, it updates automatically to reflect the changes. You can adjust the cut line's position and angle with the handles.

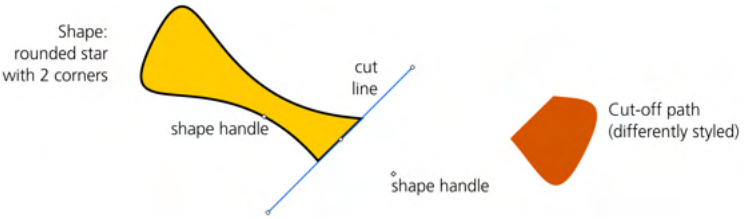


Figure 13-28: Slicing a shape

13.3.5.3 Mirror Symmetry

This effect takes the original path and creates a copy of it mirrored around an axis of symmetry that you can adjust interactively via onscreen handles in the Node tool (Figure 13-29). The length of the mirror axis is irrelevant; only its position and direction matter. Of the three handles on the mirror axis, the end ones rotate it while the central one moves it parallel to itself.

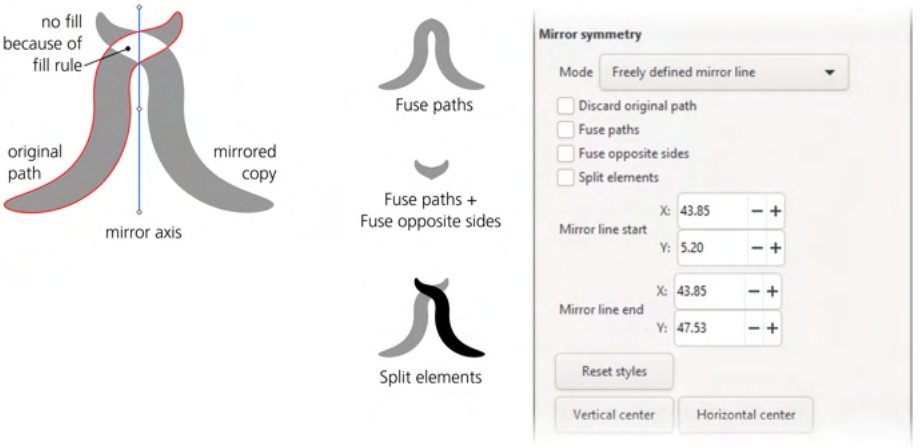


Figure 13-29: The Mirror symmetry effect

By default, the effect creates the copy as a subpath of the original path, so in the areas where the original and the copy overlap, the paint is determined by the fill rule (12.1.2). If you enable Fuse paths, the paths will never overlap because they will be clipped by the mirror axis; if you also enable Fuse opposite sides, clipping itself is mirrored, so you see only the parts of the path that go *beyond* the mirror axis (and their reflection). The Split elements option creates, instead of a subpath, a separate path element for the mirror copy, which can have a different style; you can manually transform or node-edit this mirror path—but if you then transform or node-edit the original path, the mirror copy object will synchronize, overriding your changes.

Even such a simple thing as mirror symmetry can produce amazingly deep patterns if applied, for example, to a large complex star, as Figure 13-30 demonstrates.

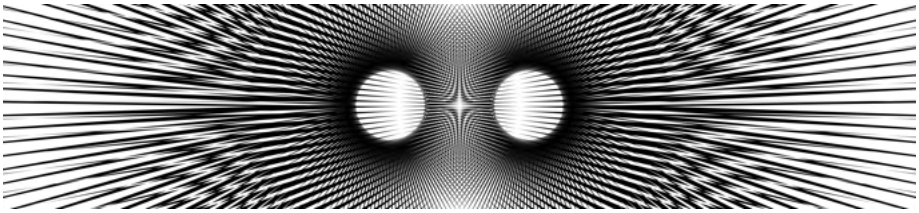


Figure 13-30: An owl's eyes: a mirrored 183-ray star

13.3.5.4 Rotate Copies

This is a close relative of the Mirror symmetry effect, except that it creates any number of copies placed on a circle whose center and starting angle you can adjust via onscreen handles in Node tool. Optionally, you can mirror every second copy. You can also enable the Split elements option that creates separate path elements, instead of subpaths, for each copy. Note that adjusting the starting angle rotates all copies of the path, including the original (even though the Node tool will still show it in its original place, if you enable the red outline), as shown in Figure 13-31.

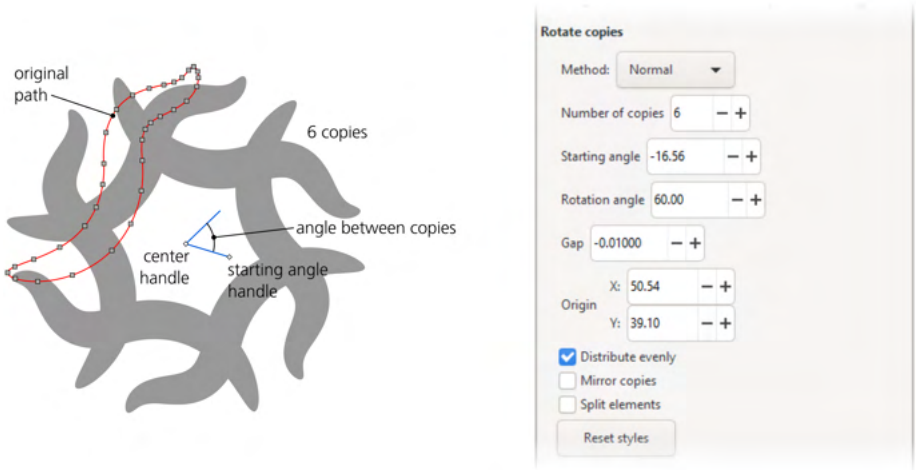


Figure 13-31: The Rotate copies effect

[1.1]

13.3.5.5 Fill Between Many

The Fill between many effect creates a new path linked to one or more original paths and covering the entire space between them—as if the ends of those paths were connected by straight line segments to form a single closed path. The original paths can have some path effects of their own (for example, path-shaping effects). Editing any of the linked paths updates the fill-between path automatically. This effect is so useful, it even has its own menu command: Path ▸ Fill between paths.

You can't node-edit or transform the fill-between path with the Selector, but you can style it with a fill and/or stroke style of its own. You can also select and delete it as a separate object without affecting the linked paths.

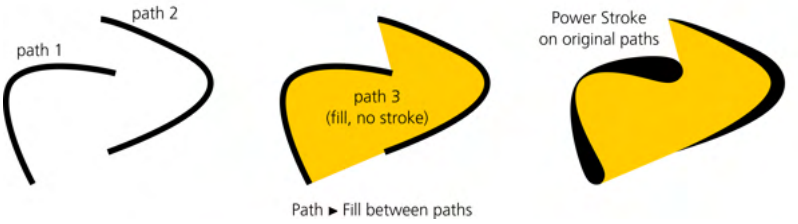


Figure 13-32: The Fill between many effect

Despite having the word *many* in its name, this effect can be just as useful with a single original path. When you assign Power stroke or Taper stroke to a path with fill (13.3.1.1), a fill-between path is also created that is linked to the original path, reproducing its original fill style (which otherwise would be lost). This way, you get a shaped stroke and the fill inside it, both updated live when you edit the path with the stroke-shaping effect.

13.3.6 VonKoch

This recursive effect takes the original path and repeats it twice (as subpaths) with shifting, scaling, and rotating; it then repeats the same operation on these copies, and so on for the specified number of *generations*. This is an example of a fractal—a self-similar shape that looks the same at different levels of zoom.

Three helper paths control the copies' transformations: a *reference segment* (initially, running horizontally across the source path) and two *generating paths* (initially, running horizontally across the two first-generation copies), as shown in Figure 13-33.

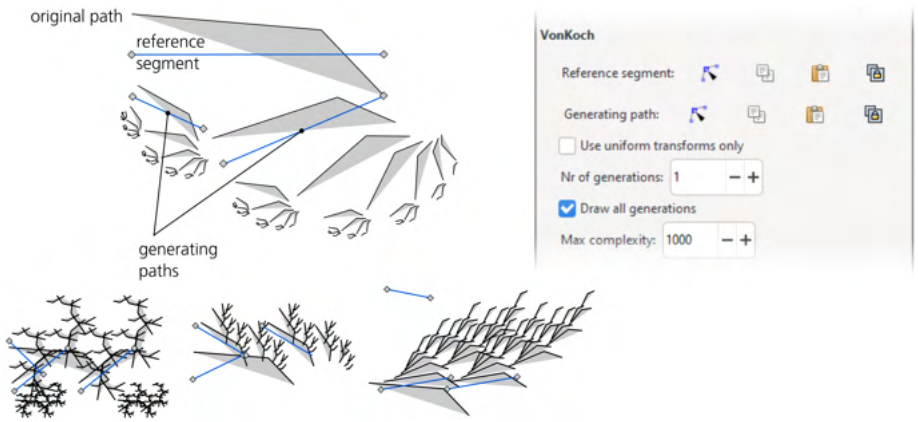


Figure 13-33: A number of VonKoch fractals for the same source path and different helper paths

These helper paths are two-node straight line segments; to edit them, use the Node tool and click the **Show next editable path effect parameter** button on its control bar once (for the reference segment) or twice (for the generating paths).

13.3.7 Splines

The word *spline*, as originally used by shipbuilders, referred to a springy wooden strip that was fixed in several points and took the most naturally smooth shape given the restrictions. In computer design, a spline is a mathematical curve that goes through given points and has some desirable properties, such as smoothness and naturalness.

Bézier curves (12.1.4) are one kind of a spline, but there are other kinds too. Bézier curves are flexible and powerful; using them for building paths has many years of tradition behind it. All modern graphics software supports them in much the same way, and millions of users are familiar with them. And yet, once you try something better, the disadvantages of Béziers become obvious.

Via path effects, Inkscape implements two other spline types: BSplines and Spiro splines. The Pen and Pencil tools can produce BSplines and Spiro paths directly (14.1.4).

13.3.7.1 BSpline

A *BSpline* is a smooth curve that is inscribed into a polygon that consists of the nodes of the original path (Figure 13-34). It does not go through those node points but rather treats them as posts on a ski run—trying to pass them closely but without losing speed (that is, without sacrificing smoothness). Only the start and end nodes of the curve lie on the BSpline, while the smooth interior nodes are *off-curve*. (In fact, these off-curve nodes behave almost as if they were Bézier handles of a single monstrous Bézier that can have any number of handles and not just two.)

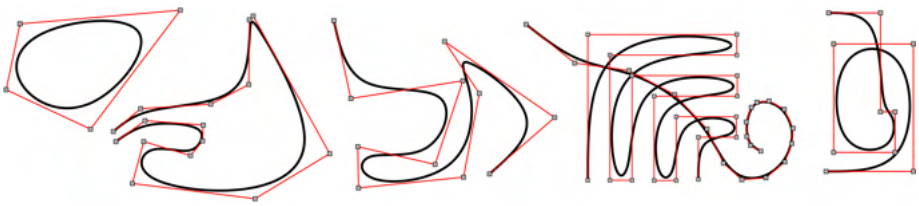


Figure 13-34: Examples of BSplines

A BSpline path can also have cusp nodes (which *do* lie on the curve, as they start or end a curved segment). A node of the original path that has both Bézier handles of nonzero length becomes an off-curve smooth point of a BSpline; a node that has at least one of its handles retracted is interpreted as a cusp node. To switch node types in a BSpline, in the Node tool, use Shift-S (to smooth) and Shift-C twice (to cusp, the first Shift-C just changes the type of the node and the second actually retracts the handles).

The fact that, with the exception of the first/last and cusp nodes, all other nodes are off-curve is perhaps the biggest disadvantage of BSplines in practice. Still, if you want a path that always remains naturally smooth and is easier to manipulate than a sequence of Béziers, the BSpline effect is worth trying.

13.3.7.2 Spiro Spline

Spiro splines, developed by Raph Levien, are another way of defining curvilinear paths. Spiros take some getting used to, but for certain tasks (such as lettershape design, Figure 13-35), they have a clear advantage over Bézier curves. Compared with BSplines, Spiro paths are often harder to manipulate—sometimes they're outright cranky—but, in return, they are smooth at an entirely new level of naturalness.

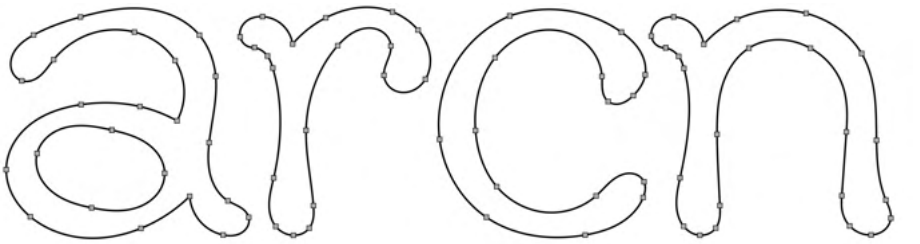


Figure 13-35: Lettershapes created with Spiro paths

A Spiro path is defined by a sequence of nodes. A Spiro path has no off-curve nodes or handles; unlike a BSpline, a Spiro goes through all of its nodes. The curvature of the path is defined entirely by the positions of the nodes and their types, as shown in Figure 13-36. The path behaves very similarly to the original shipbuilders' *spline*—a springy rod that is forced to pass through the given points and assumes the *minimum possible curvature* to satisfy the requirement.

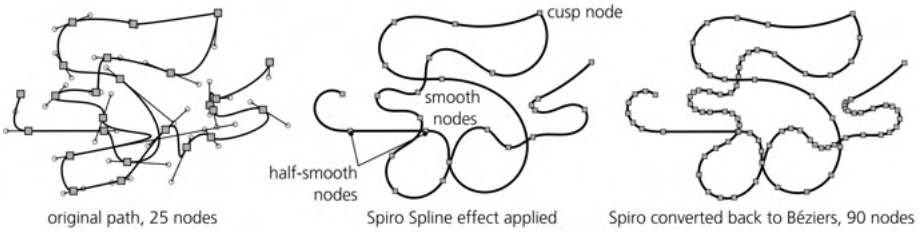


Figure 13-36: Converting a regular path to a Spiro path and back

The major selling point of a Spiro path is that it is always very smooth—not just superficially or locally smooth, as in having no cusps, but smooth at the level of the entire path, which you can approximate with Béziers only by a lot of laborious tweaking. With a Spiro, moving one node may visibly affect the curve at a distance of several nodes away. This does take some getting used to but the result may be well worth the effort.

With Béziers, the main problem is that each node has not only a position but also its own intrinsic direction and curvature, as defined by its handles. So, whenever you move a Bézier node around, you also need to adjust its handles carefully so that the curve remains naturally smooth. With a BSpline or a Spiro, just move the node wherever you want the curve to go, and the smoothness is taken care of automatically.

To create a Spiro path, select any path and assign the **Spiro spline** path effect to it. There are no parameters. Each node of your path becomes a point of a Spiro path, depending on the type of node (12.5.5), as shown in Figure 13-37:

Smooth nodes
: Nodes with two collinear Bézier handles are smooth points on the Spiro path. The length and direction of the Bézier handles on the source path are ignored, so long as they remain collinear (that is, so long as the node is smooth). Press Shift-S to line up the handles of the selected node to make it smooth.

Half-smooth nodes
: Nodes with one Bézier handle collinear with a straight line segment on the other side behave exactly the same on a Spiro path: they sit between a straight line and a curve and force them to join smoothly without a cusp. If you have a straight line segment on one side of a node, the first Shift-S will make it half-smooth.

Cusp nodes
: Nodes on the source path become corner points of the Spiro path. They behave like free hinges on the springy rod, allowing it to bend at any angle. Between two corner points, the Spiro path is always a straight line. To make a node cusp, press Shift-C twice (the first Shift-C just changes the type of the node and the second actually retracts the handles).

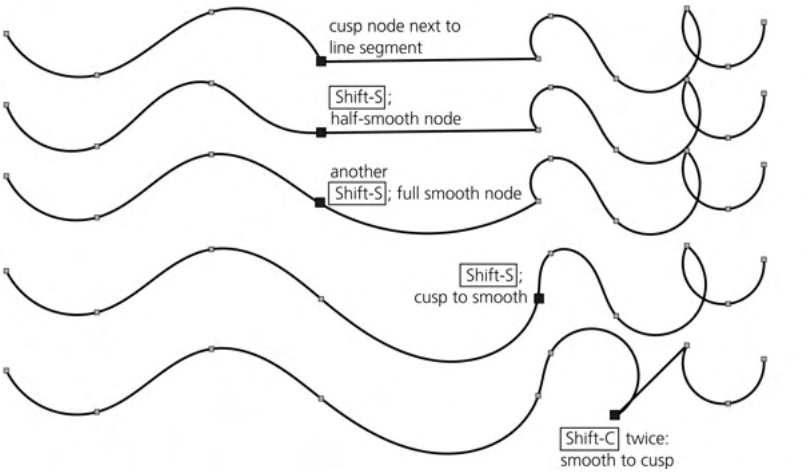


Figure 13-37: Playing with node types in a Spiro path

NOTE *The actual collinearity of a node's handles is what matters for Spiro splines, regardless of the type that the node has in the Node tool. For example, if a node designated as cusp (diamond-shaped) has collinear handles, it will still be a smooth curve point on the Spiro path.*

The biggest problem with Spiro splines is that some configurations of points are unstable and produce wild loops and spirals instead of a smooth curve (Figure 13-38). Still, sensible sequences of points usually work fine; you just need to avoid sharp changes in direction between points to prevent such instability.

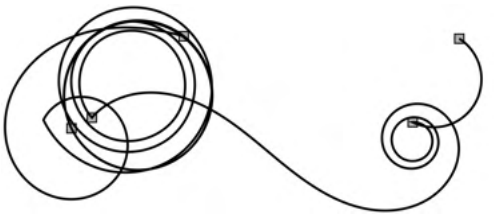


Figure 13-38: A divergent Spiro path with five nodes

When editing Spiro paths with the Node tool, the red highlight of the source path may be a distraction; you can turn it off with a toggle button in the controls bar.

13.3.8 Path Utilities

This category includes effects that do some useful tweaks to paths that would be time-consuming to do manually.

13.3.8.1 Corners (Fillet/Chamfer)

Rounding corners is a very common operation in design and technical drawing. Rectangles have convenient handles for corner rounding (11.2.2), but a generic path does not have this luxury. Of course, you can round a path's sharp corner manually, but it's a cumbersome task that makes the path harder to edit or transform later. Fortunately, Inkscape has a path effect that does all the hard work for you and preserves the flexibility of the original path. It's even better than rectangle rounding because it allows different rounding radii for different corners and supports different rounding profiles!

The term *fillet* means rounding a corner with an arc, whereas *chamfer* means cutting a corner with a straight line segment. To make things more fun, both fillets and chamfers can be *inverted*—that is, flipped around the corner-cutting diagonal. Chamfers can be multistep—that is, consisting of more than one straight line segment (and obviously, it's only for multistep chamfers that inverting makes any difference).

You can adjust the size of the fillet/chamfer for any individual node (not necessarily cusp) via onscreen handles in the Node tool, as shown in Figure 13-39. You can also change the Radius value in the parameters pane; this (as well as any other parameter change, such as corner type) will by default apply to all nodes of the path but can be limited to a selection of nodes if Change only selected nodes is on. The radius can be in absolute units or a percentage of the length of the adjacent segment.

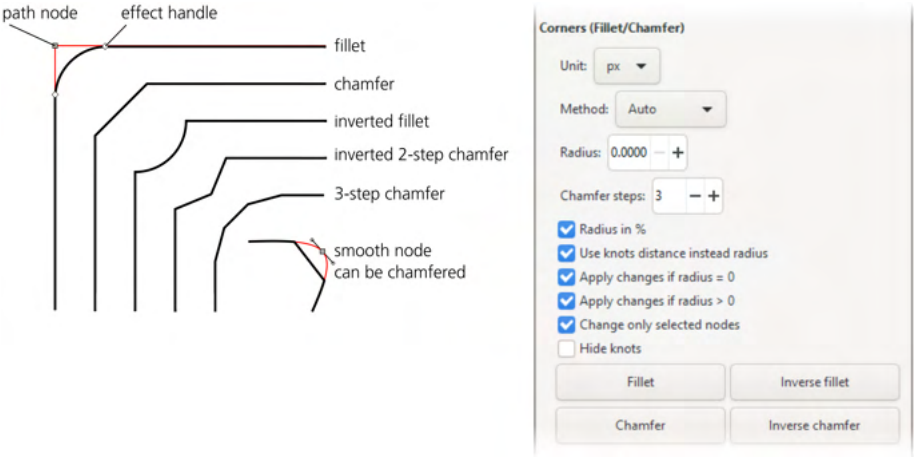


Figure 13-39: The Corners (Fillet/Chamfer) effect

13.3.8.2 Knot

This effect breaks a path into subpaths in order to create gaps between them where the path (or a group of paths) self-intersects. Figure 13-40 demonstrates how to turn a stroked path with self-intersections into a Celtic knot.

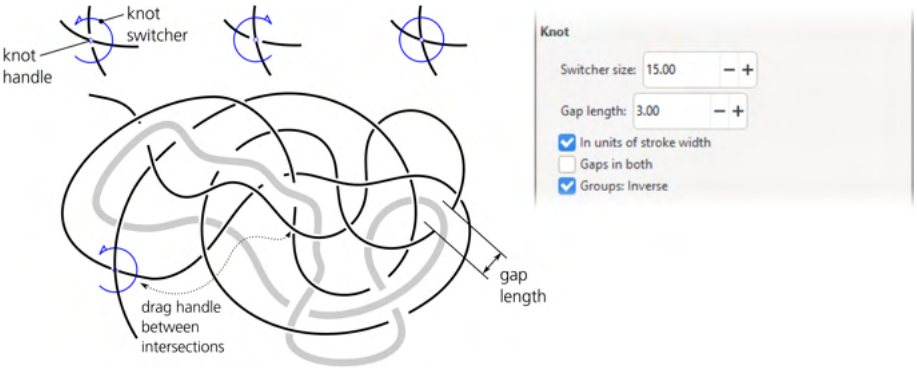


Figure 13-40: The Knot effect on a group of two paths with self-intersections

It need not necessarily be a single self-intersecting path. Remember that you can assign any path effect to a group of paths—and with Knot, this will force all paths to create gaps where they intersect all other paths in the same group.

The Gap length numeric parameter for this effect specifies the width of each gap, either In units of stroke width or in document units if this checkbox is off. If you have this effect on a group that includes paths of different stroke width, the Groups: Inverse parameter ensures that when path A creates a gap crossing path B, it takes the width of path B (and not A) as the base for calculating the gap width.

On canvas, you can control each intersection individually. In the Node tool, notice that one of its self-intersections has a diamond-shaped handle and a blue circular indicator that is open on one side. Click that handle; the indicator flips

to the other side, and the gap is now created on the *other* line at the intersection. Click it again, and you close the intersection, removing any gaps (the indicator is now a solid circle). Clicking the handle further cycles through these three states for a given intersection. To control another intersection, simply drag the handle and drop it onto the intersection you need.

13.3.9 Subpath Manipulations

A number of effects focus on the subpaths of the original path and do various interesting things to them, such as connecting or interpolating.

13.3.9.1 Stitch Sub-Paths

The surprisingly useful Stitch Sub-Paths effect works only for paths with two or more subpaths (12.1.1). It replaces the source path with a lattice of paths connecting equispaced points on the subpaths, with the Number of paths parameter controlling how densely these connecting paths go. With this effect, you can create all kinds of hair, fur, lattices, moiré patterns, or "power fields," as shown in Figure 13-41.

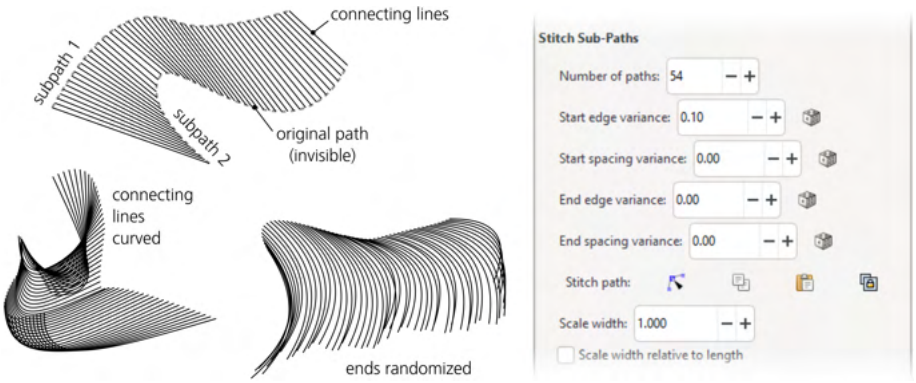


Figure 13-41: Stitching subpaths

If a path has three or more subpaths, each pair of subpaths gets its own connecting lattice (Figure 13-42). This means the number of connecting lines literally explodes as you increase the number of subpaths in the original—so don't try this effect on a path with too many subpaths or Inkscape may grind to a halt.

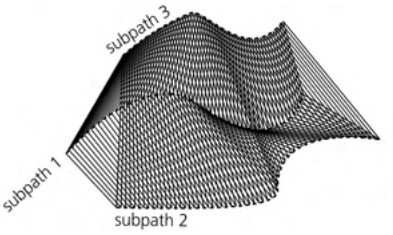


Figure 13-42: Stitching three or more subpaths

The connecting lines need not be straight—that is just the default. You can use the Stitch path link parameter to paste or link any existing open path to serve as the template for the stitches, or you can edit that template with the Node tool. The Scale width parameter scales the stitch path in the direction perpendicular to its start-end direction (the value of 1 gives it its natural width). The Scale width relative to length parameter makes the width of each stitch depend on the length of that stitch, as shown in Figure 13-43.

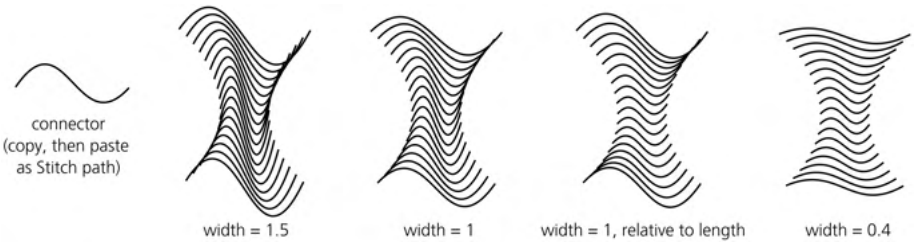


Figure 13-43: Adjusting the width of a curved stitch path

Finally, a group of randomization (variance) parameters allows you to shuffle the attachment points of the stitches, both along the path (spacing) and perpendicular to it (edge), separately for the beginning and end of each stitch, as Figure 13-44 demonstrates.

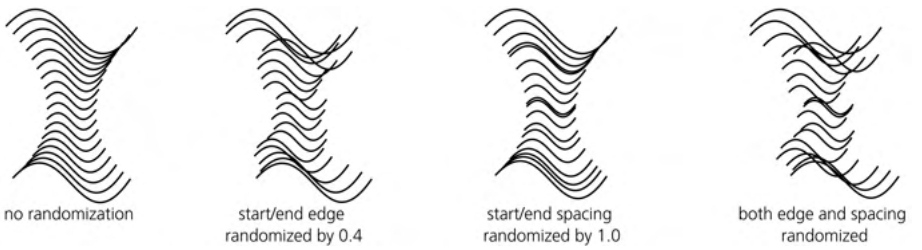


Figure 13-44: Randomizing the stitched subpaths

13.3.9.2 Interpolate Sub-Paths

This effect is a natural complement for Stitch Sub-Paths. Instead of drawing lines from one subpath to another, it draws a number of interpolated subpaths between the original path's subpaths. Just as with Stitch Sub-Paths, you can node-edit or link to a path that guides the placement of the interpolation steps, as shown in Figure 13-45.

Figure 13-45: The Interpolate Sub-Paths effect

For best results, the subpaths should be similar enough, ideally with the same number of nodes; otherwise, interpolation may create inexplicable ugly cusps.

13.3.10 Boolean Operations

You're already familiar with Boolean operations on paths (12.2). They are an essential tool, but they have a big problem: they are destructive. Once you union two paths, they no longer exist as separable objects, so you cannot copy or clone one of them for reuse, nor can you node-edit them separately. Some parts of the original objects (in the case of a union, those areas where the paths overlap) are lost forever. Once again, path effects are a natural solution.

The Boolean operation path effect applied to one path stores a link to another path and makes the first path look like a result of a Boolean operation (union, cut, division, intersection, difference, or symmetric difference) with the linked path. The second (linked) path is not destroyed but hidden (4.1), unless you uncheck the Hide linked parameter; however, you can easily select it for editing with the Select original button in the parameter panel of the effect.

For example, to make the "an" monogram shown in Figure 13-46, I assigned the Boolean operation effect to the "n" path, then selected the "a" path, copied it, selected "n" again, and pasted it to the effect's Operand path parameter. The original "a" path is still present in the document but is hidden.

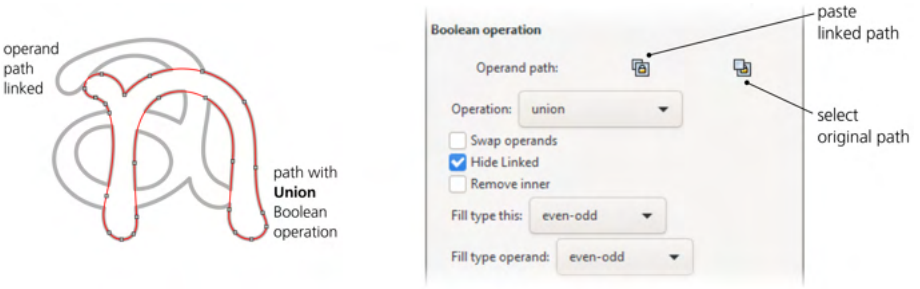


Figure 13-46: The Boolean operation effect

When I select the monogram, the bounding box enframes the entire unioned path, but the Node tool can edit only the "n" path (as shown by the red outline and highlighted nodes); to edit the "a," I need to click the effect's Select original button. Note that both letters were made with Spiro splines (13.3.7.2), so the Boolean operation effect on "n" is stacked on top of the Spiro effect.

13.3.11 Offset

You've already seen the regular offset commands (12.4) that expand (*outset*) or contract (*inset*) a path perpendicular to its direction at each point. Just as with plain Boolean operations, those commands are destructive—once you offset a path, you lose the original. Inkscape now offers a nondestructive way to achieve the same result.

The Offset path effect displays, in the Node tool, a single round handle on the path. You can drag that handle to any point on the path where it's convenient; what matters for the offset is its distance from the closest point on the original path. You can also specify the amount of offset numerically with the Offset parameter.

In addition, you can choose the type of join and the miter limit (compare 9.2). An outset path is similar to what you would get by adding a stroke to the original path, so these parameters give you access to the properties of this imaginary stroke. In particular, by setting Join to Miter, you make sure the sharp corners of the original path remain sharp in the outset path, as shown in Figure 13-47.

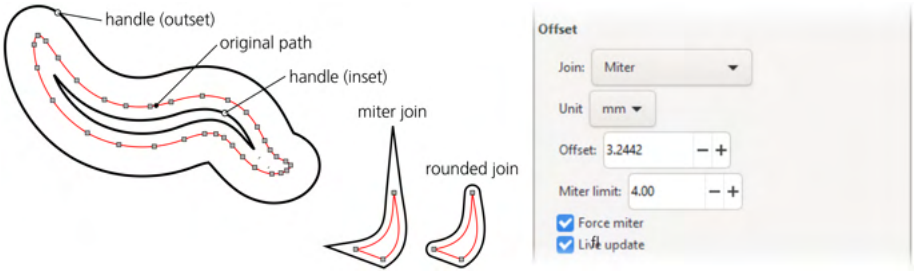


Figure 13-47: The Offset path effect

For historical reasons, Inkscape has a couple more implementations of path offsets, also interactive and adjustable on canvas but not using path effects (they were added to the program before the path effects were invented). These are the Dynamic Offset and Linked Offset commands in the Path menu and their shortcuts (Ctrl-J and Ctrl-Alt-J, correspondingly). A Dynamic Offset is very similar to the Offset path effect, whereas a Linked Offset can be emulated by the Offset effect on top of the Clone original effect (13.3.5.1). I see no reason to use these obsolete implementations now that there is a path effect that does the same thing but better (notably, the path effect can offset an unclosed path without closing it).

13.3.12 Power Clip and Power Mask

Unlike Boolean operations, plain SVG clips and masks (18.3) are already non-destructive: you can always recover the original object by removing the clip or mask. So why reimplement those features as path effects too?

In fact, there's nothing particularly "power" about the Power clip and Power mask path effects. They were created with a single purpose: to enable creating an *inverted* clip or mask that would reveal what the plain clip or mask hides and vice versa. As such, they are not even assignable via the Path Effects dialog; instead, go to the menu and choose **Object ▸ Clip ▸ Set Inverse (LPE)** or **Object ▸ Mask ▸ Set Inverse (LPE)**. After that, in the Path Effects dialog, you can check out the effect's parameters, which include checkboxes to turn the effect on or off as well as toggle the inversion.

13.3.13 Dashed Stroke

What on Earth could be improved in standard SVG's dashed strokes (9.4)? It turns out, a lot.

In SVG, a dash pattern is specified in units of stroke width, but it is not in any way coordinated with the path itself—its shape or size. When working with a simple symmetric shape, such as a rectangle, it is annoying when you can't make the dashes meet symmetrically in the corners or when two opposite and equal sides' dashes are not in sync. The Dashed Stroke path effects solves all of these problems.

In this effect, you can tell Inkscape exactly how many dashes you want in your path, not how long they need to be. This way, the dashes always stay in sync with the shape even when you scale it or change the stroke width. If the Use segments parameter is on, you get the specified number of dashes in each segment—that is, between each two nodes of the path; if the segments are not equal, the dashes will have different lengths too. Turn on **Equalize dashes** to fix this: now the entire path will have dashes of the same length as the shortest of its segments.

The gaps between the dashes are controlled by the Hole factor parameter that ranges from –1 (all gaps, no dashes) to 1 (all dashes, no gaps). The default value of zero makes the gaps the same length as the dashes.

The Half start/end parameter, on by default, makes each segment start and end with a dash of half width, so that the combined dashes at the segment boundaries have the same length as elsewhere. When this parameter is off, you will have double-length dashes at each node between segments (Figure 13-48).

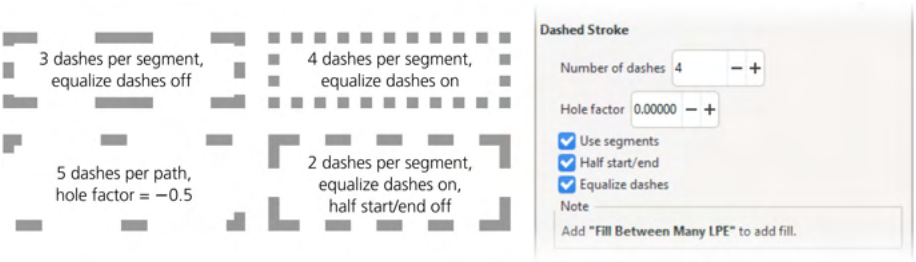


Figure 13-48: The Dashed Stroke path effect

The biggest drawback of this effect is that, since it emulates dashes by separate subpaths, the path as a whole cannot have a fill because SVG cannot fill in between subpaths of a path. One solution is to use Clone original (13.3.5.1) to clone the path *without* the effect and apply the fill you need to that cloned path.

13.3.14 Helper Effects

The Bounding Box and Show handles effects are useful when you need to create lasting visualizations of your paths—for example, when you're creating illustrations for a book such as this one. The Bounding Box effect turns one path (whose path data will be discarded) into a dynamically updated rectangular bounding

box of another (linked) path. The Show handles effect adds subpaths imitating nodes and node handles of the path as they are shown in the Node tool.

The Ruler effect turns any path (not necessarily straight) into a ruler with minor and major marks and with adjustable size and spacing. The Measure effect is a complex engine for creating measurements on segments of a path; this partly overlaps with what the Measure tool does (6.9.6) but is more powerful for its specific task—with a ton of options for colors, sizes, precision, and label formatting, as shown in Figure 13-49.

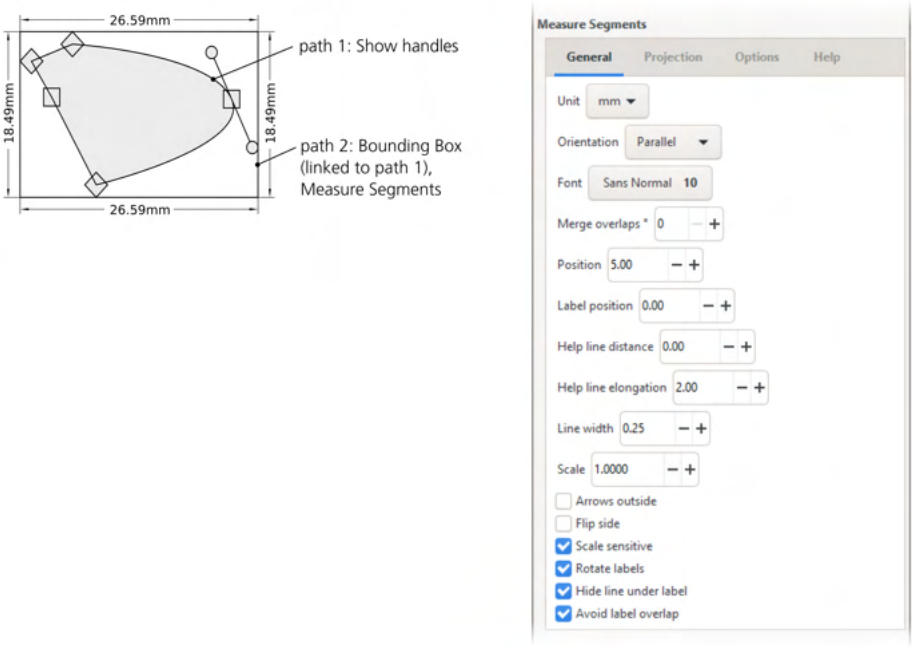


Figure 13-49: Examples of Show handles, Bounding Box, and Measure Segments effects

13.3.15 Geometric Constructions

A final group of path effects will help you do some simple interactive geometric constructions (Figure 13-50). In the Path Effects dialog, they are currently marked as "experimental" and hidden by default.

Circle by three points
: Using the Pen, click three times to create a three-point path, assign this effect, and Inkscape will create a circle that passes through those three points. A circle is never clipped—even though it can potentially be infinite when the three lines are on the same straight line (in practice, placing the points on the same line makes the effect misbehave). This path effect has no parameters.

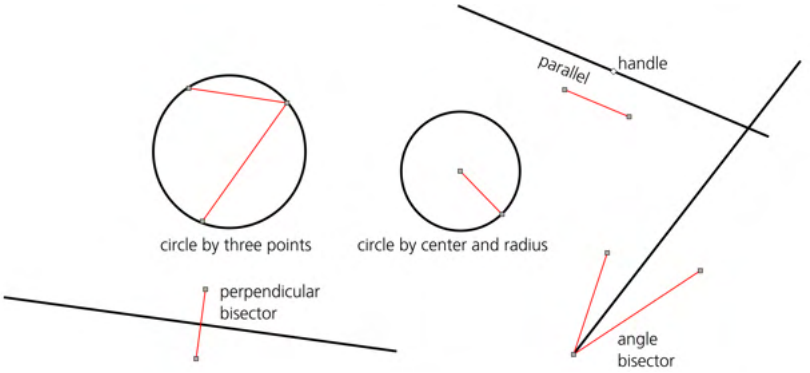


Figure 13-50: The geometric constructions extensions

Circle by center and radius
: The first node of the path gives the center, and the last one sets the radius. This path effect has no parameters.

Parallel
: This effect creates a line running through an *anchor point* parallel to the line connecting your path's start and end nodes. You can move the anchor point via a diamond-shaped handle in the Node tool or via the Offset parameters of the Parallel path effect. To change how far to the left and right of the anchor point you want the parallel line to go, use the **Length left** and **Length right** parameters.

  The original path—to which the new line is parallel—becomes invisible when you create the parallel: the parallel is now the visible path of that object, and the original nodes are editable only in the Node tool. If you need two visible straight lines that are always automatically parallel to each other, create *two* lines and make the second one a **Clone original** (13.3.5.1) of the first but choose the **Without LPE's** option. Then, select the first one and turn it into its own parallel using this tool.

Perpendicular bisector
: This works similar to the Parallel effect except that instead of a parallel line, it creates a line that is perpendicular to the given segment and goes through its center. If you need two lines to be automatically perpendicular to one another, create a **Perpendicular bisector** on one of them, then open the **Path Effects** dialog, add a **Clone original** effect to it, and move it up before the **Perpendicular bisector** effect. Finally, select the second line object, copy it, and paste into the **Link** parameter of the **Clone original**.

Angle bisector
: This works similarly to the Perpendicular bisector except it bisects an angle created by a three-node path. Just assign this path effect to a path with three nodes.

Mirror symmetry
: See 13.3.5.3 for a description of the Mirror symmetry effect.

Ellipse from points

This effect adapts to the number of points in the original path. For two or three points, it creates a circle; for five points, it creates an ellipse that goes exactly through those five points. For any other number of points, the effect can't be exact, but it tries as best it can, creating an *approximate* ellipse that minimizes the deviations from all points (Figure 13-51).

Optionally, you can add a frame or axes to the ellipse. If you set the Method parameter to Force circle, the effect will always create a circle, either exact or approximate.

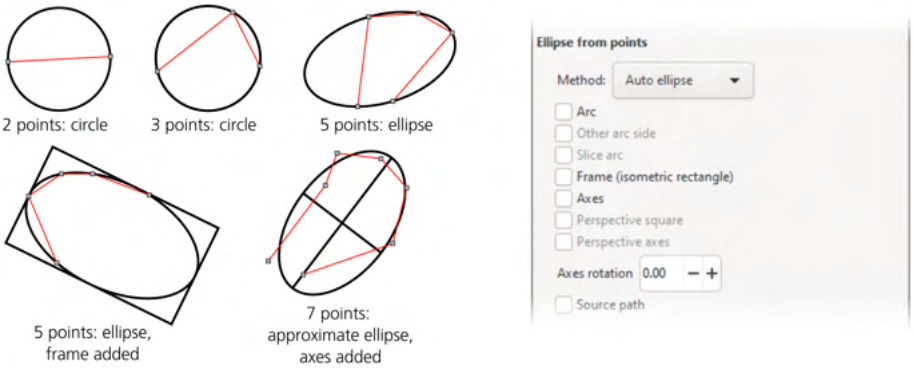


Figure 13-51: The Ellipse from points effect

13.4 Path Extensions

In the Extensions ▸ Generate from Path, Extensions ▸ Modify Path, and Extensions ▸ Visualize Path submenus, Inkscape has a number of extensions (Chapter 19) that work with paths. Through the years, many of them have been reimplemented as path effects—with on-canvas editability, better integration with the rest of the program, and, typically, richer options. However, Inkscape still includes most of the obsoleted extensions.

One reason is that an extension, as a one-off operation, does not introduce any non-SVG attributes into the document, so it is not prone to potential incompatibilities that path effects may face (13.1). Extensions are also much easier for curious Inkscape users to study, experiment with, and develop in new directions. Finally, some of these extensions still have some unique capabilities or options not available in path effects.

13.4.1 The Generate from Path Submenu

13.4.1.1 Inset/Outset Halo

This extension adds to the selected path a specified number of inset or outset paths (Steps), at a Width from each other, each further offset having lower and lower opacity (Figure 13-52).

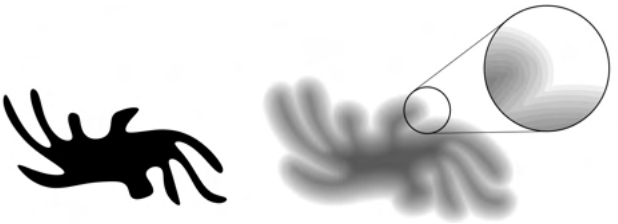


Figure 13-52: The Halo extension may be considered "a poor man's Gaussian blur."

13.4.1.2 Extrude, Motion

These extensions create a primitive 3D effect by extruding the selected path at a given angle to a given distance. The result is a group of two objects, one being the original path and the other its extrusion skirt that you can style differently. A similar Extrude path effect is currently marked as experimental (Figure 13-53).

Figure 13-53: The Extrude extension

13.4.1.3 Interpolate

This extension creates an interpolation, or *blend*, between two paths—a set of smooth transition steps when you transform one path into the other. You can specify the number of the Interpolation steps and the Exponent, which, if different from the default 0, shifts the blend toward one of the ends, as shown in Figure 13-54.

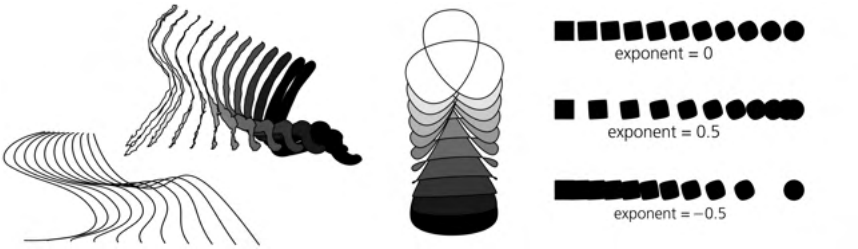


Figure 13-54: The Interpolate extension

This is similar to the Interpolate Sub-Paths effect (13.3.9). Unlike the path effect, however, this extension has the Interpolate style option that paints intermediate steps with intermediate colors.

The extension generates the intermediate steps as a group of paths; the Duplicate endpaths option adds copies of the original paths to this group as well. Since interpolation always connects beginnings and ends of the two paths, you may need to reverse one of the paths (Path ▸ Reverse) in order to get the result you want.

13.4.1.4 Pattern Along Path

This is similar to the path effect of the same name (13.3.2) and, in fact, is the first version of this functionality created before path effects were invented. The extension has one advantage compared to the path effect: it can use a group of objects (each with its own style) as the pattern; in the path effect, you are limited to a single-pattern path. To use the extension, select the pattern path or group and the skeleton path (the pattern must be on top of the skeleton in z-order), and apply the command; the options are similar to those for the corresponding path effect.

13.4.1.5 Voronoi Diagram and Voronoi Pattern

A Voronoi diagram (also used by the pixel tracer, 18.5.2.6) is a tessellation of a plane into polygons in such a way that each point on a polygon is closer to the center of its own polygon than to the centers of other polygons (Figure 13-55).

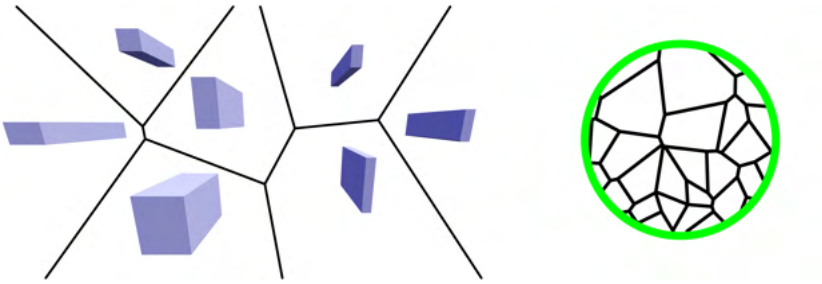


Figure 13-55: Voronoi diagrams from selected 3D boxes (left) and as pattern on a circle (right)

The first of these extensions generates such a tessellation from the centers of all selected objects; the second one creates a random Voronoi pattern and assigns it to the selected objects as fill (and you can then use it on other objects as well, see 10.8 for details).

13.4.2 The Modify Path Submenu

13.4.2.1 Add Nodes

This useful extension creates extra nodes in a path without changing its shape. You can specify either the maximum allowed distance between adjacent nodes or the number of segments into which each segment will be divided. Figure 9-11 on page 169 shows this extension in action.

13.4.2.2 Flatten Béziers and Straighten Segments

The Flatten Béziers extension approximates each Bézier curve in the selected path with a sequence of straight line segments; higher Flatness values result in rougher approximation with fewer segments. The Straighten Segments simply shortens all Bézier handles by the given percent, so that setting this to 100 percent turns each Bézier curve into a straight line segment, as Figure 13-56 demonstrates.

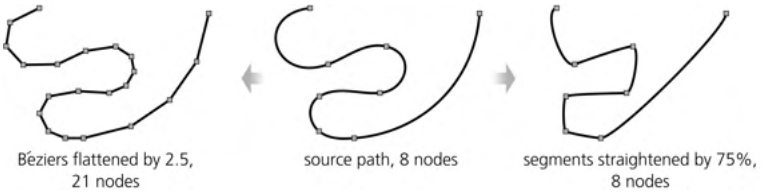


Figure 13-56: The Flatten Béziers and Straighten Segments extensions

13.4.2.3 Jitter Nodes

This extension randomizes the selected paths by displacing all nodes in random directions and at random distances, limited by the Maximum displacement parameters that can be set separately for X and Y (for example, Maximum displacement in X set to 0 will jitter nodes only vertically). Also, you can enable jittering separately for nodes themselves and for their Bézier handles, as Figure 13-57 demonstrates.

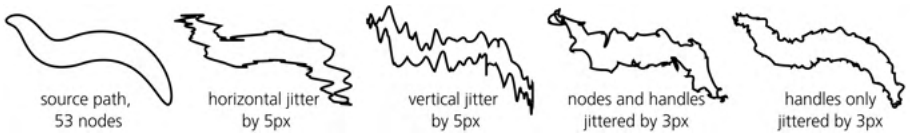


Figure 13-57: The Jitter Nodes extension

The Distribution of the displacements parameter governs the probabilities of different displacement values, as Figure 13-58 demonstrates. By switching from the default Uniform to Pareto, you can make smaller displacements a lot more probable than larger ones (much like in life, the probability of a random city having a huge population is very small). Gaussian distribution has a more blunted peak, with the probabilities falling off slowly around the peak but faster as you move further off. The Log-normal distribution looks like Pareto, but its extreme values are more extreme and may fall far outside the Maximum displacement range.

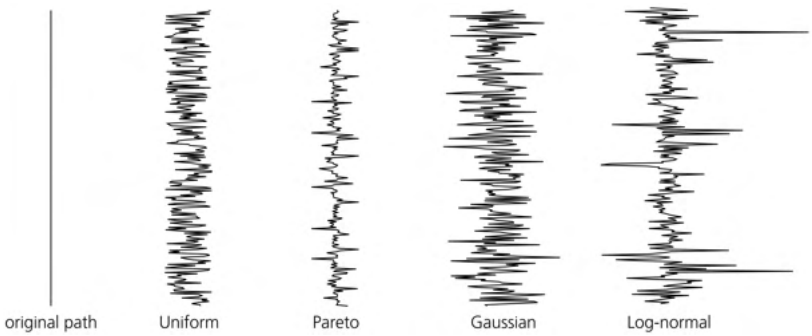


Figure 13-58: Probability distributions in Jitter nodes: horizontal displacement only, ranges are the same.

13.4.2.4 Fractalize

This is another way to randomize a path. It flattens any Bézier curves in the original path, then creates new nodes between the existing ones and moves these new nodes around, leaving the original nodes in place. The Subdivisions parameter determines how many times each segment will be subdivided in two (for example, 10 subdivisions turn a two-node path into one with 1,025 nodes because $2^{10} + 1 = 1025$). Smoothness changes how far the new nodes can move; a lower Smoothness produces a rougher path. This effect is perfect for creating coastal lines in fantasy maps (Figure 13-59).

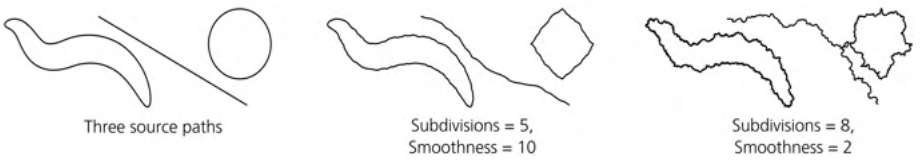


Figure 13-59: The Fractalize extension

13.4.2.5 Mesh

The two extensions in this sub-submenu convert a mesh gradient (10.7) to a path and back, which might be a lifesaver given that Inkscape's tools for editing mesh nodes are currently rather poor (10.7.3).

13.4.2.6 Pixel Snap

Use this extension to snap all nodes in all selected paths to px grid boundaries, so as to minimize anti-aliasing when exporting your artwork at 96 dpi (18.6.1.2).

13.4.2.7 Rubber Stretch and Whirl

These extensions distort selected paths as Figure 13-60 shows. They work best on paths with many nodes because they only displace existing nodes without creating new ones. The Whirl extension works around the center of the view in Inkscape when you start the extension; to place the center of the view at the geometric center of the selection, press 3 or choose **View ▸ Zoom ▸ Selection** (3.11).

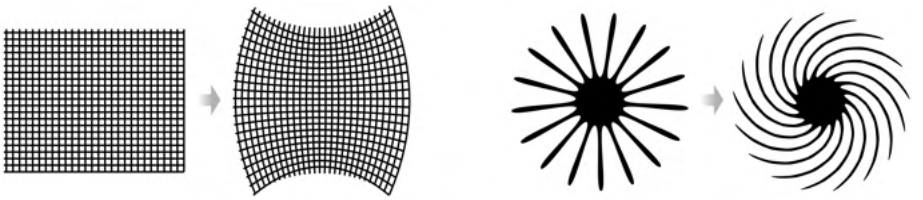


Figure 13-60: The Rubber Stretch (left) and Whirl (right) extensions

13.4.3 Visualize Path Submenu

13.4.3.1 Number Nodes

This replaces a path with a group of dots, each marking a node of the original path with a sequential number (a text object). You can adjust the dots' size and the font size of these numbers.

13.4.3.2 Draw Handles

This extension visualizes the Bézier handles of the selected paths, similar to the Show handles path effect (13.3.14). For each selected path, it creates a new path with each handle being a subpath.

13.4.3.3 Dimensions and Measure Path

The Dimensions extension creates a frame and dimension lines around the selected object (not necessarily a path); use the **Measure Path** extension to add the actual length measurements to it. This is similar to what the Measure effect does (13.3.14).

14

DRAWING

Historically, one of the names of vector editors was *vector drawing*—or simply *drawing*—applications. Bitmap editors, on the other hand, are sometimes called *paint programs*. The contrast between drawing and painting, stemming from traditional media (paper versus canvas), is thus carried on into the digital realm. Even though vector programs are now used—perhaps even preferentially—for higher-level tasks, such as composition and layout, simply *drawing* is still the most basic use case for this kind of tool.

It is true that vector drawings can never be as naturalistic and "painterly" as bitmaps; even the best vector art has that recognizable smooth, computer-generated look. Often, however, this is not a problem—it can even be an advantage. Also, vector editors have something that no bitmap editor can match: the ability to treat each stroke as a separate object that never merges or "flattens" into others (unless you tell it to). As it turns out, this infinite tweakability is sometimes more important for producing a good freehand drawing than the ability to imitate pastel strokes or wet-on-wet watercolor.

Inkscape has a number of drawing tools to choose from, depending on the type of art you want to create (Figure 14-1). If you need straight-line geometry or spline-shaped paths with precise placement of nodes, use the Pen tool (14.1.1). For freehand paths, more or less smoothed out, use the Pencil tool (14.1.2). For complex, naturalistic, pressure-sensitive filled strokes imitating various physical effects, such as trembling or inertia, use the Calligraphic pen (14.2). To do cartoon-like fillings between multiple complex strokes, Inkscape has the Paint Bucket tool (14.3).

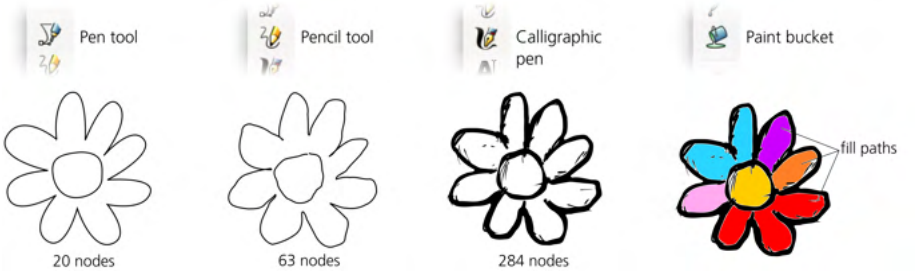


Figure 14-1: Inkscape's drawing tools at a glance

A number of other tools deserve an honorable mention in this chapter on drawing. You can quite literally draw with copies or clones of objects using the Spray tool (4.7). The Tweak tool is essential for artwork creation, whether you use it for shuffling your strokes around (6.10), applying color (8.9), or freehand path pushing (12.6). You can use the Eraser tool (14.4) to wipe out parts of your drawing. Finally, Inkscape has a specialized Connector tool for drawing diagrams (14.5).

This chapter would be incomplete without mentioning that you can easily rotate Inkscape's canvas, with all its contents, using Ctrl-Shift-wheel (3.13). You can also flip (mirror) the canvas with commands in the View ▸ Canvas Orientation submenu. For an artist, the ability to place the drawing at the most convenient angle for stroking is a big help.

14.1 The Pen and Pencil Tools

Traditionally, ink pens are associated with writing and draftsmanship, and graphite pencils with artistic sketching and drawing. In Inkscape, the sibling Pen and Pencil tools are both used for creating paths. The difference is that you use the Pen when you want to control placement of individual nodes, whereas the Pencil is for freehand doodles where Inkscape takes care of the nodes and other details. As all metaphors are, this one is somewhat arbitrary, but it has stuck.

In this section, I first describe the basic operation of Pen and Pencil and then look at the advanced features—drawing modes and stroke shapes—that these tools have in common and that make use of path effects (Chapter 13).

14.1.1 The Pen Tool

The Pen tool, where you create individual nodes by clicking the canvas, is reminiscent of the Node tool (12.5) except it is optimized for creating new nodes instead of editing them.

Switch to the Pen tool (Shift-F6 or B), make sure it is in the default Bézier mode (the leftmost button on the controls bar), and click the canvas. You have created a first node of a new path. Just moving the mouse around (without clicking) moves a red segment that shows you how the path will go if you create the next node where you are.

Keep clicking to add more nodes. To finish the path, double-click (this adds one more node and finishes the path), or right-click, or press Enter (both these methods finish the path and cancel the red segment without adding any more nodes). It is only at this point that the entire path is actually created as an object; what you saw until now was just a virtual scaffold. Another way to finish a path is to *close* it; if you create the next node exactly over the first node of the path (it is marked by a square anchor), the path is closed and finalized.

If you click and drag instead of just clicking, you're creating a smooth Bézier node (Figure 14-2); the blue straight line with round handle(s) at the end(s) that moves while you drag is the single handle (for the start node) or two symmetric handles (for a mid-node, 12.1.4). As soon as you release the mouse button, the node's handles are fixed (but the path is still being created).

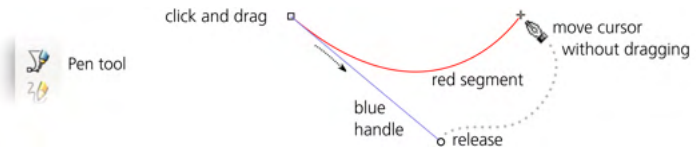


Figure 14-2: Pen: first node set, ready for the second

After you create the second node, the part of the path between the two nodes defined so far becomes green, and you once again have a red segment stretching from the last created node (Figure 14-3). You can repeat this indefinitely—keep clicking and dragging (for smooth symmetric nodes) or just clicking (for cusp nodes); on each step, the path you have laid out so far is green, and the last segment you are about to create is red.

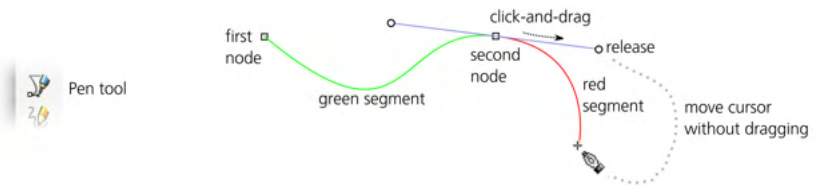


Figure 14-3: Pen: two nodes set, ready for the third

To cancel creating a path at any time, press Escape or simply switch to another tool.

NOTE *Most key and mouse shortcuts for zooming, scrolling, and panning (Chapter 3) work in the Pen tool without disrupting path creation. For example, you can middle-drag the canvas and middle-click to zoom in so you can better position your next node.*

14.1.1.1 Node Types

As you've just seen, click-and-drag creates a smooth node with two symmetric handles (unless this is a first node of a path that has only one handle). A simple click creates a cusp node without any handles. You can also create cusp nodes with noncollinear handles; for this, click-and-drag, but then, once the opposite handle is where you want it, press Shift while dragging. This fixes the far handle and lets you move the near handle independently, as Figure 14-4 demonstrates.

Figure 14-4: Creating cusp nodes with the Pen

You can also press Ctrl while dragging a handle to snap it to 15-degree increments (compare 6.3; you can choose another snap value in Preferences ▸ Behavior ▸ Steps). Pressing Ctrl while moving the mouse around between creating nodes will similarly snap the direction between the last created node and the mouse point; this is an easy way to create strictly horizontal or vertical lines, as shown in Figure 14-5.

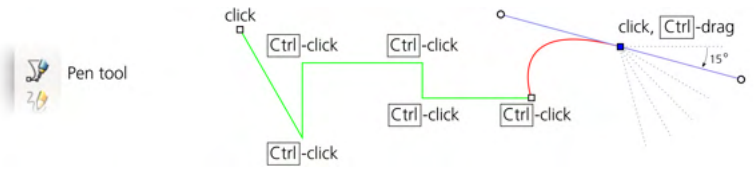


Figure 14-5: Pen: snapping with Ctrl

14.1.1.2 Going Back

You can always step back in your path creation process. To cancel the last added green segment, press Backspace or Delete once, or press it repeatedly to remove several segments. Removing the last remaining node cancels the entire path.

It is not always easy to click exactly where you want the node to appear on the first attempt. Inkscape allows you to move the last created node (that is, the end of the green segment) without finalizing the path by using the arrow keys with the same convenient modifiers as in other tools: Alt to move by 1 screen pixel, Shift to move by 10 times the distance (6.5.1).

14.1.1.3 Continuing a Path

When you're in the Pen tool, any single selected path displays its end nodes, if it has any (that is, if it is open), as little square *anchors*. (This is the case for any path you just created with this tool itself—after finalizing, it remains selected.)

These anchors allow you to continue adding to the selected path (by placing your first node in the anchor) or to close it (by penning from one anchor to the other).

You can also add a new subpath (12.1.1) to the selected path. To do this, just hold down Shift *while* you click to create the first node. After that, create and finalize the path as usual; your path will be added as a subpath to the selected path. All open subpaths of the selected path show anchors, which means you can close them or connect them to one another (Figure 14-6).

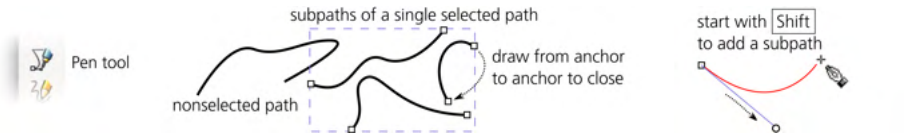


Figure 14-6: Pen: adding or closing a subpath

14.1.2 The Pencil Tool

The Pencil tool (F6 or P) is similar to the Pen tool. The main difference is that with the Pencil, you don't worry about individual nodes or handles; you just draw a freehand line and Inkscape approximates it with a path.

14.1.2.1 Smoothing

When you draw with the Pencil, the resulting path is always smoother than the actual trace of your mouse drag. To change how much smoothing is applied, use the Smoothing section of the tool's controls bar. Smaller values of Smoothing produce more precise and node-rich approximations of what you have drawn; conversely, larger values create more generic and simplified paths with fewer nodes (Figure 14-7). At the maximum of 100, most mouse drags will create just a single Bézier segment between two nodes.

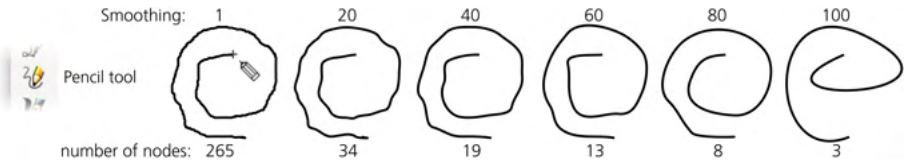


Figure 14-7: Pencil strokes at various smoothing levels

Next to the numeric Smoothing value, a toggle button called LPE based interactive smoothing controls how the smoothing function works. With this button unpressed, you activate the old method where smoothing, at the specified level, is applied once when the path is finalized, and the unsmoothed source path is discarded. This way, smoothing is applied only to newly created paths in the Pencil tool and cannot be changed afterward.

With this button pressed (now the default), instead, the source path is stored at the minimum level of smoothing but then a Simplify path effect (13.3.4.4) is applied to it. This method of smoothing is fully reversible (you can just remove the path effect) and adjustable (you can change the amount of smoothing, whether via the effect's parameters in the Path Effects dialog or simply by varying

the Smoothing value in the Pencil tool's controls bar when any path with the Simplify effect is selected). If you like the interactivity of the new method but don't want paths with path effects in your document, the Flatten button will remove the effect and produce a bare path at the current Smoothing level.

This approach is wonderful in theory, but its implementation has some kinks. I found that if you want the absolute minimum of smoothing, you should set the value to 1 (out of the maximum of 100), not zero. Also, at lower values of Smoothing, the Simplify path effect sometimes produces more, not fewer, nodes than the original had.

14.1.2.2 Pressure Sensitivity

There was a time when the only tool in Inkscape that used the pressure signal of a tablet pen was the Calligraphic Pen (14.2). Over time, it was joined by Tweak (6.10, 8.9, 12.6), Eraser (14.4), Spray (4.7) tools. Now, the Pencil tool can also use a pressure-sensitive input device to create paths with variable stroke width. It does this via the Power stroke path effect (13.3.1.1).

Enable pressure-sensitive drawing by clicking a toggle button on the controls bar. Now the Shape controls (14.1.5) are gone (because the pressure-controlled strokes cannot have a predefined shape), and the drawing mode buttons (14.1.4) are disabled. Instead, you see numeric controls for the minimum and maximum width (which correspond to the minimum and maximum pressure of the pen), as well as a drop-down for choosing the type of stroke caps. These controls apply only to the newly created strokes; if you want to change width or caps in an existing stroke, you can do so via the Power stroke path effect it has (13.3.1.1).

Pressure-enabled drawing leaves a trail of red circles; their sizes indicate the pressure in each point, and their spacing is determined by the speed of your pen movement. Once you lift the pen, a path with Power stroke is created that approximates (with some smoothing) the pressure data recorded during your stroke, as shown in Figure 14-8.

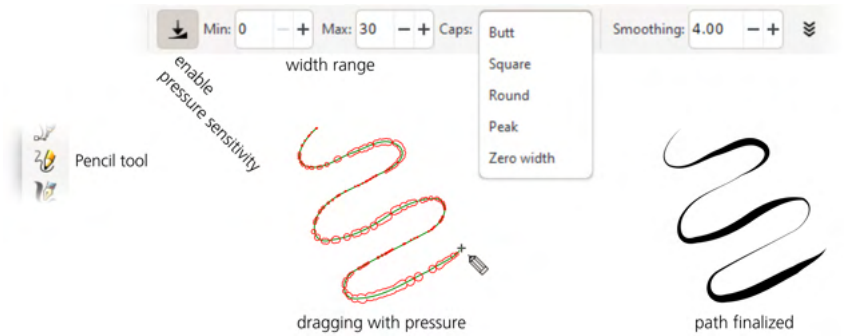


Figure 14-8: Drawing with the Pencil using a pressure-sensitive tablet pen

14.1.2.3 Continuing a Path

Similar to the Pen tool, in the Pencil tool, any selected path displays anchors on the open end nodes of all its subpaths. You can continue, close, and connect subpaths by drawing from one such anchor to another, and you can add new subpaths to the selected shape by beginning to draw with Shift held down.

14.1.2.4 Sketching with Alt

Drawing with Alt in the Pencil tool enables *sketching*. Any number of drags that you do before letting go of Alt will be *averaged*, and that single averaged path will be created only after you release Alt. You can think of this as an inversion of what the Sketch path effect does (13.3.4.1). Sketching allows you to "feel your way" when drawing—to zero in on the best line without creating a thicket of tentative discardables.

On the Pencil tool's Preferences page, the Average all sketches setting (on by default) averages *all* sketching strokes made while you hold Alt with equal weight. If you turn it off, the tool will average between the latest stroke and the previous average, thereby giving your latest stroke more weight.

14.1.3 Style

Like all object-creating tools, the Pen and Pencil tools can use either the last set style or their own tool style (11.1.2). By default, they use their own style, which is initially set to no fill, 1 px black stroke. This is because more often than not, the last set style has fill and no stroke, whereas when drawing with these tools, you likely want the result to have stroke but not necessarily fill. Refer to Figure 11-2 on page 202 for how to change the tool style. As usual, the style that will be used for the next path you create is displayed on the right end of the controls bar (shown in Figure 11-1 on page 201).

NOTE *When using a Shape other than None (14.1.5), you may not be able to create fill at all because the path effect that is applied to such strokes emulates shaped stroke with a closed filled path. In that case, Inkscape does what you'd expect: it applies the stroke properties of the tool's style to the fill. Thus, if None creates a black stroke with no fill, Ellipse will create an ellipse-shaped stroke with black fill but no stroke (Figure 14-11 on page 301). However, if no stroke is set in the tool style, the style's fill properties (if any) will be used for the shaped path.*

14.1.4 Drawing modes

Both Pen and Pencil have a number of modes, apart from the default *Bézier mode*, that let you create paths out of other types of splines or segments. These correspond to the Mode: buttons on the controls bars of these tools (Figure 14-9). The first button is the default mode that creates regular SVG paths out of straight lines and Bézier curves.

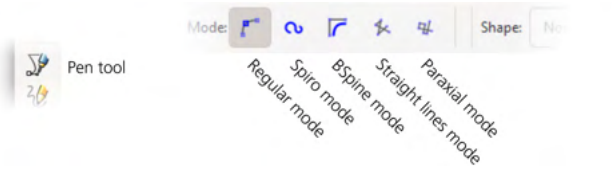


Figure 14-9: The modes in the Pen tool (Pencil doesn't have the last two)

The second mode creates *Spiro paths*—that is, paths with the Spiro spline path effect applied (13.3.7.2). In this mode, a simple click in Pen creates a smooth node, and a Shift-click creates a cusp node; there's no way to create a half-smooth node (12.5.5).

The Pencil tool, as you've seen (14.1.2.1), first applies some smoothing to your hand-drawn stroke, by default using the Simplify path effect. Then, it applies the Spiro spline effect on top of Simplify. Since, at the (default) low values of Smoothing, Simplify tends to produce *more* nodes than the original path had, the result may not look very much like a Spiro at all—because the extra nodes added by Simplify make Spiro track the path unusually closely. If you're puzzled by this, just remove Simplify from the effects stack of the path.

The third button creates *BSpline paths*—that is, paths with the BSpline path effect applied (13.3.7.1). In this mode, a simple click in Pen creates a smooth off-curve node, and a Shift-click creates a cusp node.

The two other modes (Pen-only) are restrictions of the regular mode. The *Straight lines* mode disables creation of smooth nodes: even if you drag, you end up with a cusp node without handles. The *Paraxial* mode additionally restricts the segments to being parallel or perpendicular to the first segment you create in a path; additionally, each segment is perpendicular to the one that precedes it. For example, if you restrict the first segment to vertical by pressing Ctrl, the next segment will be horizontal, then vertical, then horizontal again, and so on. That may not be much of a trick, but the true usefulness of this drawing mode is its ability to draw within a slanted coordinate system: just draw the first segment at the correct angle and then keep going in this rotated world.

In the Straight lines and Paraxial modes, if you Ctrl-click without creating a path, you create a single *dot*—a little circle shape (11.4). This is convenient in geometric and technical drawings. You can set the size of the Ctrl-click dots, in units of stroke width, in the Pen tool's preferences (double-click the tool icon in the toolbox to access its page in the Preferences dialog); the default is three stroke widths. If you use Shift-Ctrl, the dot is twice as big (Figure 14-10).

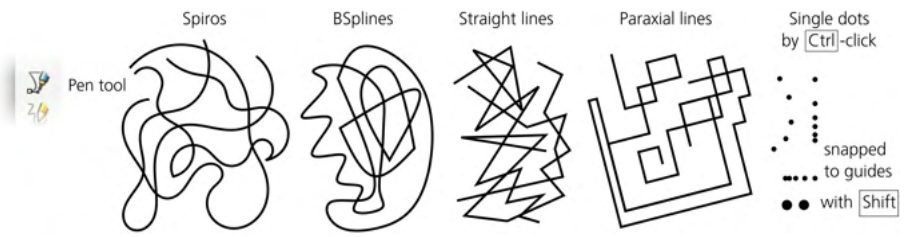


Figure 14-10: Pen: Spiros, BSplines, straight lines, paraxial lines, and single dots

14.1.5 Stroke Shapes

The Shape drop-down menu for the Pen and Pencil tools lets you choose the shape of the stroke for the path you are creating. The Scale value next to it adjusts the width of the shaped stroke (in relative units). In Pencil, these controls are shown only if you don't have pressure sensitivity enabled (14.1.2.2).

By itself, SVG does not support shaped strokes; in SVG, a stroke is always a constant-width strip (Chapter 9). All other stroke shapes are emulated by Inkscape via path effects (Chapter 13). That is why there's a difference in behavior between the default None and the rest of the stroke shapes in this list (Figure 14-11).

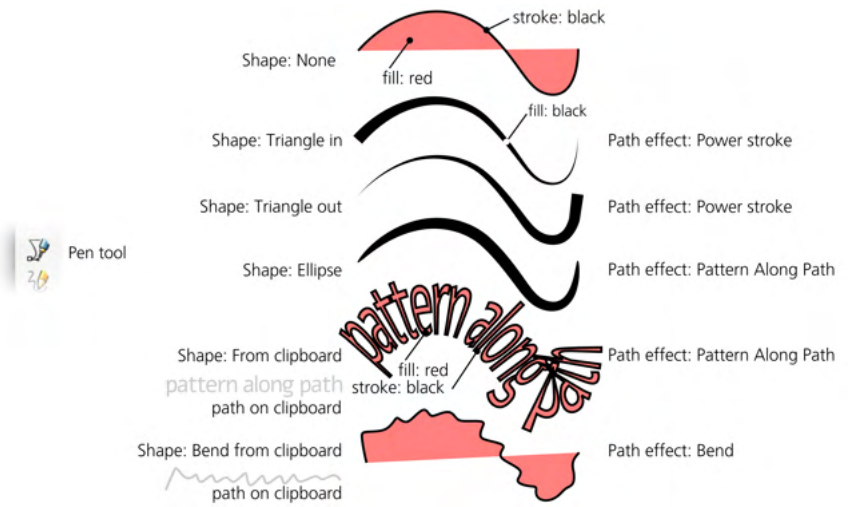


Figure 14-11: Shaped strokes with the Pen tool

With None, you get the barebones SVG path with a plain, constant-width stroke. Here's what the tool does for the other Shape options:

- Two triangles: Triangle in *decreases* the width to zero toward the end of the path, and Triangle out *increases* the width toward the end. These are created using the Power stroke path effect (13.3.1.1). To adjust width, use the Pen or Pencil's **Scale** control or drag the onscreen purple-colored handle in the Node tool.
- Ellipse: a circle with the diameter of 10 px is applied using the Pattern Along Path effect to your path (13.3.2). To adjust width, use the Pen or Pencil's **Scale** control or the onscreen handle (perpendicular at the end node) in the Node tool.
- From clipboard: any path you had copied to the clipboard gets applied to the path you draw as a pattern using the Pattern Along Path path effect (13.3.2). To adjust width, use the Pen or Pencil's **Scale** control or the onscreen handle (perpendicular at the end node) in the Node tool. To edit the shape of the stroke, use the **Show next editable path effect parameter** button in the Node tool (12.5.1).

  For example, if you create a line of text and convert it to a single path, copy it, and choose this option in the Pencil, each stroke will be painted by this line of text (appropriately bent and stretched).
- Bend from clipboard: any path you had copied to the clipboard gets applied to the path you draw as a skeleton using the Bend path effect (13.3.2.2). For example, if you create an arc, copy it, and choose this option in the Pencil,

each freehand doodle you create will be *bent* by this arc, on top of its own shape.

- Last applied: whatever shape you used with this tool the last time will be used again. Why is this useful? You can create a brush shape, copy it, apply it to a Pen or Pencil stroke once using From clipboard, then switch it to Last applied and keep using this brush so that the clipboard can be used for other things.

The path you create may or may not have a fill inside it, depending on the style the tool uses (14.1.3). With Power stroke, you cannot have any fill *inside the path*, because the tool automatically applies the style of stroke to the fill and discards the original stroke style. With Pattern Along Path, however, you can have fill *inside the pattern* that is stretched along your path. With Bend, you can have both fill and stroke as with the default None.

Path effects creating stroke shapes are added on top of any other effects that the Pen or Pencil tools add, including Spiro spline or BSpline (both Pen and Pencil) and Simplify (Pencil only). For example, if you draw a Spiro path with the Ellipse shape, then switch to the Node tool and adjust the Spiro nodes, the entire elliptical shape will bend and curl along the Spiro spline. In the Path Effect Editor dialog for this path, you will find both Spiro spline and Pattern Along Path effects stacked on top of each other. To turn a path with any path effect(s) into a regular path, use the **Object to Path** command (Shift-Ctrl-C).

Figure 14-12 shows some examples of Pencil paths created with splines and/or stroke shapes.

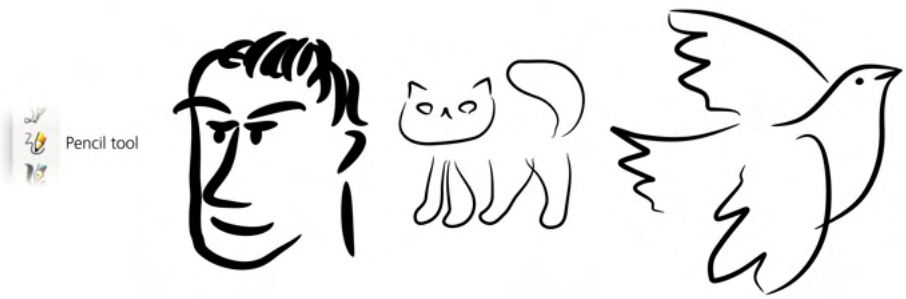


Figure 14-12: Shaped strokes with the Pencil tool

14.2 The Calligraphic Pen Tool

The name *Inkscape* is quite apt: the most sophisticated drawing tool in the program, the Calligraphic pen, feels *inky* indeed. As its name implies, this tool was initially intended for calligraphy—that is, beautiful handcrafted lettering. With time, however, it became more versatile and is now used for general artistic sketching, drawing, and inking.

Switch to the Calligraphic pen (Ctrl-F6 or C) and drag on the canvas. Inkscape will create a filled path, its width and shape depending on the angle of the stroke, the speed of dragging, and the pressure of your pen (if you are using a pressure-sensitive tablet). The result may be similar to the Pencil tool in the pressure-sensitive mode (14.1.2.2), but the Calligraphic pen has a lot more

options and behavior subtleties. The result of this tool is always a plain filled path without any path effects.

14.2.1 Width

The most important setting on the tool's controls bar is Width, specifying the width of the stroke (or more precisely, its *maximum* width; other factors may also affect width, but it will never exceed the value you set here), as shown in Figure 14-13. You can adjust the width value with the ← and → keys at any time while using the Calligraphic pen tool.

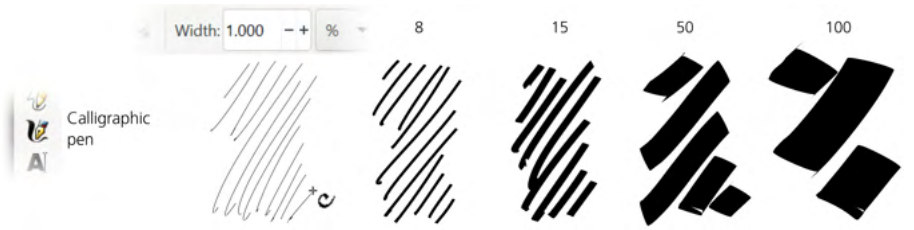


Figure 14-13: Calligraphic pen: varying width (pressure sensitivity is off)

Along with the absolute units (px, mm, and so on), you can also set the tool's width in relative units (%), which makes the width relative to the size of your document window, not to any objects or measurements in the document. In other words, if you zoom out, your stroke will be wider in absolute units but will *look* exactly the same width to you. The value of 1 always gives a hairline, and 100 always gives a stroke about 2 centimeters wide as measured on your screen.

This approach may seem strange at first, but it has the advantage that you can always have the same *feel* for the tool, no matter what your level of zoom is, providing for an intuitive workflow. First, you zoom out and lay down the main broad strokes of your drawing; then, you zoom in to add finer and finer details—all without shifting the Width back and forth.

14.2.1.1 Pressure Sensitivity

If you have a pressure-sensitive tablet as your input device, Inkscape can use the pressure information to change the width of the stroke: press lightly for a thin line, press harder for a wider brush (Figure 14-14).

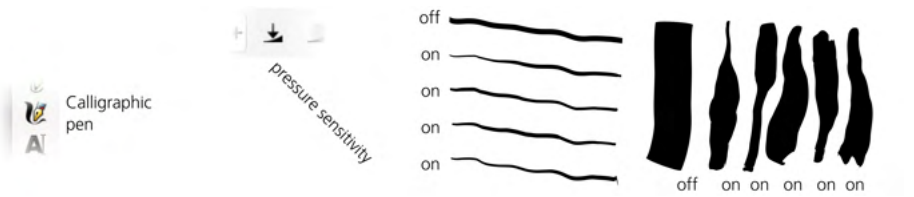


Figure 14-14: Calligraphic pen: drawing with or without pressure sensitivity (Width is the same)

To enable pressure sensitivity, click the toggle button to the right of the **Width** control. Without pressure sensitivity, the tool draws as if maximum pressure is applied.

14.2.1.2 Tracing Background

Another thing that can affect the width of the Calligraphic pen stroke—if you enable it—is the darkness of the color of the background over which you are drawing: your brush is at its thinnest when you draw over white and at its broadest when you draw over black. This is most useful when you use *guide tracking* (14.2.7) to create a hatching over an existing drawing or bitmap and want to achieve a shading effect with strokes of smoothly varying width, as shown in Figure 14-15.

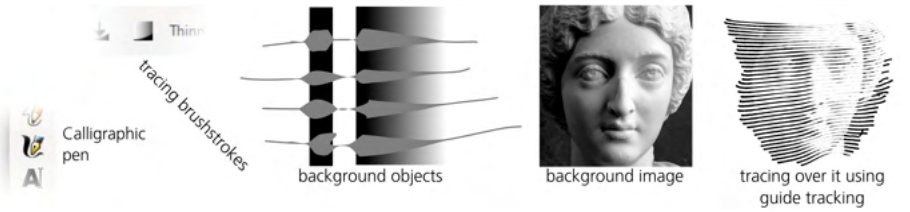


Figure 14-15: Calligraphic pen: tracking background by width

To enable tracing the background, click the next toggle button after the **Pressure Sensitivity** toggle. You can combine this setting with Pressure Sensitivity, although it usually makes sense to turn Pressure Sensitivity off when you're tracing.

14.2.1.3 Thinning by Speed

Finally, the speed of your drawing—how fast you're dragging the mouse or pen—may also affect the width of your stroke. This is controlled by the Thinning value, which takes values from –100 to 100 (Figure 14-16). Positive values make your stroke thinner as you go faster; negative values make it thicker. If you set this parameter to 0, thinning by speed is disabled.

Figure 14-16: Calligraphic pen: thinning by speed (pressure sensitivity is off)

Since a stroking movement is usually faster in the middle of a drag, the result is a shape with a fixed width at the ends and a "breathing" middle—thinner for positive values of this parameter. Pressure sensitivity usually has the opposite effect, as pressure in the middle of a stroke is typically higher than at the ends. An interplay of these two factors gives the tool its natural feel.

A real-world felt pen or pencil usually leaves a thinner trail when you move it faster; the default value for this parameter is set to 10 to emulate this behavior. However, you can also push it all the way to –100 or to 100, which produces curiously "exploding" or "imploding" strokes very much unlike anything in the real world!

14.2.2 Angle

The Angle and Fixation parameters treat the Calligraphic pen as a flat-tipped calligraphic pen that can be held at a varying angle (Figure 14-17). The Angle sets the angle of the pen's tip to the horizontal: 0 is horizontal, +90 is rotated all the way to the vertical counterclockwise, and –90 is rotated clockwise. You can also change this value by pressing the ↑ (increase) or ↓ (decrease) arrow keys.

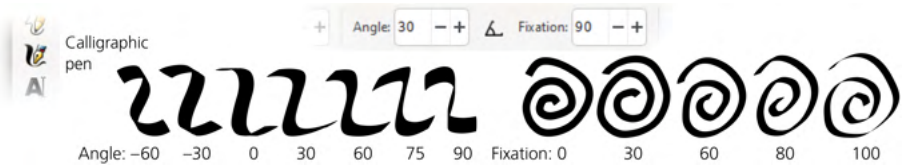


Figure 14-17: Calligraphic pen: the Angle and Fixation

Setting an angle with a nonzero fixation is most useful for producing calligraphic lettering. For most styles of calligraphy, the angle should be somewhere between 30 and 60 degrees.

NOTE *If your tablet supports pen angle detection, you can, for extra realism, bind the* Angle *parameter of the tool to the actual physical angle of your pen. For this, just press the toggle button next to the* **Angle** *slider.*

The Fixation (in the range of 0 to 100) controls how strictly the Angle is enforced. When Fixation is at its maximum, the tip is always rotated at the set angle, regardless of the direction of your stroke—so that drawing parallel to the pen angle always gives a hairline stroke, whereas drawing perpendicular to it produces maximum width. With zero Fixation, the direction of the pen is always perpendicular to the direction of movement, which makes Angle irrelevant and gives you the effect of a felt-tip pen or a round brush. Intermediate values of Fixation produce a stroke that is affected *both* by Angle and the direction of movement, in varying proportions.

14.2.3 Caps

By default, Calligraphic pen strokes are cut blunt at the ends. This is best for calligraphy work, but at other times you may want a more rounded appearance. Increase the Caps parameter to about 0.5 for slight bulges at the ends, to 1.3 for approximately round caps, and up to 5 for long, protruding caps, as shown in Figure 14-18.

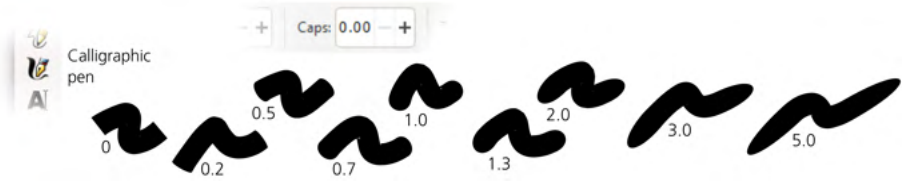


Figure 14-18: Calligraphic pen: caps of a stroke

14.2.4 Tremor, Wiggle, and Mass

What you've seen so far of the Calligraphic pen tool is useful, if maybe a little boring. It's time to add some fun! The last three parameters—Tremor, Wiggle, and Mass (all in the range of 0 to 100)—modify the behavior of the tool in some uncommon ways.

Even when using a graphics tablet with pressure sensitivity, the Calligraphy pen's brush strokes often look too smooth and computer-ish. Increasing tremor adds small-scale disturbances to the stroke for a more natural look—rough, trembling, or even splotchy. The frequency of the tremor is temporal, rather than spatial, which means if you draw faster, the roughness will be stretched along the stroke and therefore look smoother.

The Wiggle parameter also disturbs the stroke but at a larger scale, making it waver in wavy or loopy patterns, departing sometimes quite far from the actual position of the cursor, especially at sharp turns (Figure 14-19).

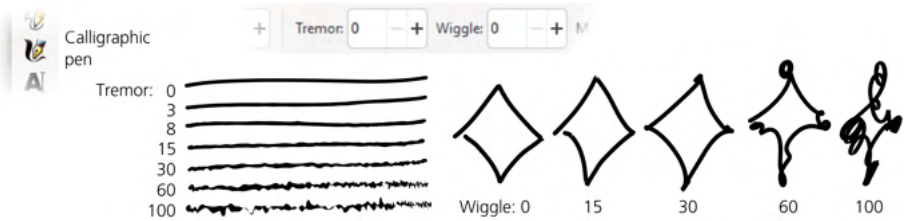


Figure 14-19: Calligraphic pen: effects of tremor (left) and wiggle (right)

Mass makes the brush lag behind the cursor, as if slowed down by inertia, resulting in smoothing of sharp corners and shortening of the fast flights of your pen. The default settings for tremor and wiggle are 0, but mass has a small nonzero default value (2 out of 100) so that the Calligraphic pen feels light and responsive but not entirely weightless.

14.2.5 Calligraphic Presets

The Calligraphic pen is one of the most settings-rich tools in Inkscape. Of course, tweaking several sliders and buttons every time you want to switch from a smooth marker pen to a wiggly brush is time-consuming. Presets are a solution to this problem: you can set multiple parameters at once by choosing one of the presets from the drop-down menu on the left end of the toolbar. Several presets come with the program:

Dip pen
: Is an imitation of a soft calligraphy pen: smooth (no tremor or wiggle), pressure-sensitive, angle fixed at 30 degrees, no caps rounding.

Marker
: Is a plain, constant-width, felt-tip marker: smooth, no pressure sensitivity, no speed thinning, no angle fixation, round caps.

Brush
: Compared to Marker, adds pressure sensitivity, negative thinning (the stroke is wider when you move faster), and some wiggle.

Wiggly
: As its name implies, features high wiggle and some tremor for a characteristic "dirty" look.

Splotchy
: Is a very wide brush with high tremor and high negative thinning, which can produce a series of "splotches" connected by thin streaks (Figure 14-16, right).

Tracing
: Is similar to Marker but with the background tracing feature turned on (14.2.1.2).

To add a new preset with the current parameters, click the button next to the presets list and provide a name for your preset.

14.2.6 Adding and Subtracting

If a single path was selected before you started drawing, holding Shift when you release the mouse button or lift the pen will *add* the new stroke you just created to the selected path, forming a new single path via the Union path operation (12.2).

Similarly, holding Alt will *subtract* (via the Difference operation) the new stroke from the selected path. This makes it easy to quickly "patch" or "carve" any path. The Cut mode of the Eraser tool (14.4) works the same, so you can also use this functionality in a separate tool without having to hold Alt.

14.2.7 Tracking a Guide Path

Drawing in the Calligraphic pen with Ctrl pressed activates the *guide tracking* feature, which causes your pen to slide at some constant distance from the edge of a selected *guide path* and lets you trace around or along that guide.

The inspiration for this feature came from the traditional line engraving techniques that, for a long time, were the only practical way of reproducing lifelike shading in black-and-white print; about a century ago, line engravings were almost completely replaced by automatic halftone screens. *Hatching*—filling space with many parallel straight or curved lines of varying width to represent gradual shading—is a very labor-intensive process. Inkscape's guide tracking, along with background tracing (14.2.1.2) and the Tweak tool (12.6), take the pain and boredom out of this ancient art. While you still need a keen eye and a lot of assiduousness, with Inkscape you can create authentic-looking line engravings, entirely in vector, in a reasonable amount of time.

One way to approximate a hatching grid is by using path interpolation (for example, with the Interpolate Sub-Paths effect, 13.3.9.2), but this method is not very flexible and produces obviously computer-generated output that lacks a "human touch." Manual hatching, on the other hand, is tedious and nearly

impossible to do uniformly enough. The guide tracking capability allows you to hatch quickly and uniformly yet not too perfectly, and you have considerable manual control over the process.

Here's how to do it. First, select the *guide path* that you will track. It may be another calligraphic stroke, any path or shape, or even a letter of a text object. Switch to the Calligraphic pen tool and, before starting to draw, press Ctrl. You will see a gray *track circle* centered at your mouse pointer touching the closest point on the selected guide path. (If you have no guide path selected, a status bar message will tell you to select it.)

Now, move your mouse closer to the guide path, so that the track circle radius is equal to the desired spacing of your hatch pattern, and start drawing along the guide path. As soon as you start drawing, the radius of the circle locks and the circle turns green. Now the circle slides along the guide path—and the actual stroke is drawn by the center of the tracking circle, *not* by your mouse point. As a result, you create a smooth stroke that goes parallel to the guide path, always at the same distance from it (Figure 14-20).

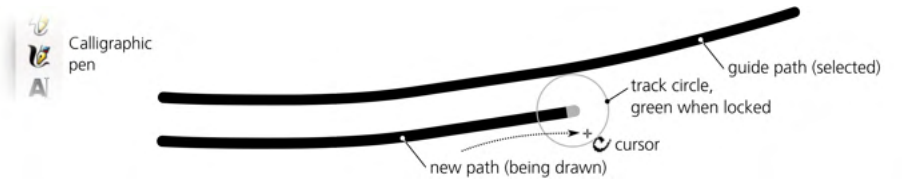


Figure 14-20: Calligraphic pen: tracking a guide path

When the stroke is ready, release your mouse button (or lift your tablet pen). However, do not let go of Ctrl yet, because so long as you have it pressed, the tool remembers the hatch spacing you set when you started drawing. Since you have just created a new stroke, that stroke object is selected instead of what was selected before—which means it now becomes the new guide path. Next, draw a second stroke along the first one, then a third one along the second, and so on. Eventually, you can fill your desired space with a nice uniform hatching, as shown in Figure 14-21.

Figure 14-21: Calligraphic pen: creating uniform hatching

The attachment to the guide path is not absolute. If your mouse pointer strays far enough from the guide path, you'll be able to tear it off (the track circle will turn from green to red) and move away—but not quite freely: the pen will maintain some inertia in the guide-tracking direction. This is intentional; for example, it allows you to continue drawing a stroke *past the end* of a guide path, in order to cover a wider area than the initial guide path would allow. With inertia, this tearing off is usually pretty smooth, but it is not possible to completely

suppress jerks. If jerking and unintended tear-offs still bother you, try increasing the Mass parameter (14.2.4).

Tracking a guide also allows for some feedback by gradually changing the tracking distance in response to your drawing behavior. If you're consistently trying to draw closer or farther from the guide than the current tracking distance, the distance will decrease or increase a bit, so you will get a hatching that is spaced slightly closer or wider.

If you've accidentally deselected your last created stroke (for example, by undoing a bad stroke), you can reselect it without leaving the Calligraphic pen tool by pressing Shift-Tab (5.11). You can combine guide tracking with adding or subtracting; press Shift-Ctrl to add the new stroke to the selected guide path or Alt-Ctrl to subtract from it.

Tracking is designed for smooth engraving patterns, so it will not work on guide tracks with sharp turns or those that are too uneven (such as calligraphic strokes with high tremor, 14.2.4). Since tracking follows the *edge* of the stroke, strokes of varying width (for example, when tracing background, 14.2.1.2) will result in gradual bending of the hatching pattern as you proceed. Finally, you can track only paths; if you have a bitmap that you want to hatch over, you will have to create the first stroke in each hatching area manually so you can then track it as a guide (Figure 14-22).

Figure 14-22: Take a bitmap (top), hatch over it with the Calligraphic pen tool, and tweak the resulting hatching with the Tweak tool (bottom).

It is convenient to use the Tweak tool with guide tracking in the Calligraphic pen. A hatching rarely comes out perfectly: loose stray ends, the wrong slant or curvature, and incorrect stroke widths (hatching that's too dark or light) are the most common problems. With the Tweak tool, you can fix those problems so you don't have to redo the hatching. Use the Shrink/Grow mode to clear the loose ends as if with an eraser, as well as to adjust stroke widths; the Push mode can bend or shape the hatching. With these powerful tools, using a photo as reference, you can create a complex and believably shaded hatching.

14.3 The Paint Bucket Tool

The concept of a "paint bucket" or "flood fill" tool is probably familiar to you from bitmap editors. In Inkscape, the Paint bucket tool works exactly as you would expect: click in any area bounded on all sides, and it will fill the area with color. Being a vector tool, however, Inkscape's Paint bucket creates a new *path* that fills in the area where you clicked, as shown in Figure 14-23.

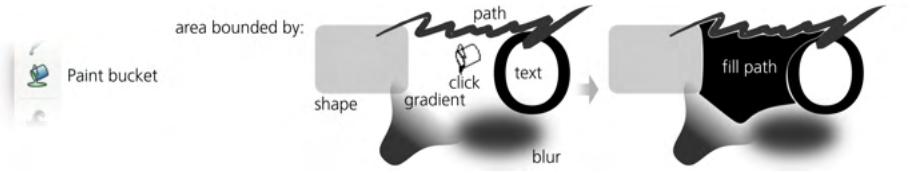


Figure 14-23: The Paint bucket filling a bounded area

On the other hand, Inkscape's Paint bucket tool is *perceptual*, not *geometric*. This means that, when looking for the boundaries around the point you clicked, it identifies as boundaries any *visible* color nonuniformities. Filling will therefore stop at gradients, blurs, and color boundaries in bitmaps, but it will ignore any paths or other objects that are fully (or almost) transparent or for whatever reason do not stand out from the background.

For example, you can scan a pencil sketch, import the bitmap into Inkscape (18.1), and quickly "trace" it by filling all of its bounded areas with colors. This convenient and interactive way of digitizing your paper drawings makes the traditional bitmap tracing (the Trace Bitmap dialog, 18.5.2) unnecessary in many cases.

Internally, the tool works by performing a bitmap flood fill on a screen-resolution rendition of the visible canvas and then tracing the resulting fill into a vector path. The *resolution* of the rendition bitmap that the tool uses is that of your current zoom; the closer you are zoomed in to an area, the higher the resolution of the flood fill. If the result of the Paint bucket fill is too imprecise, has rounded corners, or doesn't go into small nooks and crannies where it is supposed to go—just undo it, zoom in closer (but so that the entire area to fill is visible), and fill again from the same point. Conversely, if the fill leaks out through a small gap, zoom out to make the gap less visible and fill again (or adjust the automatic gap closing parameter, 14.3.5).

14.3.1 Filling Techniques

To fill a bounded area, switch to the Paint bucket tool (F7 or U) and click inside the area. If you click with Shift, the resulting path will be unioned (12.2) with the selected path; for example, if your first attempt did not fill in all of the desired area, just Shift-click the remaining corner to finish it off, as shown in Figure 14-24.

Figure 14-24: Adding a Paint bucket fill to an existing path

If you click-and-drag (your drag path is visualized by a red line), you will fill from *each of the points* that you passed by dragging. From each point, the fill spreads to its neighbors with colors similar to that point. In other words, it's like clicking with this tool at each point of the drag path and combining the results.

This way you can easily fill an area occupied by a gradient or blur—just drag from the darkest to the lightest points in the area you want to fill (Figure 14-25).

Figure 14-25: With one click, only a narrow strip of a gradient is filled; drag across an area to fill the entire gradient.

Alt-click and drag is similar to simple drag, except from each point of the drag path, the fill spreads to its neighbors (if any) whose colors are similar to the color of the *initial point* (the point where you started the drag), not the point through which you're dragging. This lets you fill a series of similarly colored yet separated areas (for example, multiple cells in a cartoon) by starting the drag in one of those areas and Alt-dragging through all the other areas.

14.3.2 Filling by Channel

When looking for color boundaries to stop at, the Paint bucket tool, by default, looks at the same colors that you see on the canvas. However, you can restrict its vision to a specific color channel. The Fill by drop-down menu, apart from the default Visible Colors, allows you to choose any of the three RGB channels (Red, Green, Blue), any of the HSL channels (Hue, Saturation, Lightness), or the Alpha channel (opacity).

For example, if you select the Red channel, even the sharpest green/blue color boundaries won't stop the fill. Choosing the Alpha channel makes the tool ignore any colors and look only at where opacity changes; for example, if you have a complex, multicolored, but fully opaque object over a transparent background, clicking this object in Alpha mode will fill up to the outline of the entire object.

14.3.3 Threshold

The Threshold parameter, with a range from 0 to 100, controls how different a point's color must be, compared to the initial click point, to stop the propagation of the fill. Zero tolerance means only the area of strictly the same color will be filled; the larger the tolerance, the larger the filled area will be, and the easier it will be for the fill to leak away into adjacent color areas. The default value is 5.

14.3.4 Growing and Shrinking

With the Grow/shrink by parameter, you can control the amount of inset or outset to be applied to the created fill path. This works much the same as the Outset and Inset path commands (12.4), except it's done automatically after every fill.

A positive Grow/shrink value causes the fill path to be larger than the filled bitmap area it represents; this is often useful for eliminating anti-aliasing gaps between the fill and its boundary. A negative value makes the path smaller, ensuring a constant-width gap between the fill and the boundary.

14.3.5 Closing Gaps

With the Close gaps parameter, you can make the Paint bucket tool ignore any gaps in the area boundaries that would normally cause the fill to spill out of the desired area. There are four levels of automatic gap closing:

- None
- Small (gaps up to 2 screen pixels in size are closed)
- Medium (gaps up to 4 screen pixels in size are closed)
- Large (gaps up to 6 screen pixels in size are closed)

Setting this parameter to anything other than None makes the tool noticeably slower, especially when filling large areas.

14.3.6 Style

Like all object-creating tools, the Paint bucket can use the last set style (11.1.2) for the objects it creates (this is the default), or it can use its own fixed style. You can switch between these modes in the tool's Preferences page (double-click the tool icon to access it). As with other tools, the style swatch on the far right of the controls bar shows the style that will be used for the next fill. To change the last set color without affecting any object, just deselect all (Escape) and click a color on the palette.

Ctrl-click in the Paint bucket tool is special. With Ctrl, the tool does not fill an area, but performs instant styling of an existing object on which you clicked (without selecting it): it simply changes that object's fill color to the current fill color of the tool. Similarly, Shift-Ctrl-clicking changes the stroke to the current stroke color of the tool (if no stroke is set, then Shift-Ctrl-clicking behaves like Ctrl-clicking). For example, you can use this to recolor some of the fill objects you created previously, without needing to select them first and without leaving the Paint bucket tool.

14.4 Eraser Tool

As the name suggests, this tool is for quickly and conveniently erasing stuff. Inkscape's Eraser tool (Shift-E) implements three different modes, represented by the three toggle buttons on the controls bar (Figure 14-26).

Delete mode

In this mode, you drag across objects (leaving a red trace), and after you release, the tool deletes those objects touched by your drag path (compare 5.8). This mode has no further options.

Cut mode

In this mode, Eraser draws a path, just as the Calligraphic pen does, and then uses the Difference Boolean operation (12.2) to subtract it from the other paths or shapes that happen to be underneath. The options in this mode are analogous to those of the Calligraphic pen, including Width (14.2.1), Thinning (14.2.1.3), Caps (14.2.3), and even Tremor and Mass (14.2.4). The last toggle button forces the tool to break apart (12.1.1) each path that was cut into disjointed parts.

Clip mode

This mode works like the Cut mode, but instead of subtracting, the tool clips (18.3) the objects you draw over. This is a better approach because, first, the clipping operation is nondestructive (you can always recover the original object using Object ▸ Clip ▸ Release), and second, clipping works not only for paths but also for other objects such as bitmaps. An object in SVG can have only a single clipping path, but the tool is smart enough to work as expected when you clip an already clipped object: it adds the new path to clip already in effect.

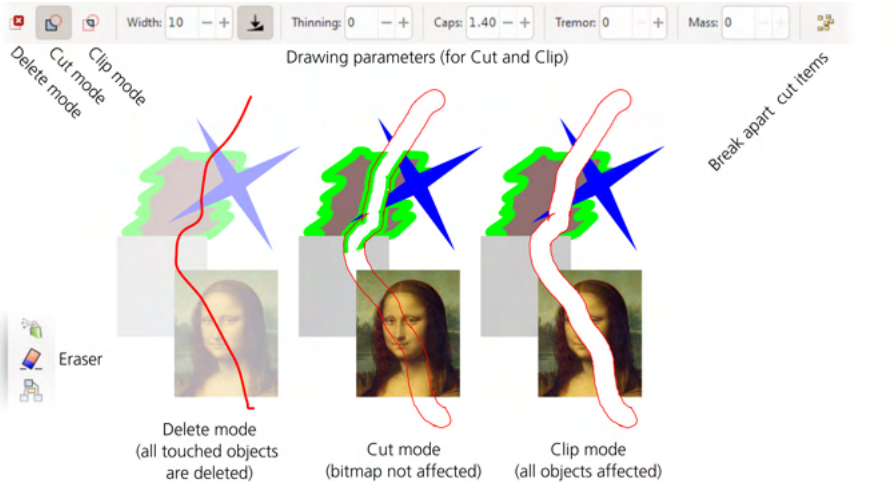


Figure 14-26: The Eraser tool

NOTE *To be affected by the Eraser (in any mode), an object must be selected. Use Ctrl-A to select all objects in the current layer (5.11).*

INKSCAPE'S ERASER VS. AI'S KNIFE

Eraser is the closest Inkscape has to the Knife tool in Adobe Illustrator.

14.5 Connector Tool

Diagramming applications, of which the best known is Microsoft Visio, are a popular class of software with a lot of overlap with vector graphics. Inkscape's Connector tool extends its functionality in this area—certainly not enough to make it a Visio replacement but sufficient for small diagramming projects.

Basically, a diagram is a vector drawing that has objects connected by different kinds of lines—to illustrate workflow sequences, links, calls, classifications, or other types of relationships. Crucially, the program is aware of what is connected to what and updates the connector lines whenever you move around the connected objects. That's exactly what the Connector tool does.

To try it out, draw a few rectangles in the Rectangle tool. You can also create text labels and group each label with its rectangle. Then switch to the Connector tool (O). You will see, as you hover your mouse over one of these objects, a white square handle in the center. Grab it, drag toward another rectangle, and drop it onto the other rectangle's central handle. That creates a straight connector line that goes between the edges of the rectangles, along the line connecting their centers (Figure 14-27). Now move the rectangles around (in Selector), and you will see the connector updating live.

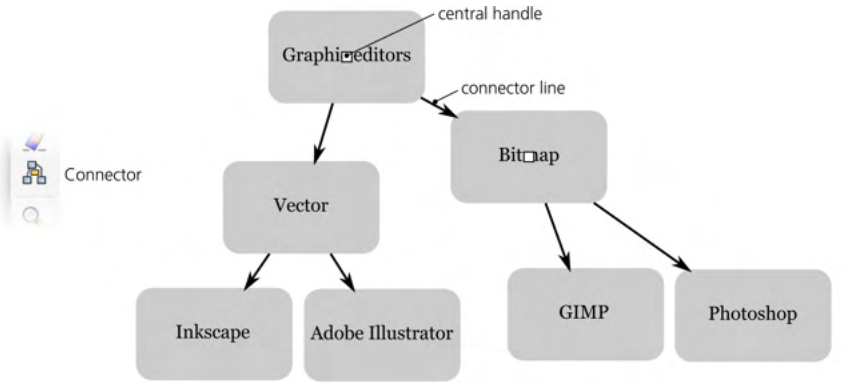


Figure 14-27: The Connector tool

The connectors are regular SVG paths that you can style as you wish—for example, by assigning arrowhead markers (9.5) to them. You can also node-edit them as paths, but you really shouldn't. As soon as one of the connected objects moves, its connectors are automatically updated, discarding any node edits you have done to them.

As you move objects around, you will naturally want to avoid situations where connector lines cross the connected (or any other) objects. The Connector tool can do that for yOu, too. The first two buttons on the tool's controls bar will enable and disable the *avoid property* for the currently selected objects. If an object has this property, connector lines will be routed in such a way as to avoid crossing or touching it. If you select all the blocks of your diagram and enable this property for all of them with one click, you will have much more freedom in moving stuff around without making the diagram unreadable. You can adjust the gaps between the objects and the connectors with the Spacing parameter (Figure 14-28).

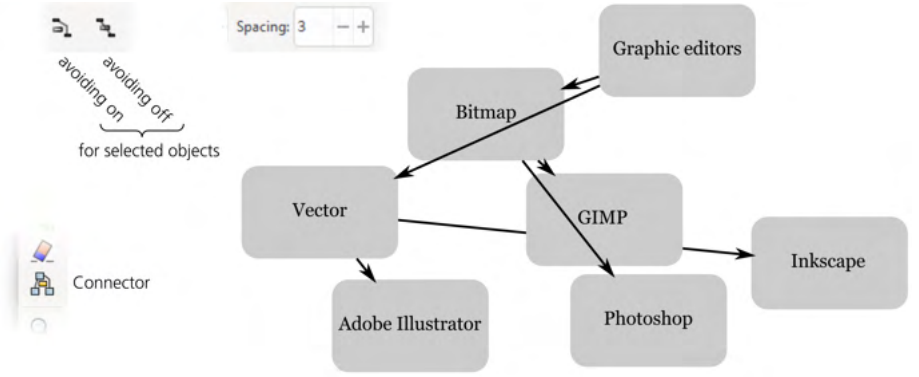


Figure 14-28: The Connector tool: enabling the avoid property on blocks

The Connector tool can also produce another style of connectors: *orthogonal*, consisting of vertical and horizontal segments, optionally with rounded corners, as shown in Figure 14-29.

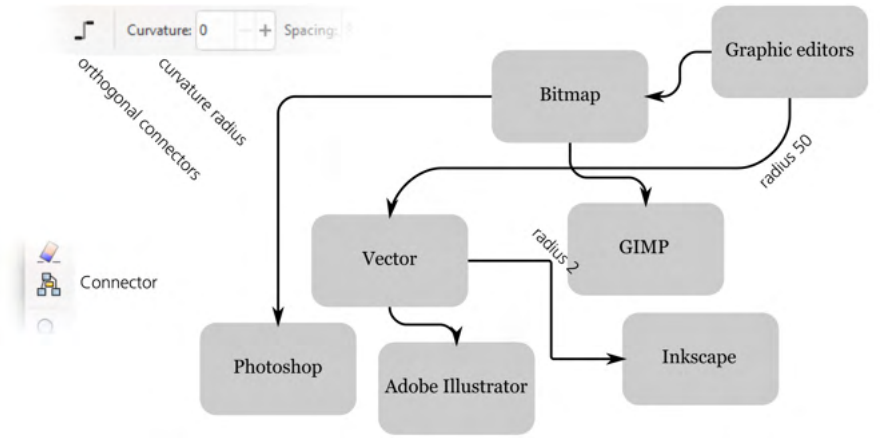


Figure 14-29: The Connector tool: orthogonal connectors

You can change the curvature radius individually for each connecter (click it in the Connector tool to select). This option can be combined with the avoid property for the diagram blocks.

PATH CONNECTORS IN SVG

Diagram connectors do not use path effects (Chapter 13), but they implement a similar approach: the visible path is automatically calculated by Inkscape from higher-level metadata stored in the connector's `svg:path` element. This metadata includes the start and end objects that the connector is attached to, the type of connector, and its curvature.

15

TEXT

As you would expect with a vector editor, any text you add to an Inkscape drawing remains fully editable *as text* at any time, no matter what styling, filters, or transformations you apply to it. Creating and editing text objects is the domain of the Text tool that allows you to type or delete words, wrap text into columns (manually or automatically), apply style to text spans, adjust kerning and spacing, and so on.

Inkscape does not compete with word processors or desktop publishing software; it is not very convenient for handling large amounts of text. Inkscape has no structural formatting tools, such as automatic headings or style libraries. Instead, as a visual design tool, Inkscape will give you complete and precise control over every glyph, word, and paragraph, in most cases without sacrificing text editability.

15.1 Basic Editing

If you have an existing text object, select it and switch to the Text tool (F8 or T). This places the text editing cursor (a blinking vertical line) at the end of the text object. You can now click anywhere within the text to position the cursor,

or you can move the cursor with the arrow keys. Or, in the Selector tool, just double-click a text object—this will switch tools, select the object, and place the cursor where you clicked, all at the same time.

Clicking or double-clicking a text object is only possible when your mouse cursor has an "I" shape and a blue frame appears around the text—these are the indications that you are over a clickable text object (Figure 15-1).

Figure 15-1: Placing the text editing cursor by clicking

Typing and editing text works the same way in Inkscape as in a word processing application. Use Delete or Backspace to erase characters (including newlines); press Enter to start a new line. You can move the text cursor with the arrow keys (with Ctrl to jump by words), Home and End (with Ctrl to move to the beginning or end of the text object), and Page Up and Page Down. The canvas will automatically scroll, when necessary, so that the text cursor is always visible.

If, for some reason, editing text on the canvas is inconvenient, open the **Text and Font** dialog (15.3.2) and, on the **Text** tab, edit the selected text object in a standard text area. Unfortunately, clicking Apply loses any formatting you may have applied to spans within your text (15.3.1), but if you just open the dialog to read the text or copy it to the clipboard without making any changes, it is perfectly safe.

15.1.1 Selecting

You can select inside a text object by dragging *over* some characters (as with clicking, you must start with a character body, not empty space) or by moving the text editing cursor with the Shift-arrow keys (Figure 15-2). This *text selection*, visible as a greenish-blue rectangle overlay, is not to be confused with the program-wide *object selection* (Chapter 5); text selection is unique to the Text tool and allows you to style, kern, or delete spans inside the selected text object. When editing text with the Text tool, pressing Ctrl-A selects not all objects but all text inside this text object.

Figure 15-2: Text selection with the Text tool

Copying a text selection with Ctrl-C (or cutting with Ctrl-X) and pasting at the cursor location with Ctrl-V is an easy way to move pieces of text around—within the same text object, among text objects in the same document, across

documents, or even between Inkscape and other applications. For example, you can copy a piece of text in a word processor and insert it into a text object in Inkscape. Copy/paste in Inkscape never transfers formatting, only textual content.

15.2 Types of Text Objects

How do you create a new text object? This depends on the type of object you want to create. Text objects in Inkscape can belong to different types that are broadly similar but differ in details of their capabilities and behavior.

15.2.1 Regular Text

To create a *regular text object*, switch to the Text tool (F8), click the canvas (not an existing text object!) to place a text cursor, and start typing. (If you drag instead of click, you will create a text-in-a-shape object, 15.2.2.) Press Enter to start a new line.

Once you have typed at least one character, the new text object is added to the document. At that point, you can switch to any other tool to deal with the newly created object as you would with any other object—for example, transform it with the Selector tool (Figure 15-3), paint it by clicking a palette color (this can be done in any tool), or draw a gradient across it with the Gradient tool.

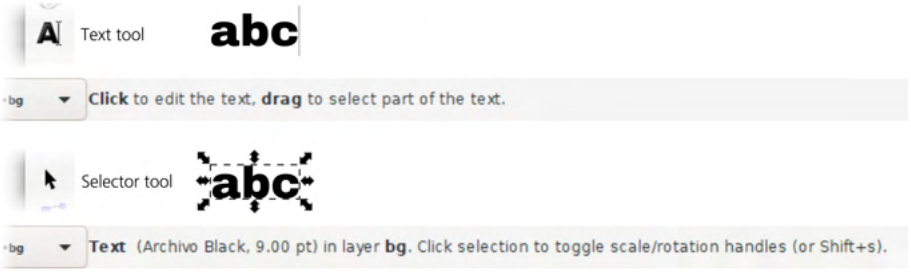


Figure 15-3: The status bar description of a text object in the Text and Selector tools

REGULAR TEXT IN SVG

A regular text object is SVG's standard `text` element. Each line in a multiline object is a `tspan` with a `sodipodi:role="line"` attribute and explicit x and y attributes that Inkscape calculates automatically, based on the coordinates of the root text object and line spacing (15.4.5).

15.2.1.1 Line Wrapping in Regular Text

In regular text objects, by default there's no automatic line wrapping; you need to press Enter to start a new line. If you don't do that and just keep typing, there's no limit to the length of the line.

To enable automatic line wrapping in a regular text object, notice the diamond-shaped handle at the right end of the text. (The smaller square mark on the left is the baseline origin mark, which is not draggable.) As soon as the handle is moved away from its default position, the text displays two vertical guides that define the column into which the text is flown (these guides are visible only in the Text tool). You can adjust the width by dragging the handle. Any text line wider than the column is wrapped, as shown in Figure 15-4.

Figure 15-4: Wrapping long lines in a regular text object

In right-to-left text (15.4.1), the baseline origin mark is on the right and the column width handle is on the left, and you can adjust only the guide on the left of the column. With vertical text, the guides are horizontal; you can reflow text by moving the bottom guide.

Regular text can be aligned to the left, center, or right (15.4.2), but full justification is not available even when the text is autowrapped into a column. If you need a fully justified column of text, use text-in-a-shape (15.2.2).

If the column is narrower than the longest word of your text, Inkscape faces the tough choice of what to do with the words that don't fit. Inkscape has no idea how to break words properly (if only because word-breaking rules depend on language, and Inkscape does not know what language your text is in); one thing it does try is to break any hyphenated words at the hyphen. Eventually, Inkscape gives up and truncates the text—hiding everything starting from the first word that doesn't fit; such a truncated text object displays red column guides instead of blue. To fix a truncated text object, make the column wider, or reduce the font size, or edit the text so that all words fit.

WRAPPED TEXT IN SVG

If you have set a column width on regular text, its `text` element gets the `inline-width` CSS property that specifies the width of the column. However, Inkscape (perhaps with good reason) does not rely on other SVG viewers to interpret this attribute (which is new in the SVG 2.0 draft) correctly: it still creates automatic `tspans` with explicit x and y attributes (but without `sodipodi:role="line"`) for each line break.

15.2.2 Text-in-a-Shape

As you have seen, a regular text object may or may not have a specific column width, depending on whether you have dragged the diamond handle. By contrast, a *text-in-a-shape* always has its own intrinsic width and height—more generally, a *frame*—and it automatically wraps the text to fill its frame. When typing in such a text object, pressing Enter starts a new paragraph. A text-in-a-shape can be *truncated* if it is too long to fit completely into the frame (compare 15.2.1.1).

15.2.2.1 Internal Frame

The easiest way to create a text-in-a-shape is by *dragging*, not clicking, over the canvas using the Text tool. This creates a rectangular frame, much like dragging with the Rectangle tool creates a rectangle (Figure 15-5). After releasing the mouse, you can type or paste your text, which will wrap upon reaching the edge of the frame. You can also drag the handle in the bottom-right corner to resize the frame and see the text automatically reflow to the new size.

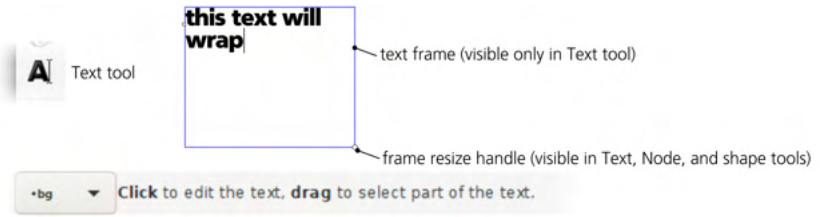


Figure 15-5: Text-in-a-shape with an internal frame

Text is wrapped at word boundaries; automatic hyphenation is not supported. If there's more text than fits the frame, extra text (starting at a word boundary) will be truncated—hidden but not lost; if you reduce font size, or delete some text, the end will move in and become visible.

This kind of text object is said to have an *internal frame*: its rectangular frame is part of the object itself, not a separate object. For example, when you transform the text, it is transformed with its frame as a whole.

NOTE *The bounding box (4.3) of a text-in-a-shape object covers only the visible characters of the text itself, regardless of the size of its frame.*

15.2.2.2 External Frame

Instead of its own internal rectangular frame, a text object can be linked to any external path as its frame. For this, select two objects—any text (regular or text-in-a-shape) and a path—and choose **Text ▸ Flow into Frame**, as shown in Figure 15-6.

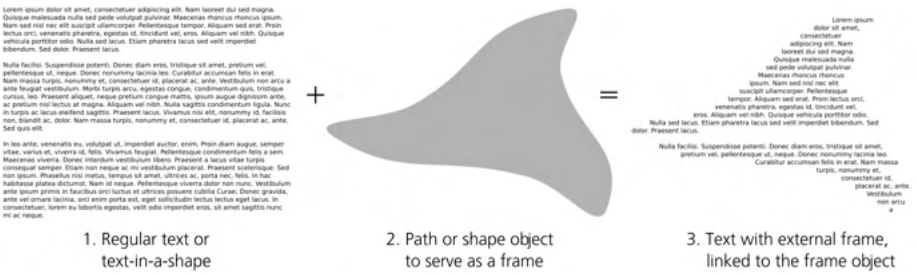


Figure 15-6: A text object with an external frame

This command does not delete or hide the frame object; the text and the frame remain separate, and the text is said to be *linked* to its frame, much like a clone is linked to its original (Chapter 16). Editing the shape of the frame object forces the linked text to reflow to the changed shape, but styling the frame (for example, hiding it by making fully transparent) does not affect the text.

You can move or transform the linked text separately from the frame, but moving or transforming the frame affects the text as well (compare with the Move according to transform clone compensation mode, 16.2). Selecting a linked text object and pressing Shift-D will select its frame, just as it does with a clone and its original (16.4).

To get rid of the frame, use the **Unflow** command in the **Text** menu. This deletes the text-in-a-shape's internal frame (or breaks the link to the external frame) and converts it into a regular text object. Each paragraph becomes a line.

TEXT-IN-A-SHAPE IN SVG

Any text-in-a-shape (with either an internal or external frame) is represented in SVG by a `text` element with the `shape-inside` CSS property that refers to the frame object. Inside, Inkscape emulates wrapping with `tspans` so that the text looks the same even in browsers that don't support `shape-inside` (which is new in SVG 2.0).

15.2.3 Flowed Text

If you open a file created in an old version of Inkscape, you may find another type of text object that looks and behaves almost exactly like text-in-a-shape but is called *flowed text* in the status bar description. This type of object (represented in SVG by a non-standard `svg:flowRoot` element and its children) remained, for many years, the only way to create automatically wrapped text in Inkscape. This kind of flowed text was never supported outside Inkscape, and while you can still view and edit flowed text objects in old Inkscape files, it is a good idea to convert them to regular wrapped text objects by the Text ▸ Convert to Text command.

15.2.4 Text on a Path

Aside from using a path as a frame (15.2.2.2), a single-line text object in Inkscape can link to a path to use it as a guide to bend the text's baseline. Just select both the text and the path and choose **Text ▸ Put on Path**, as shown in Figure 15-7.

Figure 15-7: Putting text on a path

This connection is live—both the text and the path remain editable, and reshaping the path makes the text follow the new shape. If the text is longer than the path, its end is hidden (but it is still there: if you delete some text at the beginning, the end will move in and become visible). If the path consists of more than one subpath (12.1.1), the text will continue from one subpath to the next at character boundaries (but not word boundaries).

NOTE *Text on a path can only displace and rotate individual glyphs (letters) but cannot deform them; if what you need is curvilinear deformation of letter shapes, convert text to path (15.5) and use one of the deformation path effects (13.3.3).*

It makes no difference whether the path has any stroke or fill; the only thing that matters for text is its geometric shape. Path direction also matters: if you want to put text on the other side of the path and in the reverse direction, choose **Path ▸ Reverse** on the path, as Figure 15-8 demonstrates.

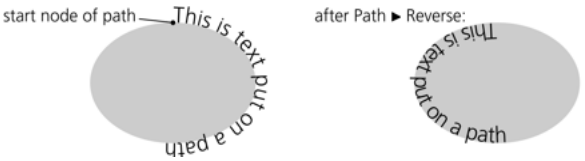


Figure 15-8: Path direction affects the text on the path.

You can easily hide the guide path by making it fully transparent or by removing both its fill and stroke. Transforming the path (for example, moving it around) affects its linked text-on-a-path; however, you can scale, rotate, or move the text object away from its path and it will keep this separate transformation without breaking the link.

More than one text object can be linked to the same path. Use **Text ▸ Remove from Path** to convert text on a path into regular linear text, cutting its link to the path.

TEXT ON A PATH IN SVG

Text on a path is a standard SVG feature. In SVG, it has a `textPath` child element under `text` that contains the actual text and links to the path object with an `xlink:href` attribute (compare 16.1).

15.3 Styling Text

In many ways, text objects are no different from other object types: you can transform and style (choose fill or stroke, adjust opacity, apply filters) a text object as a whole without ever going into the Text tool (and, of course, without losing the ability to edit the text). However, only the Text tool allows you to change style properties specific to text (such as font family and font size) as well as apply style to *spans* (fragments) inside a text object.

TEXT SPANS IN SVG

To have a different style, a span of text must be enclosed in a `tspan` element. You don't need to think about it: for any overlapping or nested spans that you create or delete, Inkscape manages the `tspan` elements automatically.

The two places where you can change text styles are the controls bar of the Text tool (above the canvas) and the Text and Font dialog (Shift-Ctrl-T). The former is faster and more convenient, while the latter provides access to additional options.

15.3.1 Non-Text Style Properties

When you have a text span selected in the Text tool, most (but not all) style-changing commands and style-reporting UI elements apply to that text span, not to the entire text object. For example, you can easily change the color of a span of text by selecting it and clicking a color swatch in the palette, using the Fill and Stroke dialog (8.2) or color gestures (8.7), as shown in Figure 15-9. Similarly, you can assign or remove a stroke or adjust opacity of the span. However, blur or other filters cannot be applied to a span; if you try this, the entire text object will be blurred or filtered.

Assigning a gradient or pattern fill to a span is possible, though it is tricky. You need to create a separate object with the gradient or pattern you want, copy it (Ctrl-C), then select a text span with the Text tool, and paste the style (Shift-Ctrl-V) onto it. Unfortunately, you cannot then edit such a gradient on a span using handles in the Gradient tool.

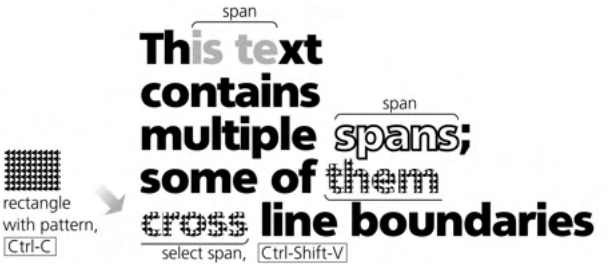


Figure 15-9: Styling text spans

In general, the paste style trick is handy when styling text spans—it allows you to transfer styles (including not only color but also font, size, letter spacing, and so on) between objects and parts of your text. To copy a style from a text span, you don't even need to select it; just place the text editing cursor anywhere inside the span and press Ctrl-C.

15.3.2 Fonts and Variants

Inkscape allows you to use any outline fonts (TrueType, OpenType, Type 1) that are installed in your operating system. You can also provide additional fonts, that only Inkscape will be able to use, by placing them in the folder *inkscape/fonts* under the Inkscape data folder that you can look up in the Preferences, System page. In addition, you can instruct Inkscape to look into other font folders by listing them in the Preferences, Tools ▸ Text page under Additional font directories.

A list of all font families (not individual fonts), with graphic samples for each family, opens from the controls bar of the Text tool; you can choose a family by scrolling through the list. If you know the name of the font you need, click to place the cursor in the editing field (or press Alt-X) and start typing; a drop-down list of possible completions will appear (Figure 15-10).

Most fonts contain variants (styles) within the same family; apart from italic/slanted and weight variations (light, medium, bold, heavy, and so on), they may include variations along the width axis (condensed or stretched). On

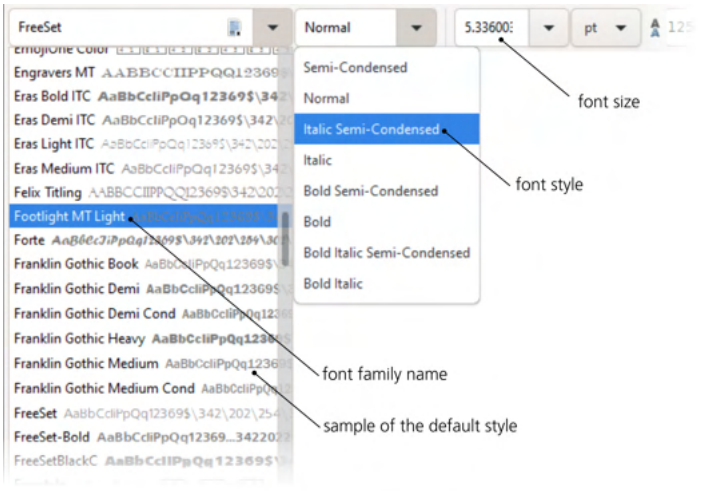


Figure 15-10: Choosing a font family and style in the Text tool

the controls bar, the list of styles for the selected family is to the right of the family drop-down. When editing, the Ctrl-I and Ctrl-B shortcuts will try to toggle the selected text to the italic or slanted (with the same weight, if possible) or to the bold style of the current font.

The same lists of font families and styles are also available in the Text and Font dialog (Shift-Ctrl-T, Figure 15-11). Here, instead of a small generic sample, you can see a preview of the selected text object in the chosen font with the chosen font size. Once you choose the style you want in the dialog, click **Apply** for the changes to take effect.

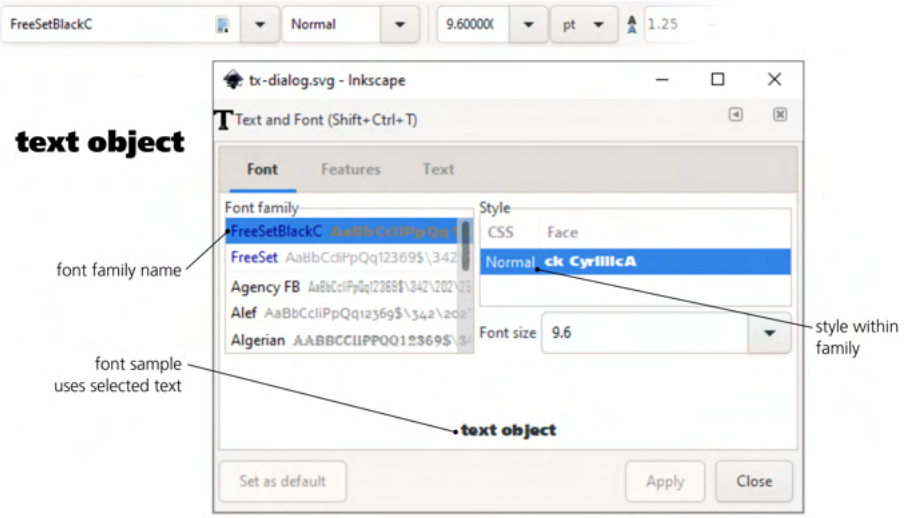


Figure 15-11: The Text and Font dialog

NOTE *With a text object selected, the font you choose is applied to that object; if you select a span inside that text, only that span will be styled. With nothing selected, choosing a font style in the controls bar changes the default style of the Text tool that will be applied to newly created text objects. In the* **Text and Font** *dialog, click the* **Set as default** *button to make the style chosen in the dialog the default style.*

15.3.2.1 Variable fonts

A *variable* font, instead of providing distinct styles as separate font files, has one or more *design axes* that allow you to smoothly vary some font parameter—most commonly weight or width. This is a feature of OpenType fonts (OpenType is a successor to TrueType); a number of freely available fonts, including those on Google Fonts (*https://fonts.google.com/*), make use of it. Inkscape supports variable fonts in its Text and Font dialog: when a variable font is selected, its design axis (or axes) is shown as slider (or sliders) above the font sample that reflects, live, the values of the variable parameters.

15.3.3 Font Size

In many cases, you don't even need to set font size numerically; you can simply scale an entire text object with the Selector tool (6.2) or keyboard shortcuts. Otherwise, use the font size controls on the controls bar (with a unit chooser) or in the Text and Font dialog (which always uses px units). Both font size controls will then take this scaling into account when displaying font sizes; for example, if you set a text object to 12 px font size but then scale it up twice, you will see its font size reported as 24 px.

15.4 Text Layout

15.4.1 Writing Mode and Orientation

Not all languages write left to right, top to bottom like English. Some languages are written right-to-left (for example, Hebrew), and some prefer vertical placement of glyphs (for example, Japanese).

Of the three drop-down switches that control language-specific text parameters (at the far right end of the Text tool's controls bar), the first one switches the text to vertical (top to bottom). For vertical text, it also controls the *block progression* (blocks in this context refer to paragraphs or lines within a paragraph) that can go right to left or left to right. This gives three options in this switch: horizontal text, vertical text with right-to-left block progression, and vertical text with left-to-right block progression.

The second switch controls the orientation of individual glyphs inside vertical text. This makes a distinction between those characters that are normally always in horizontal text (such as those in western languages) and those that are usual in vertical text (such as Japanese characters). The three options—mixed, upright, and sideways—are illustrated in Figure 15-12.

The last of the three buttons sets the inline writing direction of the text (left-to-right or right-to-left). Normally, right-to-left text (such as Hebrew) works just

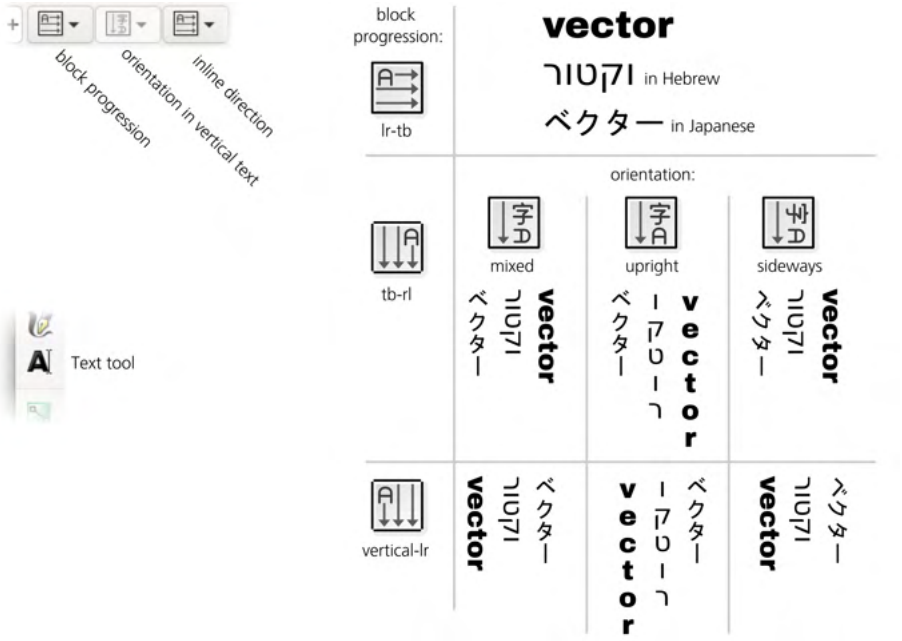


Figure 15-12: Text direction, block progression, and glyph orientation options

as well without touching this option; it does not actually reverse the order of characters in your string but marks your text as right-to-left as well as reverses alignment and changes the way cursor movement and selection keys work.

15.4.2 Alignment

Each text object has a certain alignment: *left* (default), *right*, or *center*, which you can choose with the drop-down on the Text tool's controls bar (Figure 15-13). For example, when editing left-aligned text, pressing Enter moves the cursor horizontally to the *beginning* of the next line, and the text you type grows to the right of this alignment edge, moving the cursor with it. In contrast, with right-aligned text, Enter moves the cursor under the *end* of the previous line, and the text you type shifts to the left while the cursor remains in place.

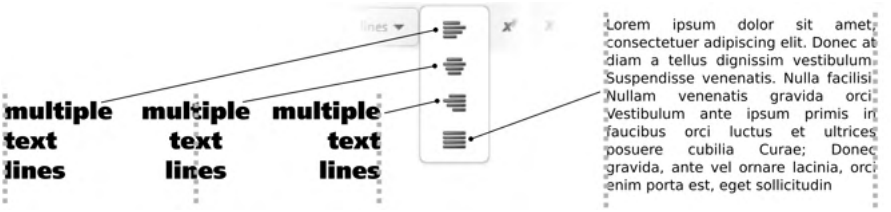


Figure 15-13: Text alignment and justification

For text-in-a-shape, there's an additional alignment option: *justify*, which expands the spaces in each line to make both edges of the text column line up.

15.4.3 Subscript and Superscript

To turn part of your text into superscript, select it and click the superscript button (**x^{y}**) on the controls bar. For example, type `1.496 × 1011`, then select the last two characters and turn them into a superscript to record the Earth to Sun distance in meters in scientific notation: `1.496 × 10`^{11}. Another button produces subscript; if you want to have both subscript and superscript attached to the same character, as in T^{2}_{2n+1}, you will likely need to apply some horizontal kerning (15.4.4) to make it look good.

15.4.4 Kerning

When working with text, Inkscape gives you the freedom of moving each individual character and adjusting the spacing between letters.

Position your cursor before a character and press Alt-→. The rest of the line to the right of the cursor will move to the right by 1 screen pixel at the current zoom; with Shift-Alt, the arrow keys move 10 times the distance of Alt (compare 6.5.1). That's an interactive way to adjust the *kerning interval* between the two characters; you can also do it numerically via the Horizontal kerning value in the controls bar (which works in px units).

You can kern any pair of characters closer together or further apart to achieve the best overall balance and visual rhythm in a line of text. You can also kern spaces to move *words* closer together or farther apart as needed. To visualize the kerning you have added, simply select the text (Ctrl-A). Whenever two letters were manually kerned closer together, their selection rectangles will overlap, and you will see a darker band; contrastingly, a kerned-out pair of characters will have a gap in the selection overlay between them (Figure 15-14). The width of the darker band or gap corresponds exactly to the amount of kerning you have added at that point.

By zooming in closer, you can make finer kerning adjustments. Don't get carried away, however; always zoom out to see how your interval looks in the context of the entire text line.

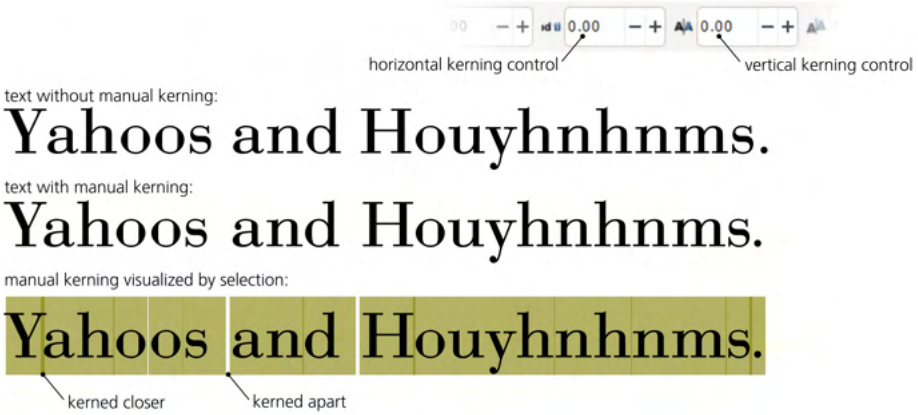


Figure 15-14: Adjusting kerning intervals in a string

Horizontal kerning is especially useful for text on a path. As you can see in Figure 15-7 on page 322, letters tend to be too close together in concave bends and too far apart on convex arcs. With manual kerning, it is easy to counteract this effect and make the characters spread evenly along the curve.

Many fonts contain built-in kerning instructions; for example, a font may specify that whenever *Y* and *a* are next to each other, they are kerned together by 0.03 of the font size. Inkscape honors these automatic font kerning instructions. However, if you try to adjust kerning in such a pair of characters manually, you will disable the automatic kern. This is why the first Alt-← to kern *Y* and *a* closer together may result in these two letters jumping, unexpectedly, further apart. Don't worry; just keep pressing Alt-← to achieve the interval you need.

You can also kern characters *vertically* by pressing Alt-↑ and Alt-↓ or via another controls bar widget. A combination of horizontal and vertical kerning gives you an absolute freedom in positioning individual letters in a text string, as Figure 15-15 demonstrates.

Figure 15-15: Combining horizontal and vertical kerning

If you select one or more characters (with a mouse drag or by pressing Shift-arrows), kerning shortcuts will effectively move the selected fragment relative to the rest of the text by inserting two opposite kerns before and after it, as shown in Figure 15-16.

Figure 15-16: Moving a selection via kerning shortcuts

Finally, you can *rotate* any character in a text object by pressing Alt-[or Alt-] when the cursor is before that character (Figure 15-17).

Figure 15-17: Rotating characters

This rotation is around the character's *baseline origin*—its leftmost point on the *baseline* (the line on which lie the bottoms of most letters without extenders, such as *i* or *m*). The baseline origin of the first character in a selected text object is shown on the canvas as a little square in the Selector and Text tools.

NOTE *Horizontal and vertical kerning as well as rotation are not available in a text-in-a-shape, but only in regular text (without automatic wrapping) and text on a path.*

KERNING IN SVG

In SVG, kerning information is stored in the `dx` (horizontal), `dy` (vertical), and `rot` (rotation) attributes of the `text` element. Each attribute contains a list of space-separated numbers, each number corresponding to the character at the same position. For example, `dx="0 0 -2"` means that the third character in this text is moved by 2 px to the left.

15.4.5 Letter, Word, and Line Spacing

What if you want to kern all characters in your text—not just some particular pair—closer together or farther apart? This kind of adjustment is called *letter spacing* (in other software, it may be called *tracking*). Select all or part of the text and press Alt-> to space the characters apart or Alt-< to move them closer together (these are the same shortcuts that scale an object in the Selector tool), or modify the **Spacing between letters** value in the Text tool's controls bar (Figure 15-18).

Figure 15-18: Spacing letters further apart (top) or squeezing them together (bottom)

The ideal amount of letter spacing depends on the final viewing size of your text. Usually, a smaller font size requires more airy spacing, while larger text should be tighter.

Similarly, you can adjust the spacing between words—that is, change the default width of the space character. For example, if you put 10 in the Spacing between words field on the controls bar, each space in the selected text will be 10 px wider than the default.

To adjust the spacing between lines in a text object, press Ctrl-Alt-> or Ctrl-Alt-<. On the controls bar, the corresponding Spacing between baselines value can be expressed in relative units (lines or %, which refer to the size of the font) or in absolute units such as inches (Figure 15-19). For example, text in a 20 px font with line spacing set to 1.2 lines will have 24 px between adjacent baselines. Changing line spacing affects the entire text object, regardless of whether any text is selected.

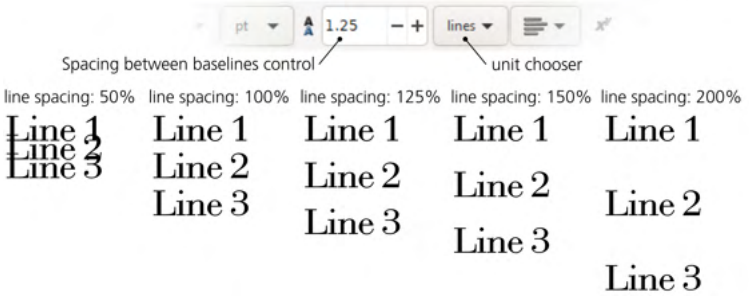


Figure 15-19: Adjusting line spacing

15.4.6 Ligatures

Some fonts have, in addition to single characters, *ligatures*—combined glyphs for pairs (or triples, or more) of certain characters. By default, Inkscape uses the ligatures defined in the font. For example, the Alef font has a ligature for the characters "fi," as Figure 15-20 demonstrates.

Figure 15-20: Ligatures in Inkscape

Having a single glyph for two or more characters, however, makes Inkscape behave weirdly when you move cursor with the arrow keys: you still need two keystrokes to pass the "fi" ligature (because it's two characters), but the visible cursor doesn't move on one of these keystrokes because Inkscape can't place the cursor inside a ligature. You need to imagine a "virtual" cursor between "f" and "i" when you start before the "fi" and press → once but the visible cursor doesn't move.

This suggests a way to get rid of a ligature you don't want. When the "virtual" cursor is inside the ligature, press Alt-← or Alt-→ once. Inkscape inserts a kern, which breaks the ligature.

To look up which types of ligatures a font supports, open the **Text and Font** dialog (Shift-Ctrl-T). On the Features tab for the selected font, you will see check marks against some of the ligature classes: Common (this includes the "fi," "fl," and some others), Discretionary ("sp," "st," "rt," and others in certain decorative fonts), Historical (mostly those with an old-style "long s"), and Contextual (those that are supposed to be used only in certain contexts but not others).

15.4.7 Special Characters

From time to time, you may need to enter a character that your keyboard doesn't have a key for. In the Unicode Characters dialog (from the Text menu), you can look up and insert any Unicode character in any of the fonts available to Inkscape.

Unicode is a worldwide standard that covers all existing and most historical alphabets, as well as a plethora of other special characters. The wealth of Unicode

characters can be divided by *script* (for example, Common, Cyrillic, or Katakana) and, more narrowly, by *range* (for example, General Punctuation, Cyrillic Supplement, or Katakana Phonetic Extensions). The best place to look up the details of the extremely complex Unicode standard is *https://unicode.org/*.

If you leave the Script and Range choosers in the Unicode Characters dialog set to all, the dialog will show you all the characters that exist in the selected font (Figure 15-21). Double-click the character you need to place it into the editing field in the bottom left, and then click **Append** to add it to the current text object.

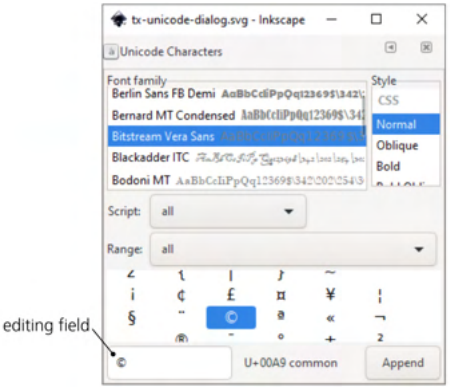


Figure 15-21: The Unicode Characters dialog

NOTE *The character is inserted not in the font in which you found it, but in the font that your text object currently has. If that font does not have the character, Inkscape tries to find a substitute font that does; in most cases, this ends up being the default sans serif font.*

At the bottom, the Unicode Characters dialog displays the hexadecimal Unicode number of the character you select in the table (for example, 00A9 for the copyright mark character). If you know that number, you can insert the character without opening the dialog. Just press Ctrl-U while editing a text object, type the number (watch the status bar for feedback), and press Enter. The character will be inserted at the text cursor.

Here are a few commonly used special characters:

Name	Character	Hexadecimal number
em dash	—	2014
en dash	–	2013
left curly double quote	“	201C
right curly double quote	”	201D
left curly single quote	‘	2018
right curly single quote	’	2019
left double guillemet	«	00AB
right double guillemet	»	00BB
ellipsis	…	2026
multiplication	×	00D7
copyright	©	00A9
registered sign	®	00AE
trademark	™	2122
round bullet	•	2022

15.5 Converting Text to Path

For all of Inkscape's text-editing capabilities, sometimes you will want to convert text to path—for example, to edit the shapes of individual letters, to clip something with your text (that requires a path), or to send your SVG document to someone who may not have the font you're using. The command Path ▸ Object to Path (Shift-Ctrl-C) will work on text objects just as it does on shapes (13.1).

Unlike a shape, however, a text object becomes not a single path but a *group* of paths, one for each glyph (character) of the original text. This allows Inkscape to fully preserve the appearance of the text, including any styling applied to spans inside it. Now you can select any individual letter with Ctrl-click in the Selector or with a simple click in the Node tool. If you really need a single path for the entire text object, choose **Path ▸ Object to Path** as above, then use **Path ▸ Combine** (Ctrl-K) to combine all of the characters into a single path object with a uniform style.

15.6 Spellcheck

Inkscape's built-in spellchecker can use up to three dictionaries at the same time. To set it up, go to the **Spellcheck** page of the **Preferences** dialog (Figure 15-22).

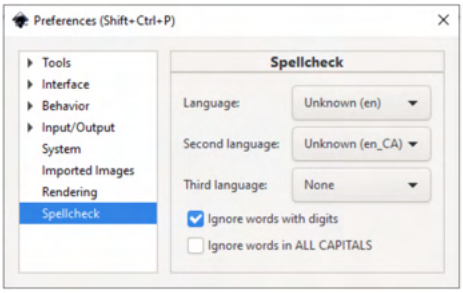


Figure 15-22: Setting up the Inkscape spellchecker in Preferences

For example, you can use French as your primary language but add English and Russian as second and third languages. This way, you can check spelling in any of those languages or a mix of them, and only the words that are missing in all three languages will be flagged as misspelled. If you need only a single language, leave the second and third options set to **None**.

NOTE *On Linux, you need to install Aspell dictionaries for the languages you want to check; use your distribution's standard software installer for this. On Windows, Inkscape comes prepackaged with dictionaries for several languages (English, French, German, Russian, and Spanish).*

Many languages have more than one dictionary. For example, English (en) has variants for USA (en_US), UK (en_GB), Canada (en_CA), and Australia (en_AU); the UK variant additionally breaks into subvariants by the preferred verb suffix (en_GB-ise or en_GB-ize), and so on.

Once you invoke the spellchecker with Ctrl-Alt-K (or Text ▸ Check Spelling), it checks all the visible text objects in your document (they need not be selected) in turn, going top to bottom and left to right. When a misspelled word is found, Inkscape displays a red frame around the word and lists the suggestions in the dialog, as shown in Figure 15-23.

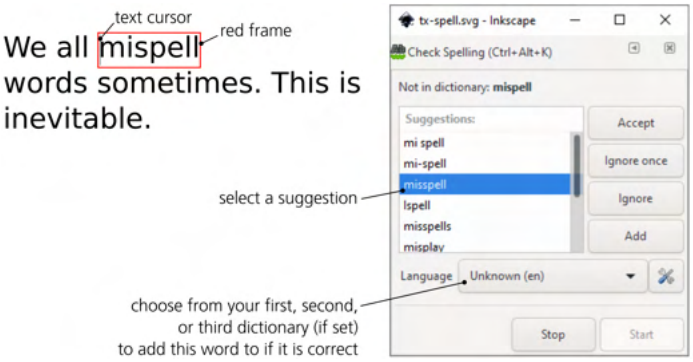


Figure 15-23: Spellchecking a document

The object with the misspelling is selected; if you are using the Text tool, the editing cursor is placed at the beginning of the misspelled word. Here's what you can do next:

- Choose one of the listed suggestions and Accept it.
- Ignore once: the next time this word is encountered, the spellchecker will flag it again.
- Ignore the word *for the rest of this session*: any other instances of the word during this check will be ignored, but the next time you run the spellchecker, it will flag it again.
- Add the word to one of the active dictionaries so that it is never flagged as misspelled again in this or subsequent sessions.

Also, since the dialog does not lock the Inkscape window, you can simply edit the word with the Text tool, as you would normally do. Once you edit it to something acceptable, the spellchecker will automatically turn off the red frame and continue checking the document.

You can stop the spellcheck at any time by clicking **Stop** (or simply closing the **Check Spelling** dialog). Restart a stopped check by clicking **Start** again.

15.7 Text Extensions

A number of extensions in Extensions ▸ Text are useful for working with text. They include:

- A group of case manipulation extensions will convert selected text objects to UPPERCASE, lowercase, Sentence case (capitalizes first letter of each sentence), Title Case (capitalizes first letter of each word), fLIP cASE (converts lowercase

to uppercase and vice versa), or rANdOm CasE. All these extensions work only on selected objects.

- A simple Replace Font extension will search and replace a specific font in the document, or replace all fonts with a single one, or just give you a list of the fonts used in the document.
- Split text will break the selected text object into lines, words, or individual characters, adding them to the document as separate text objects (unfortunately, it does not preserve positions of words and characters).

15.8 Creating Fonts

Creating new fonts is not for a casual user. The sheer amount of terminology you would need to master is daunting. And yet it may be tempting to have an original font of your own creation that you can use anywhere. The best part is that you can do it (mostly) in Inkscape, too!

Inkscape's font editing functionality is based on SVG 1.1, which included provisions for storing custom fonts right in an SVG document that can use them for text. Unfortunately, this part of the standard was removed in SVG 2.0 because it had failed to achieve widespread support in browsers. Still, Inkscape supports creating SVG 1.1 files with embedded SVG fonts, and then you can use the open source font editor called FontForge (*https://fontforge.org/*) to convert such an SVG file to a standard font format such as OpenType. You probably won't want to use this route for serious marketable fonts with advanced features, but for something simple, such as a letters-only font imitating your handwriting, it is perfectly workable.

I will not describe this functionality in detail because the whole story of font creation could easily fill a book. Instead, below is a high-level overview of the process that should get you started.

You can begin with any empty document, but there's a convenient Typography Canvas template (File ▸ New from Template) that provides a 1024 by 1024 px square with horizontal guides at the levels that define a Latin-alphabet letter: the baseline, the ascenders and descenders of the lowercase characters, and the height of the capitals. Create each of your letter shapes in a layer of its own; each letter must be a single path.

Then, open the **Text ▸ SVG Font Editor** dialog. Create a new font by clicking the **New** button. Then, go to the **Glyphs** tab and start adding glyphs by clicking **Add Glyph**; the Glyph name can be descriptive (for example, "lowercase a"), but the Matching string must contain the exact character for which this glyph will be used (that is, just "a" for a lowercase *a*). In the **Advance** column, enter the distance this character takes in the string (that is, its width plus some default spacing).

Finally, you need to copy the letter shape path you have made to the font's glyph (Figure 15-24). Make sure you have the path selected on the canvas and the corresponding glyph selected in the list and click the **Get curves from selection** button. Later, if you edit the shape of a glyph in its layer, you will need to do this copying again to update the font. To preview how your glyphs look together in a string, type something into **Preview Text** and see the result in the preview

pane. You can also specify automatic kerns for specific pairs of glyphs by going to the Kerning tab of the dialog.

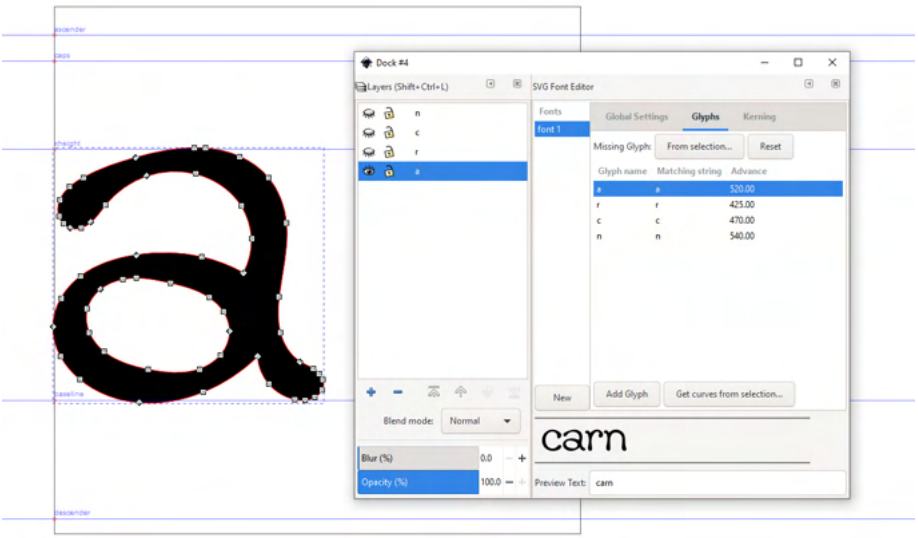


Figure 15-24: Creating a font: draw characters in layers, then create a new font in the document and copy each path to a glyph.

When your glyphs are ready, save the SVG file and open it in FontForge to check. If it looks good, save it in OpenType format with **File ▸ Generate fonts** in FontForge. You then can install the resulting *.otf* file into your operating system for use by any program. If you need to change some parameters of the font, you can probably do that in FontForge; however, you would need to go back to the original SVG file, edit the paths in their layers, and make sure to copy each changed glyph path into the font (Get curves from selection).

Figure 1: An example of a complex vector drawing containing 280 objects (mostly paths; page 4)

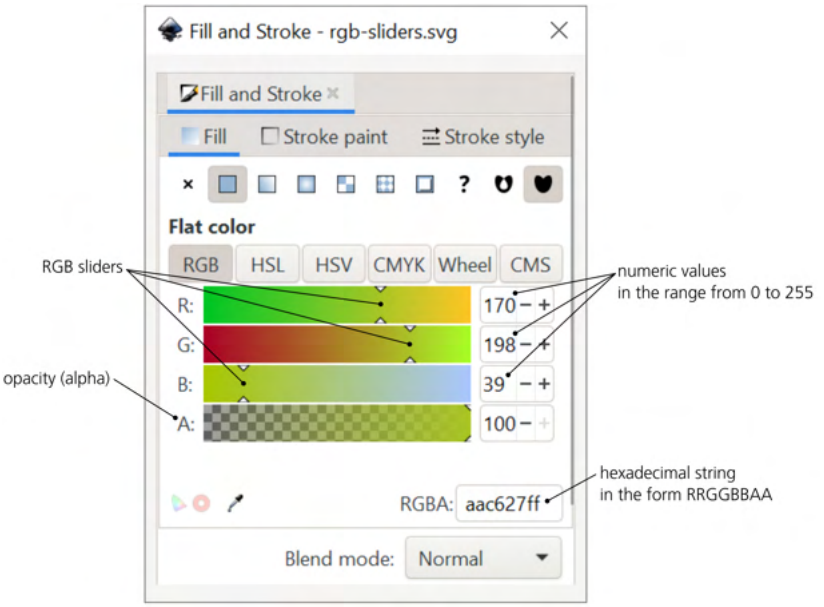


Figure 2: Editing a paint color via RGB sliders in Fill and Stroke (page 146)

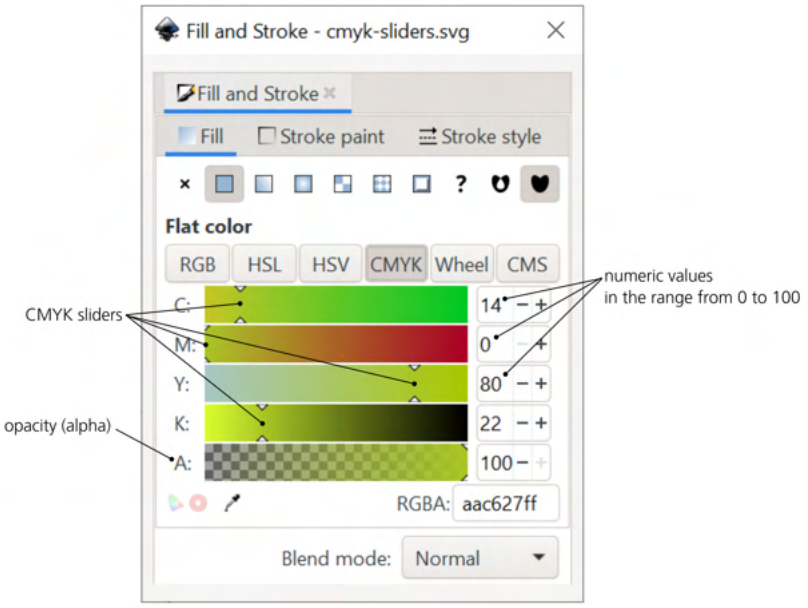


Figure 3: Editing a color via CMYK sliders in Fill and Stroke (page 146)

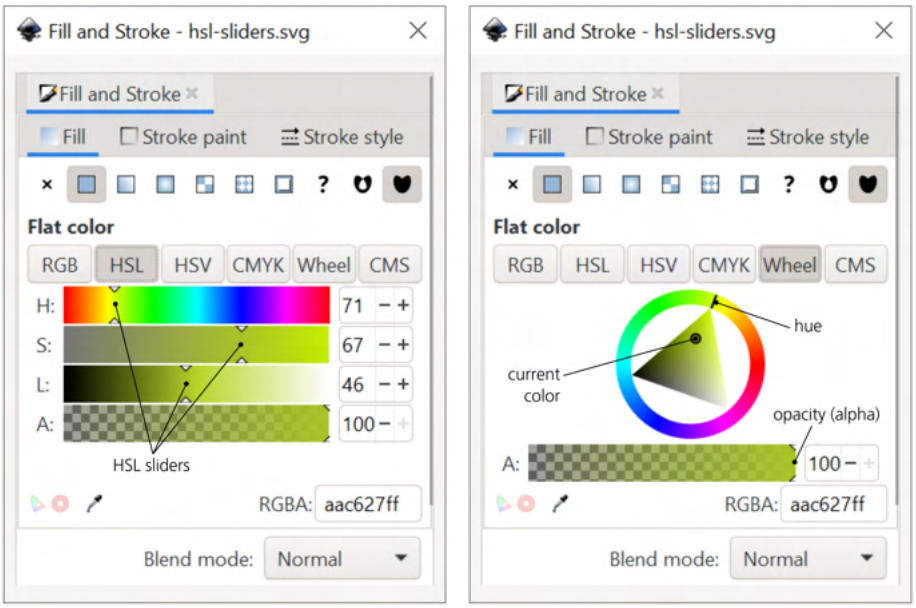


Figure 4: Editing a color via HSL sliders and color wheel in Fill and Stroke (page 147)

Figure 5: Inkscape's default color palette at the bottom of the window; here it is wrapped to show all colors without scrolling. You can choose from about 20 palettes that come with Inkscape, or create your own (page 147).

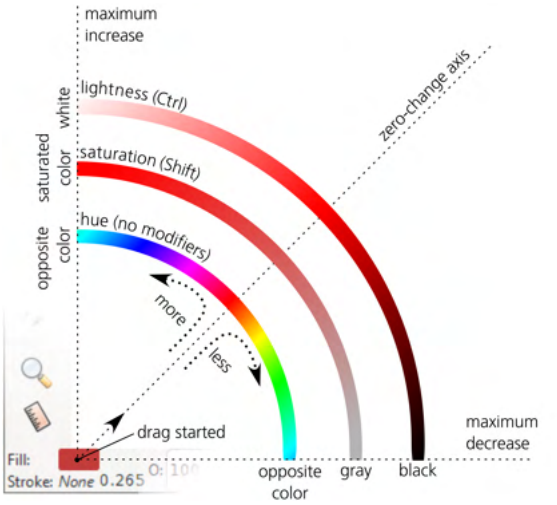


Figure 6: Using color gestures to adjust fill or stroke color in HSL space (page 151)

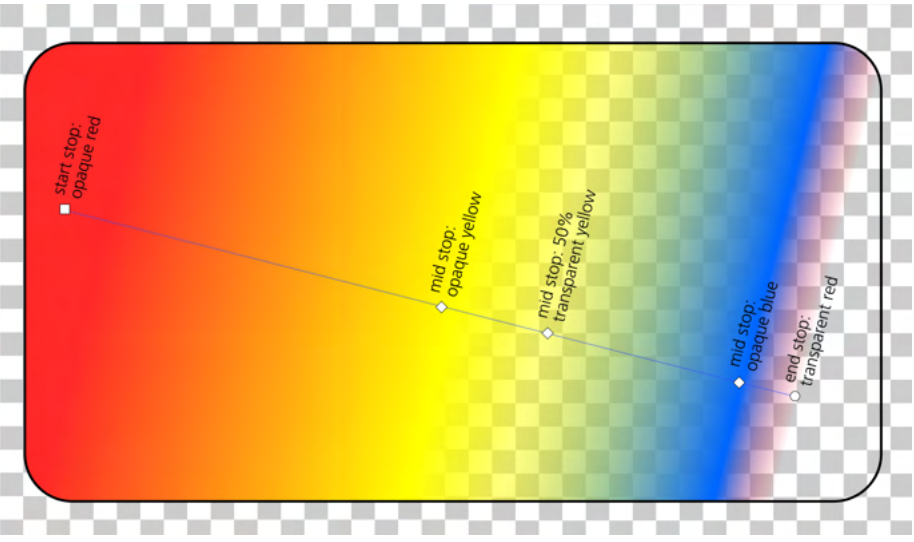


Figure 7: Middle stops in a linear gradient (page 184)

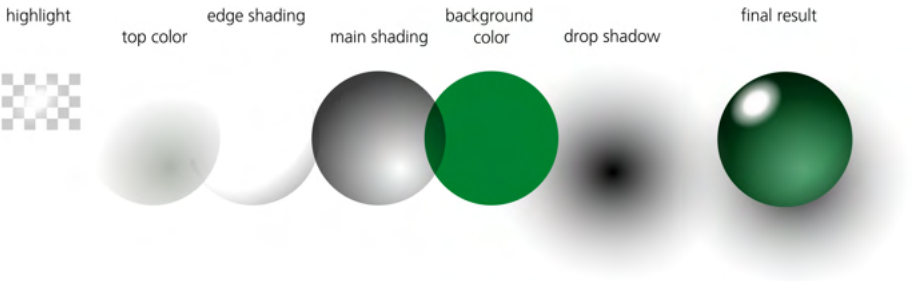


Figure 8: Creating a translucent glass effect with gradients (page 187)

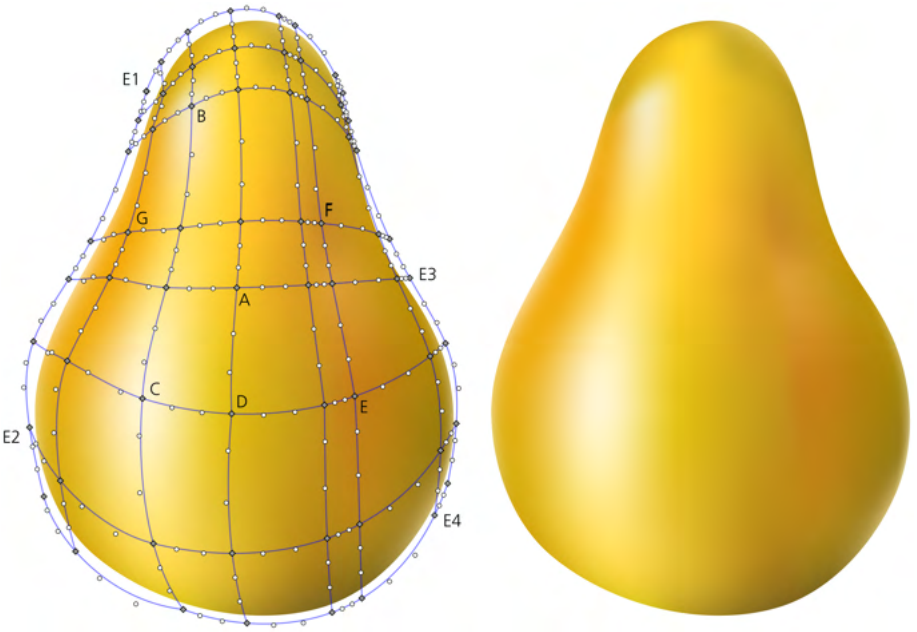


Figure 9: A yummy pear (see text for comments on the marked nodes) (page 194)

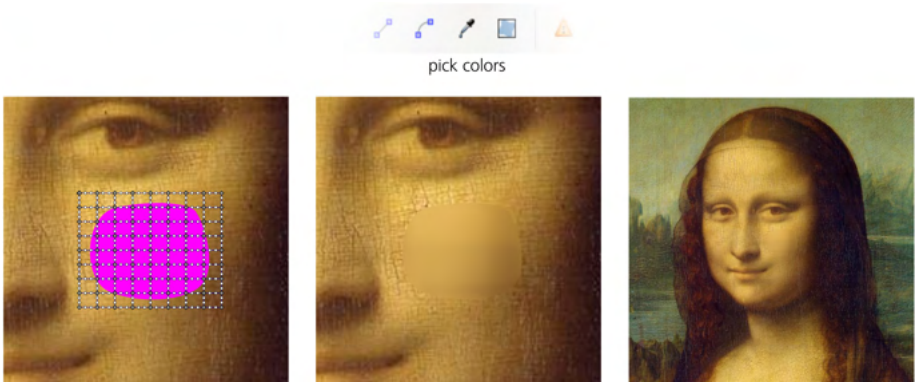


Figure 10: Patching Mona Lisa with a gradient mesh (page 194)

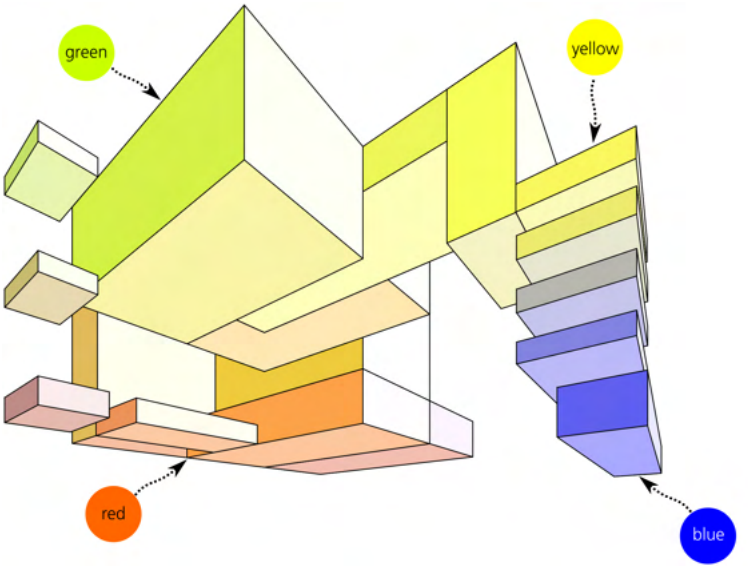


Figure 11: A composition of 3D boxes painted with the Color Paint mode of the Tweak tool using a wide brush (page 212)

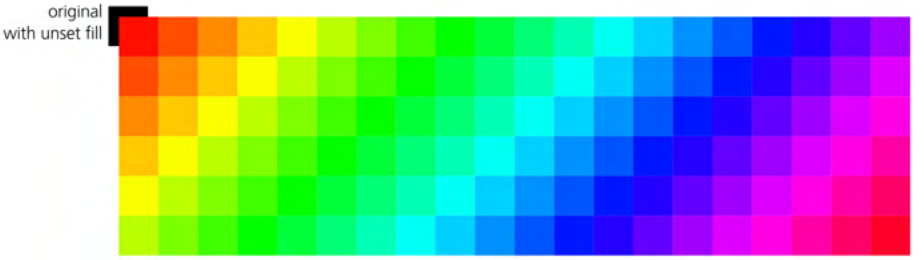


Figure 12: Rainbow pattern (page 349)

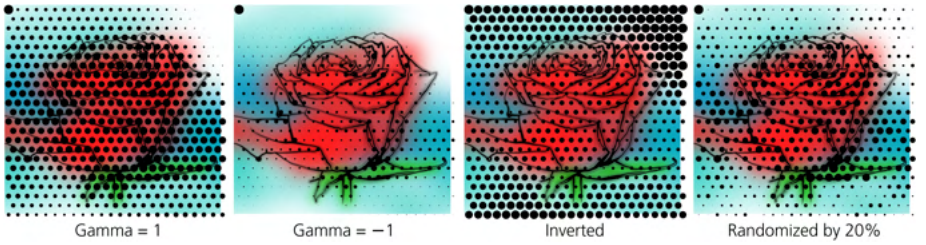


Figure 13: Processing the picked value, tracing lightness to size (page 350)

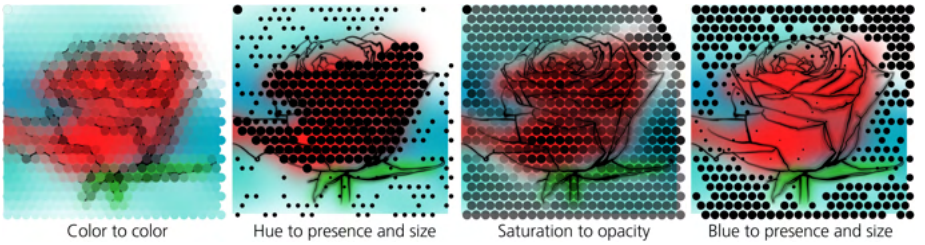


Figure 14: Applying the picked and processed value when tracing (page 350)

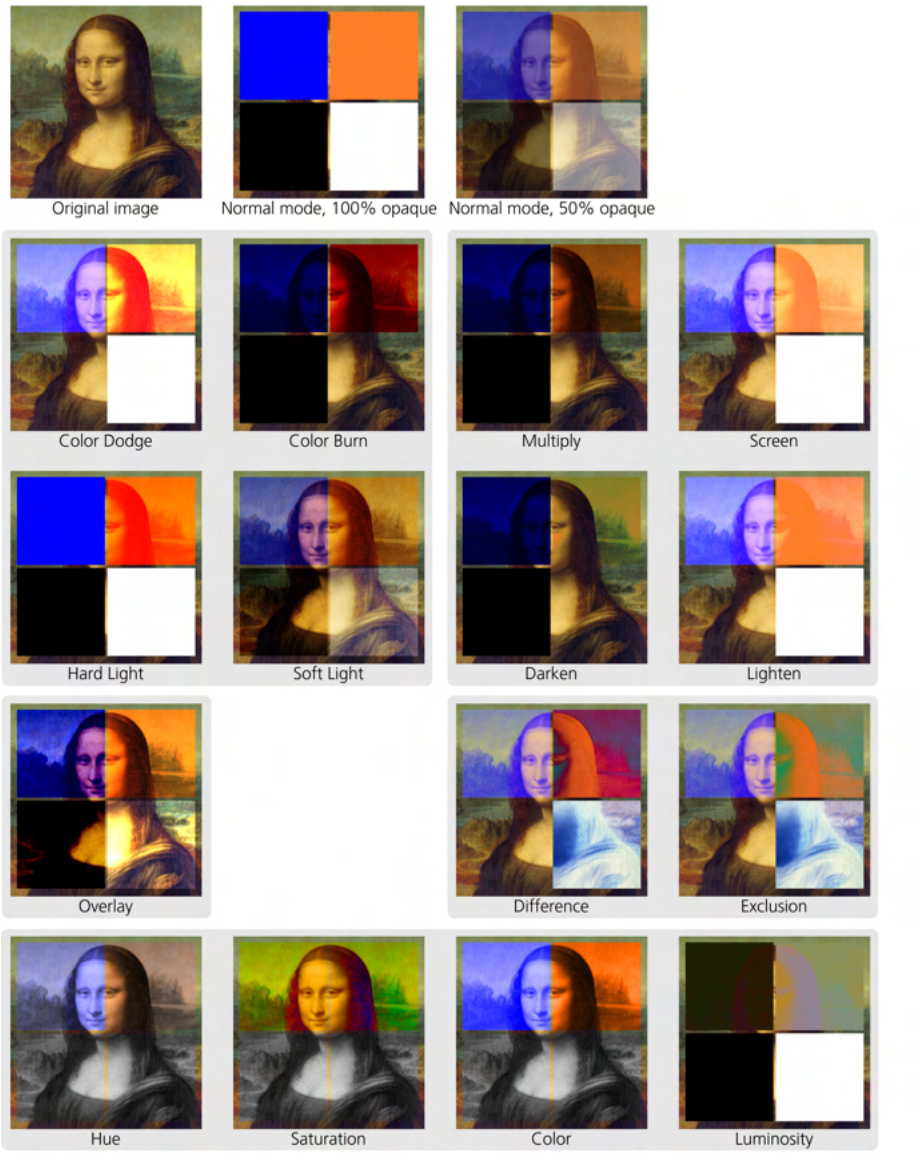


Figure 15: Blending modes applied to colored rectangles over a bitmap. All foreground rectangles are at 100 percent opacity except where indicated (page 357).

Figure 16: Multiple scans tracing: 12 colors (page 389)

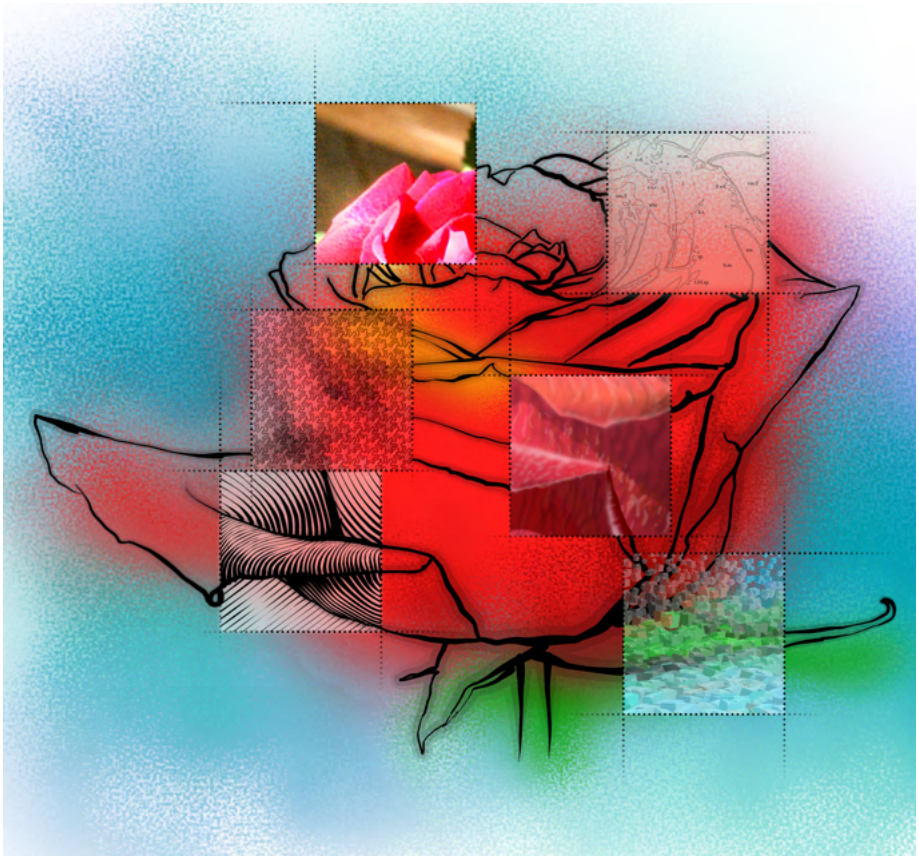
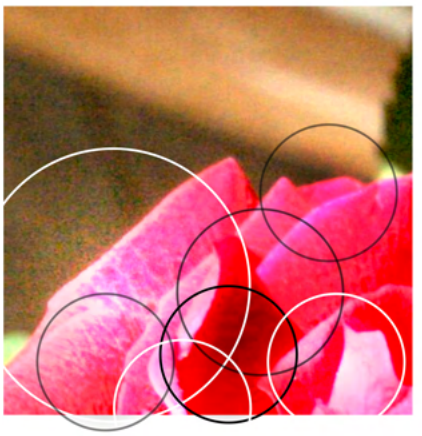


Figure 17: The rose (page 451)

original photo fragment

lightness and contrast adjusted



lights and shadows emphasized with overlays

Figure 18: Spruced-up photo treatment (page 462)

16

CLONES AND SYMBOLS

The idea of a *clone*—a linked copy of an object that updates itself when the original changes—comes naturally from the vector way of thinking about graphics. In a sense, a clone is not a real object—rather, it is just a command: "Display a copy of the object here." The document stores that command, not a real object. An actual object instead of the command is created only in the memory of an SVG application such as Inkscape when it displays the document.

There are both artistic and technical reasons to use clones. Watching several objects change, live, when you edit only one is an exciting experience that suggests many creative possibilities. On the other hand, using clones instead of duplicates can make an SVG document smaller and faster to display. Map symbols, repeated design elements such as bullets or icons, various symmetric designs or patterns—all these are natural to do with clones. Few other vector editors allow you to create live linked copies of objects with such directness and ease as Inkscape.

16.1 Creating a Clone

To clone one or several objects, just select them and press Alt-D (or **Edit ▸ Clone ▸ Create Clone**). The visible result of this is exactly the same as that of duplicating (Ctrl-D): a copy of (each) selected object is created and placed on top of the original. If you need to get a clone of several objects as a whole, group them together and clone the group.

A clone is a *linked* copy of an object, as Figure 16-1 demonstrates. What constitutes this link?

Most important, a clone copies the original's *content*. If it is a clone of a path (Chapter 12) or a shape (Chapter 11), it exactly reproduces the form of the original, and it updates automatically when you edit the original in the Node tool or in a shape tool. If it is a clone of a text object, it has the same textual content and is also updated live when you edit the original with the Text tool. Finally, if you clone a group, you can then enter that group (5.10) to add, delete, or edit objects within the group—and the group's clone will update immediately. On the other hand, since a clone has no content of its own, you cannot edit its content—no node editing, text editing, or ungrouping is possible on a clone so long as it remains a clone.

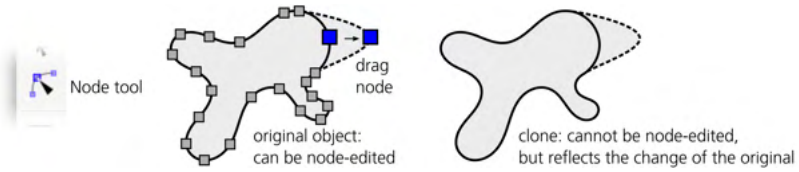


Figure 16-1: A clone is a linked copy of an object.

What about *transforms*? If you scale, rotate, or skew the original, all of its clones will do the same (Figure 16-2). However, if you just move the original object, by default the clone will *not* move (although that behavior can be changed, see below). Of course if you select *both* the original and its clone, you can transform them together—by moving, scaling, rotating, or skewing.

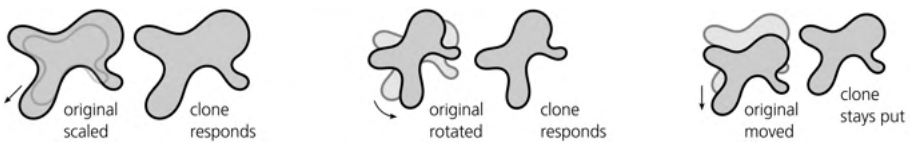


Figure 16-2: Clones respond to the original's transforms, except for moving.

You can also move, scale, rotate, or skew the clone completely independently of its original. The clone's own transform is applied *on top* of the transform inherited from the original. For example, if you squeeze a clone vertically and then rotate its original, the clone will be both rotated and squeezed—but the vertical squeeze will be applied to the shape *after* it is rotated, resulting in a skew, as shown in Figure 16-3.

Figure 16-3: A clone's own transform is applied on top of the transform inherited from the original.

The *style* of the original is also passed on to its clones. If you paint the original a different fill or stroke color, all of its clones will take the same color at once. Conversely, if you try to paint a clone, it will simply refuse to change color, remaining true to its original. (Again, this rule has exceptions, as you will see below.)

This table summarizes various things you can and cannot do on a clone and its original and how those changes affect one another:

	Move	**Scale, rotate, skew**	**Node or shape edit**	**Style**
Applied to **original**	does *not* affect clones (by default)	does affect clones	does affect clones	does affect clones
Applied to **clone**	is possible	is possible (on top of original's transformation)	is impossible	is impossible (unless unset in the original)

CLONES IN SVG

In SVG, a clone is represented by an `svg:use` element. Its `xlink:href` attribute contains a URL pointing to the original of this clone. Per the SVG standard, this URL can point to any element, be it within the same document or in a different document anywhere on the internet. However, Inkscape does not support cross-document references, so any clone created in Inkscape must have its original in the same document.

16.2 Transforming Clones

As you have seen, a clone can be transformed in any way, and this transform is applied on top of the transform it inherits from its original. You can use any transformation method—dragging with the mouse, transforming via keys (6.5), aligning, distributing, and snapping all work on clones exactly as they do on regular objects.

On the other hand, transforms of the original are classified—by how they affect the clones of that original—into two groups: simple moves and everything else. For "everything else" (scales, rotates, and skews) the transform is passed on to all clones. Simple moves, however, are treated differently.

By default, Inkscape tries to isolate clones so that they remain in place when the original is moved. Under the hood, this is done by moving the clone in the

opposite direction to compensate for the original's move. This works pretty intuitively and is usually convenient; for example, you can take the original of a large pattern of clones and move it away without disturbing the pattern. You can also grab both the original and its clones and move them anywhere, and they would behave as expected—that is, they move in parallel, even if the clone has a transform of its own.

Users can modify this compensation behavior. The Behavior ▸ Clones page of the Preferences dialog (Figure 16-4) contains the following options:

- Move in parallel forces all clones, including those rotated or scaled, to always move in parallel with the original, as if they were always selected together with it (even if in fact they are not).
- Stay unmoved (this is the default) forces those clones that were not selected to stay unmoved (but those that *are* selected are moved as usual).
- Move according to transform turns off any clone movement compensation. Now each clone, selected or not, moves according to the transform inherited from its original, without any attempt to compensate. When complex transformations are in effect, the resulting behavior may appear chaotic—but from SVG's viewpoint, it is the least intrusive option, since the inherited clone transformations are not tampered with in any way.

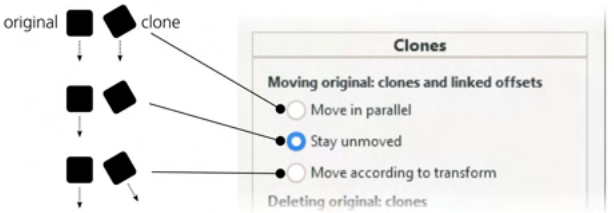


Figure 16-4: Setting up movement compensation of clones

Also, transforming clones does not honor the Affect buttons on the Selector controls bar (6.11): it always behaves as if all of those buttons were pressed. You cannot scale a clone in such a way as to leave its stroke width unchanged or its gradient unaffected—because what you're transforming is, in effect, an image of the original, and that image cannot have a different stroke width or gradient position compared to the original.

16.3 Styling Clones

I have already mentioned that clones inherit their style from their originals: a clone of a red rectangle will be red. There are, however, several exceptions and workarounds for this limitation.

First of all, opacity (8.3) and blur (17.1) are not subject to this limitation: you can easily blur a clone or make it semitransparent. This is because these properties *accumulate*—that is, if you blur something and then blur its parent, those blurs add up, and the result will be more fuzzy than from either of the

blurs taken alone. This also means that if your original is *already* blurred or has less than 100% opacity, you can make its clone *more* blurred or *more* transparent, but not less.

The same applies to filter effects (Chapter 17; actually, blur is just one of the filters). You can apply any filter to a clone, and it will work on top of any filters that the original of this clone has. For example, you can make the clone of a red rectangle green or gray by applying a color-shifting filter.

But what about plain fill or stroke colors? You can change them in a clone—but only if the original cooperates. Namely, any style property you want to change in a clone must be *unset* in the original. Unsetting (8.2) is not the same as setting to `none`: a property is unset when it is simply *not specified* for an object, which allows its clone's property to take effect instead, as shown in Figure 16-5.

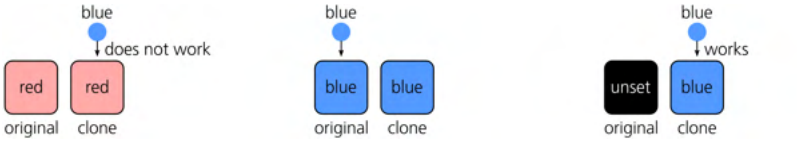


Figure 16-5: A clone can be painted if its original has the paint unset.

Inkscape has a special button in the Fill and Stroke dialog, as well as a command in the selected style indicator (8.6), for unsetting the fill or stroke properties of an object. An object with an unset stroke simply has no visible stroke, but if you unset its fill, it is shown as black. If you want to unset some other style property, use the **Selectors and CSS** dialog (8.1) to remove the property you want to override in the clones.

If you have a group as the original object, you can unset fill or stroke in some of the members of that group only, leaving others colored. Then, if you clone that group and paint the clone, only the objects with unset properties will take on that color, whereas everything else will remain true to the original.

16.4 Chaining Clones

Duplicating (4.5) or copying and pasting a clone gives you another clone *of the same original*. (You can copy and then paste a clone into a different document but only together with its original; if you try to paste a clone alone, it will end up *orphaned* and invisible.) Duplicating an existing clone is often convenient, as this duplicate will also get the first clone's own transform and style, if it had any. Of course, you can also get another clone simply by cloning the original again.

Nothing prevents you from cloning a clone object itself. The resulting object—a *clone of a clone*—will still display the content of its ultimate original, but its link to it is no longer direct: it is now a grandchild, not a child, of the original. Such a grandchild clone inherits transforms and style first from the original, then from its parent clone, and finally, applies its own transform and style on top of all that. Such clone-of-clone chains (they can be of any length) are rarely useful; in almost all cases, multiple clones of the same original can be used instead, as Figure 16-6 demonstrates.

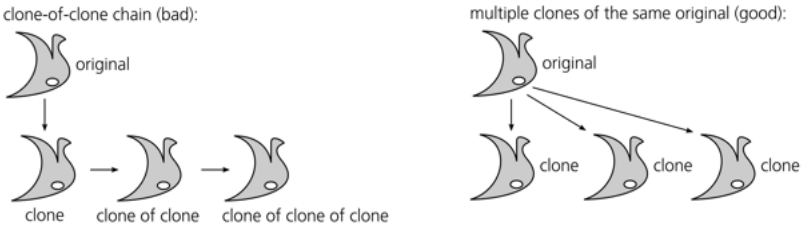


Figure 16-6: Instead of clones of clones, use multiple clones of the same object.

To figure out whether the object you selected is a clone—and if so, what it is a clone of—look at the status bar. It will describe your selected object as, for example, *Clone of: Group* or *Clone of: Clone of: Path*. If you want to know which object is the parent of the selected clone, press Shift-D: Inkscape will draw a dashed line from the clone to its original (that line will disappear after one second) and select the original object, as shown in Figure 16-7.

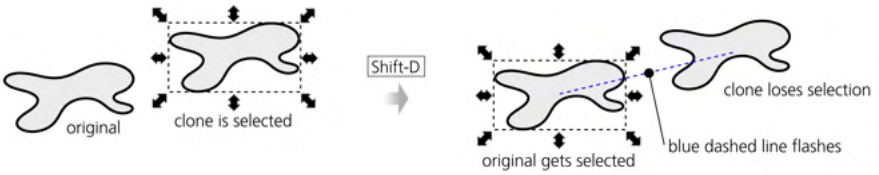


Figure 16-7: Press Shift-D to find the original of a clone.

16.5 Unlinking and Relinking Clones

A clone's live link to its original is its main feature—that is, after all, why we use clones. This link facilitates a lot of tasks in complex designs; for example, you can use clones for identical buttons or bullets on a website mockup, and then change all such elements at once by changing their common original. However, sometimes this link becomes an obstacle. If you want to edit a clone independently of its original, you need to *unlink* the clone.

Select a clone (or clone of a clone) and press Shift-Alt-D (or select **Edit ▸ Clone ▸ Unlink Clone**). Visibly, nothing changes, but the object is no longer a clone—it is now a copy of the original, with all its additional transforms and style preserved, but now independent and fully editable. You can unlink multiple clones at once.

What happens if you delete the original of a clone? By default, its clones are automatically unlinked—that is, turned into copies of the object being killed. On the same Behavior ▸ Clones page of the Preferences dialog you can, however, force clones to disappear when their original does (Figure 16-8).

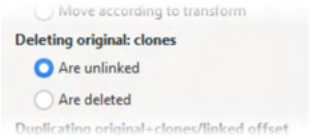


Figure 16-8: Setting up the behavior of clones when the original is deleted

When you *relink* a clone, it remains a clone, but its original changes. Here's how to do it. Suppose you have a clone C of an object A, but you want it to be a clone of B instead. Select B and copy it to the clipboard (Ctrl-C), then select clone C and do **Edit ▸ Clone ▸ Relink to Copied**. After that, C becomes a clone of B—which, depending on what B is, may result in C changing shape, style, position, transform, or any combination of these, as shown in Figure 16-9.

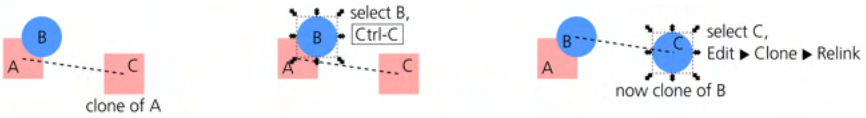


Figure 16-9: Relinking a clone to a different original

Manual relinking is a rarely needed operation. More useful is *automatic relinking on duplication*. Imagine you have a complex group of objects, some of which are clones of others—for example, a 3D-like graphic button where the shadow is a blurred clone of the main shape. Now, imagine you select this entire group and duplicate it. What happens?

Within the group, each regular object will duplicate to a regular object, and each clone will duplicate to a clone. However, the clones will still be linked to the originals in the source group—which is most likely not what you wanted! It would be more natural for the shadow in the duplicated group to be a clone of the shape *in the same group*, not some other group far away. To ensure this, on the same **Behavior ▸ Clones** page, check **Duplicating original+clones/linked offset: Relink duplicated clones**. Now after duplication, you will have two independent buttons, each with its own editable shape and a shadow linked to that shape (Figure 16-10).

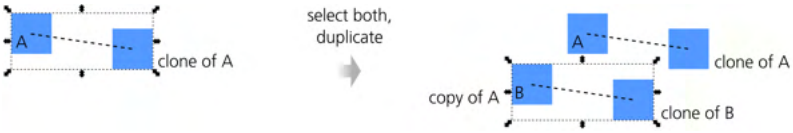


Figure 16-10: Automatically relinking clones on duplication

16.6 Tiling Clones

The powerful Create Tiled Clones dialog (Edit ▸ Clone ▸ Create Tiled Clones) is a tool for creating multiple clones at once, arranged into all kinds of spatial and color patterns ranging from absolutely regular to totally randomized. As a bonus, you can also make the pattern of the clones trace the image underneath.

16.6.1 Size and Bounding Box

The first step is to select the object you will be cloning. I recommend using a group for that; even if you have a single object, group it (Ctrl-G). This way, you will later be able to add more objects to the original group and the clones will reflect that. Place the original into the top-left corner of the area you want to fill with the pattern.

In the **Create Tiled Clones** dialog, start by specifying the size of the tiling (Figure 16-11). You can give the number of rows and columns in the pattern; or, if you have some specific area to fill, you can type or paste its width and height. Note that too-large patterns—with more than a few thousand clones—can slow down Inkscape considerably.

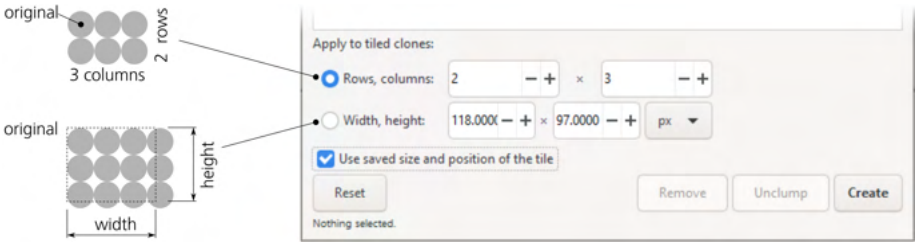


Figure 16-11: Setting the size of a pattern

Once you set all other parameters as described below (or just trust the defaults), click the **Create** button. The pattern appears on the canvas, but you still have the original object selected. The Remove button deletes any previously tiled clones of the selected object. Note that Create first does Remove—that is, when you click Create, any existing tiled clones (but not regular clones created by Alt-D) are removed and replaced by a new pattern.

The Unclump button works exactly as the same-name button in the Align and Distribute dialog (7.5.2), except it moves all the tiled clones of the selected object rather than all selected objects, as shown in Figure 16-12. Unclumping is especially useful for making randomized patterns more uniform without regularizing them. The Reset button changes all parameters of the dialog back to the defaults.

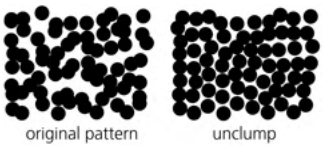


Figure 16-12: Unclumping a randomized pattern

The Use saved size and position of the tile checkbox has no effect when you create a pattern for the first time. However, if you modify the object you're tiling and create the pattern again, normally Inkscape will use the altered size of the object, which may result in the pattern changing its overall size and the alignment of the tiles. To make Inkscape use the same tile size as the last time you created a pattern from that object, even if the object's size has changed, check this checkbox. For example, you can create a pattern from a rectangle, fine-tune all the parameters, then scale up the original rectangle and re-create the pattern with this checkbox checked. That will give you the exact same pattern as last time, but with larger tiles overlapping each other.

All tiled patterns include a clone that exactly (except when randomized) overlays the original object. This means if you lose the selection of the original,

simply clicking the original's location will select the overlying clone and not the original. Use Alt-click (5.9) or select any of the clones and press Shift-D to jump to the original.

16.6.2 Symmetry

The first tab in the dialog is titled Symmetry. It contains a list of *symmetry groups* from which you select one to use for your pattern. Each symmetry group is a specific way to transform the clones to form the pattern. The number of these groups is exactly 17 because it has been proven that any regular pattern on a plane can be classified into one of these 17 types; see the Wikipedia article on "wallpaper group" for extensive details and examples. Here is a brief overview of the symmetry types:

P1
: This is the simplest possible symmetry: the pattern tile is simply repeated in a rectangular grid without any rotations or flips (Figure 16-13).

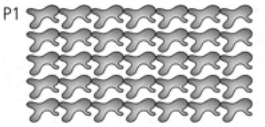


Figure 16-13: The basic symmetry: P1

P2, PM, PG, CM, PMM, PMG, PGG, and CMM
: These symmetries use rotations by 180 degrees as well as vertical and horizontal flips in various combinations (Figure 16-14). However, all of these symmetries use the same rectangular grid placement as P1, with the width and height of each unit of the grid being the same as those of the original object.

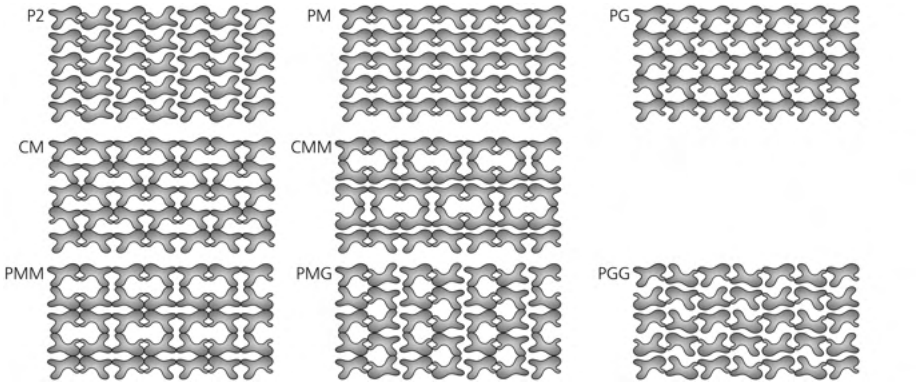


Figure 16-14: Symmetries with flips and 180-degree rotations

P4, P4M, and P4G

These symmetries involve rotations by 90 degrees, so they produce square-based patterns. The P4M symmetry results in partial overlapping of rectangular tiles; with it, you can use triangular tiles to avoid overlapping and fill the plane, as shown in Figure 16-15.

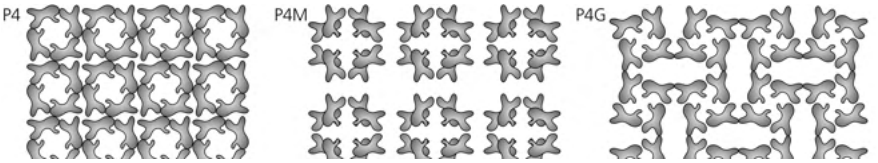


Figure 16-15: Symmetries with 90-degree rotations

P3, P31M, and P3M1

These symmetries involve rotations by 120 degrees and are thus roughly triangular in appearance (Figure 16-16). Again, P31M creates a more dense pattern with partially overlapping tiles, so you can use a "pie slice" shape to fill the plane with this symmetry without overlapping.

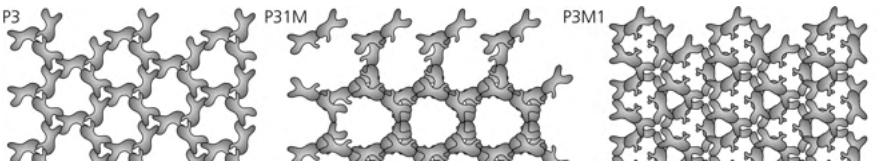


Figure 16-16: Symmetries with 120-degree rotations

P6 and P6M

These symmetries rotate the tiles by 60 degrees, forming snowflake-like hexagonal patterns (Figure 16-17). Of them, P6M again overlaps the tiles and requires a "pie slice" shape to fill the plane without overlapping.

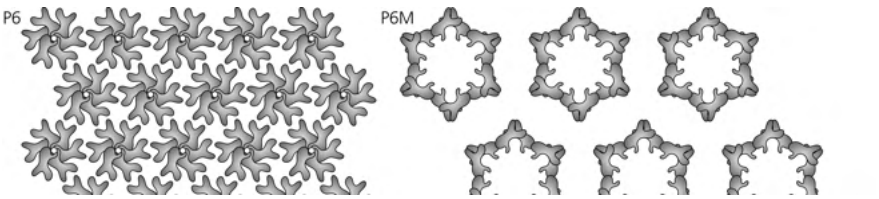


Figure 16-17: Symmetries with 60-degree rotations

For "dense" patterns (P4M, P31M, P6M) that overlap the tiles, try creating the pattern and then scaling the original down. This will make the pattern sparser, and its characterisic logic will be easier to understand.

When working on a *tessellation* (a pattern that completely fills the plane without gaps or overlapping), it would be very difficult to create the required shape of the tile in isolation, even if you understand how your chosen symmetry works. Instead, just start with any random shape, create the pattern from it using

the desired symmetry, and then node-edit the original path watching how the pattern's clones repeat its changes. In this way, you can easily produce a surprisingly sophisticated tessellation (see 25.2 for an example).

16.6.3 Shift, Scale, and Rotation

The next three tabs in the dialog allow you to specify the *additional* transforms to be applied to the pattern's tiles—that is, transforms on top of the shifts, rotations, and flips created by the chosen symmetry group (Figure 16-18).

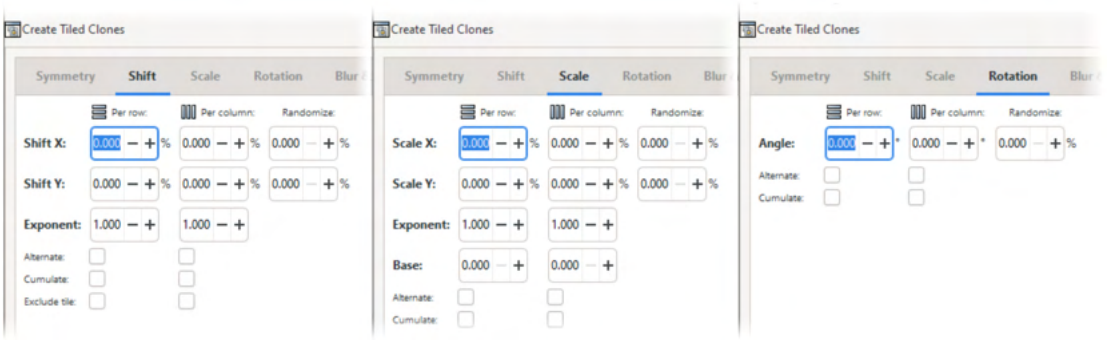


Figure 16-18: The Shift, Scale, and Rotation tabs

You can specify all of these additional transform components separately per row and per column, and each value can include a degree of randomness. You can specify, for example, an equivalent of "Make tiles in each next row 20 percent taller, rotate tiles in each next column by 5 degrees, and make the rotation angle vary randomly by 10 percent." All shifts, scales, and randomization values are measured as percentages of the original object's dimensions.

Figure 16-19 shows how it works for shifts.

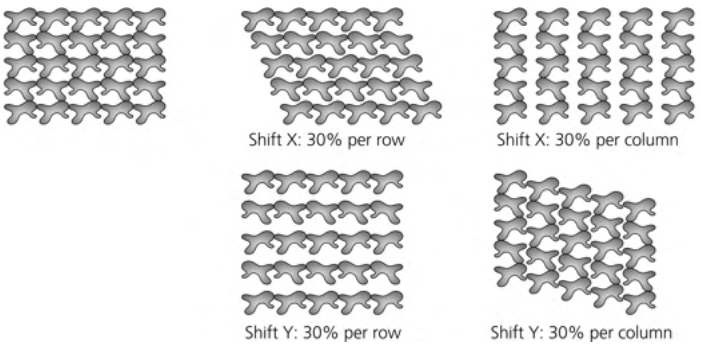


Figure 16-19: Specifying shifts per row and per column (PG symmetry)

Negative shifts are possible, too. Naturally, to get all the clones to overlay the original, you need to specify **Shift X:** –100% per column and **Shift Y:** –100% per row. If you combine this with rotation per row, it's easy to create a flower or a clock face, as shown in Figure 16-20.

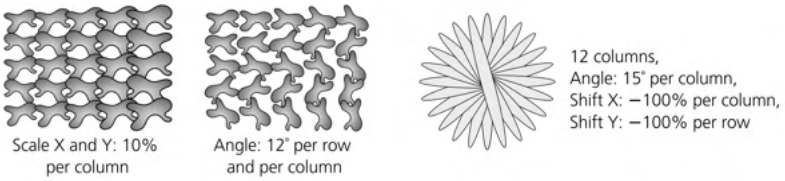


Figure 16-20: Specifying scale and rotation per row and per column

You can make the values Alternate (for example, scale clones in every second row), as shown in Figure 16-21. The Cumulate checkbox forces the values of shift or scale to accumulate. For example, normally a shift of 10 px per column means that each column is shifted by 10 px relative to the previous column; if you check Cumulate, the same value would make every column shift 10 px further than its predecessor—that is, the second column is 10 px from the first, the third is 20 px from the second, and so on.

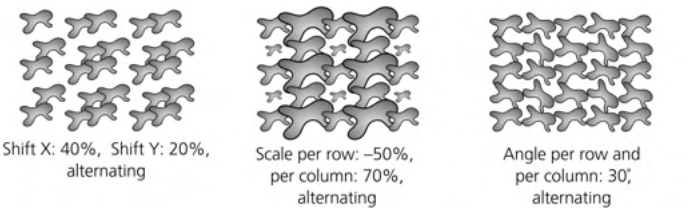


Figure 16-21: Alternating transforms

What is considered a "row" and a "column" when calculating transform values with complex symmetries? Inkscape draws complex patterns by symmetric clusters (of 3, 4, 6, or 12 clones, depending on the symmetry), going from one cluster to the next horizontally within a row. In other words, clones that belong to one cluster are considered to be in the same row but in different columns. This means that the Per row values shift rows of clusters or scale clones in each cluster uniformly, whereas Per column values affect each clone independently—and, as a result, clusters lose their symmetry, as Figure 16-22 demonstrates.

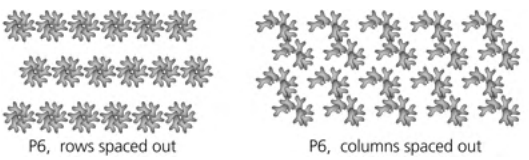


Figure 16-22: Transforming complex symmetries

If you want to space out symmetric clusters in both dimensions, just create a single cluster with your chosen symmetry, group it, and tile the group with a simple P1 symmetry, possibly with alternating shifts.

16.6.4 Blur and Opacity

This tab of the dialog looks and acts very similar to the transform tabs; here, you can adjust the blur and opacity of the clones in the pattern, per row or per column, with optional alternating or randomization (Figure 16-23). Remember

that you can only make a clone more blurred or more transparent than its original, but not less.

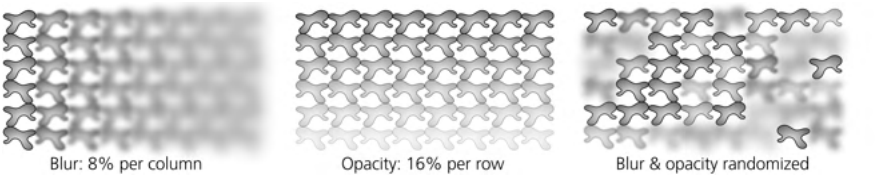


Figure 16-23: Varying blur and opacity in a pattern

16.6.5 Color

As you've seen in 16.3, if you want to paint a clone with its own color, you need to unset the corresponding property in the original. Once you've done that for the paint you're going to change (fill, stroke, or both), the Create Tiled Clones dialog allows you to create a variety of color patterns.

The Color tab of the dialog looks similar to the tabs you've already seen (Figure 16-24). Here, you can vary any of the three components of the clone color in the HSL model—hue, saturation, and lightness (8.4.3)—per row or per column, as well as alternate or randomize the changes. You also need to specify the Initial color from which all those variations will start; just click the color swatch and use the color selector dialog. Remember that the original must have unset fill or stroke; otherwise, this tab will have no effect!

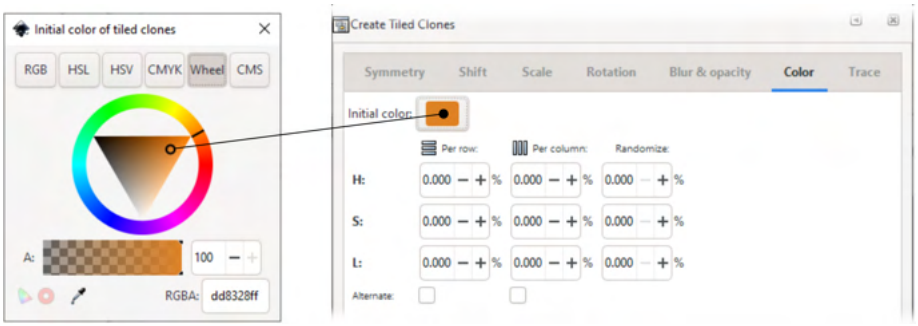


Figure 16-24: Setting an initial color for the Color tab

For example, by starting with red and varying the hue by 5 percent per row and per column, you get a slanted rainbow (see Figure 12 in the color insert).

16.6.6 Tracing

The last tab of the Create Tiled Clones dialog (Figure 16-25) is different. Here, you can make the pattern *trace* any image on top of which it is built—that is, make some aspects of each clone depend on what is immediately below it. The background image you're tracing can be an imported bitmap (such as a photo) or any vector drawing; it makes no difference to the tracer what to pick its values from.

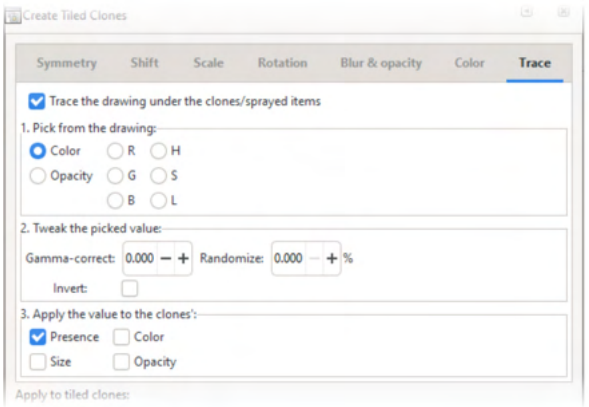


Figure 16-25: The Trace tab

The tab contains three main areas that correspond to the three main steps of the tracing algorithm. First, you pick some value at the location of each clone; second, you do some optional processing with that value; and third, you apply the result to some aspect of the clone. Enable the controls of the tab by checking the **Trace the drawing under the tiles** checkbox.

For the input value, you have the option of picking the Color, Opacity, or any single component of the color in RGB or HSL models. The value picked for each clone is averaged over the entire rectangular area that will be covered by that clone's bounding box. All these options produce a single numeric value in the range from 0 to 1, except Color, which picks all three components of the color as a composite value.

The possible ways to process the value include:

Gamma correction
: Positive gamma shifts the picked value up; negative shifts it down.

Randomization by a given percentage
: The value will have a random component of the given size: 0% randomization means the value is exactly as picked, 100% randomization means it is totally random and does not depend on the picked value at all.

Inversion
: Turns high values into low and vice versa.

If a color is picked, this processing is applied to each of its components independently (see Figure 13 in the color insert).

Finally, the resulting value can be applied to the clone's probability of presence (0 means the clone is absent at this location, 1 means it's present, and intermediate values make it appear with the given probability); color (this can directly reuse the background color if it was picked, otherwise it translates single values to shades of gray); size (from disappearance at 0 to its full size at 1); or opacity (smaller values make the clone more transparent). You can enable any number of these options at the same time; for example, you can pick lightness, invert it, and apply it to both the opacity and size of the clones (see Figure 14 in the color insert).

16.7 The Symbols Dialog

You can think of *symbols* as clones organized into libraries for reuse. Inkscape's Object ▸ Symbols dialog (Shift-Ctrl-Y) gives you access to a number of common graphic symbol sets that come with Inkscape. You can also use it to create and manage your own collections of custom symbols.

Inkscape's prepackaged symbol sets should give you an idea of what kinds of objects it may make sense to organize into a library:

AIGA (American Institute of Graphic Arts)
: Urban signage, used mostly in airports: arrivals, customs, barber shops—very recognizable and very 1970s.

US National Park Service Map Symbols
: Campgrounds, guided trails, post offices—stylistically similar to AIGA symbols.

Flow chart shapes
: The conventional blocks of flow chart diagrams, such as storage, extract, merge, display, and so on.

Logic diagram symbols
: The AND gate, OR gate, NOT gate, and friends.

Word balloons
: Various shapes of word balloons to be used in comics.

To insert a symbol in your document, just drag in from the dialog onto the canvas. The object you get behaves as if it is a clone of something, but the status bar description says *Symbol*; nevertheless, if you want to edit this object, you can do the same Edit ▸ Clone ▸ Unlink Clone as you would do for a clone. (If all you want to do is move, scale, or rotate the symbol, you don't need to unlink it.)

The search field in the Symbols dialog allows you to find a symbol with a certain word or phrase in its description. The search is performed in the currently selected symbol set or in all of them if you select Symbol set: All symbol sets.

If you switch the dialog to Current document (Figure 16-26), you will see that it already contains all the symbols you have added from the standard libraries (if any). This is where you can also add your own objects as symbols: select an object and click the **Add Symbol** button in the lower-left corner. Once the new symbol is in the list, the original object can be deleted and replaced with a copy of the symbol. Your new symbol will be inserted with the same size and styling as it had when you created it.

To delete a symbol from the document, select it in the list and use the **Remove Symbol** button; after that, an actual copy of the symbol is placed on the canvas, and all instances of this symbol are converted into clones of this visible object.

NOTE *Unfortunately, Inkscape does not ask for a title when you're creating a new symbol—it gets an unhelpful numeric name and therefore is not searchable by title. To fix this, before you turn an object into a symbol, set its* ***Title*** *in the* ***Object Properties*** *dialog (4.1).*

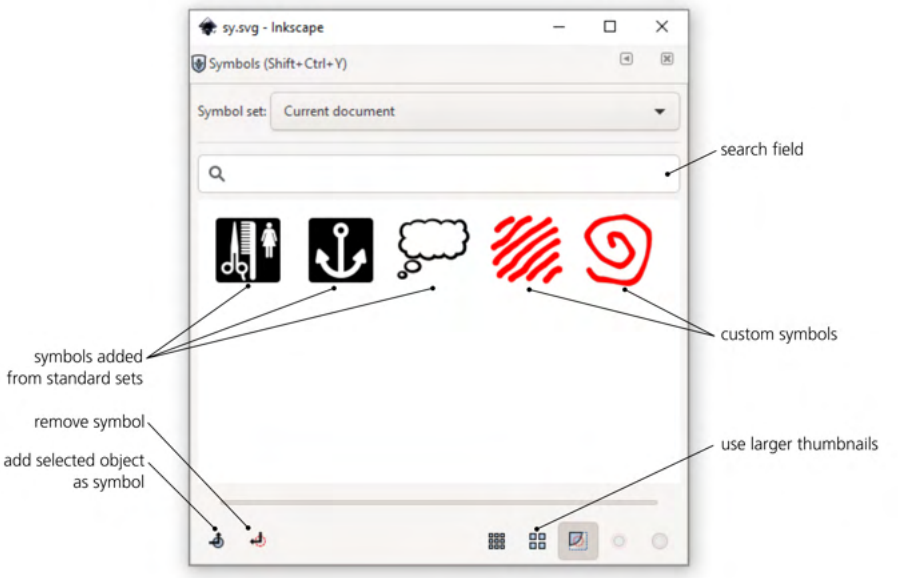


Figure 16-26: The Symbols dialog managing the symbols in the current document

If you want to share your custom symbol library between documents, all you need to do is to place an SVG document containing the symbols into Inkscape's *symbols* folder (go to the **Preferences**, **System** page, and click **Open** next to **User symbols**). After that, your symbol set will be available in the Symbol set list (under the name of your SVG file).

17

FILTERS

Inkscape filters are a way to apply complex bitmap processing algorithms to the objects in your drawing. True to the vector spirit, filters are nondestructive: you can always change the parameters of any filter, and the original vector object remains fully editable. Examples of what filters can do include blurring, sharpening, color adjustments, adding texture, various distortions, 3D-like effects such as bevels, and many others. Moreover, you can combine filters into arbitrarily complex *filter stacks*.

SVG filters (as defined by the SVG standard and implemented by Inkscape) are extremely powerful; a whole book could be written on their uses and capabilities. Unfortunately, they can also be quite technical, especially if you're trying to compose your own filter stacks. In this chapter, I start by looking at some simple ways to use filters, such as blurring (17.1) and blend modes (17.2). Then I describe in more general terms how to manage filters on objects (17.3) and give an overview of the library of preset filters that come with Inkscape (17.4). Finally, for those who want more power and are not afraid to get more technical, I describe the Filter Editor dialog (17.5), where you can create your own filter stacks from standard filter primitives.

17.1 Blur

Properly called *Gaussian blur* (named after Carl Friedrich Gauss, a German mathematician), this effect smoothly dissolves an object, as if you were viewing it through an out-of-focus lens. It is just one of the 14 filter primitives that Inkscape supports, but of all these primitives, Blur is the most easily accessible. For any selected object, you can apply blurring with a slider at the bottom of the Fill and Stroke dialog, as shown in Figure 17-1.

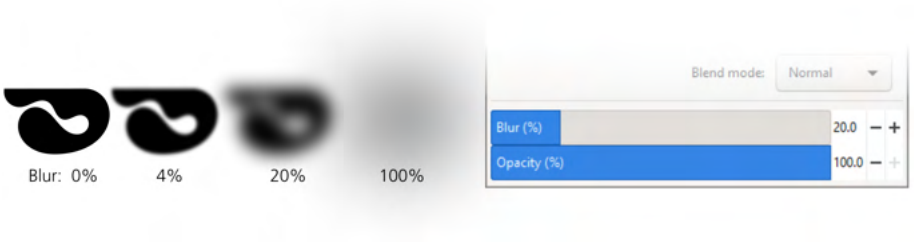


Figure 17-1: The Blur slider in the Fill and Stroke dialog

The same paired opacity and blur sliders are available in the Layers dialog (4.9.4), where they apply to the selected layer as a whole.

Easy blurring expands Inkscape's capabilities enormously. Blur is everywhere in the physical world—non-crisp shadows, glows and halos, or anything viewed out of focus or in motion—so it is essential whenever you're drawing anything realistic. It is also not something you can easily imitate with regular vector shapes or gradients. Inkscape therefore treats blur as a fundamental property of an object, similar to opacity and the blend modes (17.2).

The Blur slider controls the *amount* of blur in a range from 0 to 100 percent; 1 percent barely changes the appearance of an object, and 100 percent turns any object into a shapeless puff. Technically, 100 percent blur makes the blur radius equal to half the size of the object, but the scale is not uniform: going from 0 to 10 percent increases the blur radius much less than from 90 to 100 percent because you need more precise control at the lower end of the scale.

Setting blur as a percentage ensures that it works the same for objects of any size: a large object blurred by 10 percent has a larger *absolute* amount of blur (larger blur radius), but it looks proportional to a small object that is also blurred by 10 percent. If you want to get the same blur radius in objects of different sizes, you would need to use different blur percentages, as Figure 17-2 demonstrates.

Figure 17-2: Blur amount (same in A and B) and blur radius (same in A and C)

Blurring an object expands its bounding box but only if the visual bounding box type is used (which is the default, 4.3). For example, if you want to export a blurred object without cropping the blur, you cannot use the object's geometric bounding box as the export area (18.6).

You can think of each blurred object as residing in its own fully transparent layer, so it never "smears" any adjacent nonblurred objects. Since that per-object layer is transparent, the edges of a blurred object become partially translucent. However, you can blur a group, and in that case, members of the group are blurred together, as if placed on a single layer blurred as a whole.

In Figure 17-3, on the left are two adjacent rectangles with no gap between them and no blur. In the middle, each rectangle is blurred by 20 percent separately; as you can see, the striped background shows through the blurry gap between the objects. On the right, however, the same rectangles are grouped, and then the group is blurred by 20 percent; now there is no gap—blurring only adds transparency at the edges, but it cannot reveal what was fully covered in the center of the opaque group.

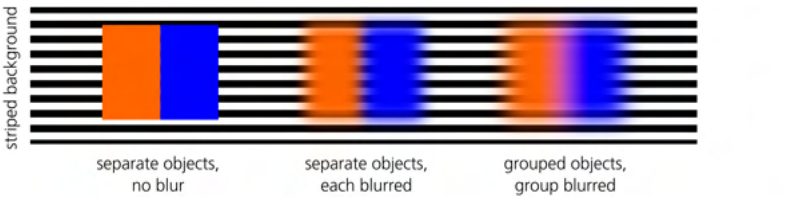


Figure 17-3: Blurring separate objects vs. blurring a group

Grouping also allows you to apply several layers of blurring to the same object. For example, you can blur the object itself, then group it (possibly even with itself, to produce a group of one object) and blur the group. Naturally, by blurring a group, you can make objects look *more* blurred than before, but not *less*. (Similarly, as you saw in 16.3, you can make a clone more blurred than its original object, but not less.)

NOTE *Blurring is not the same as feathering, as Figure 17-4 demonstrates. Blur affects the entire object—in particular, lines on a bitmap object or object boundaries inside a group are all blurred. Feathering, on the other hand, is simply blurred transparency that masks out the outermost edges of an object (such as a photo). Inkscape can do feathering as well, using a predefined composite filter (17.4.2) or a blurred mask (18.3.2).*

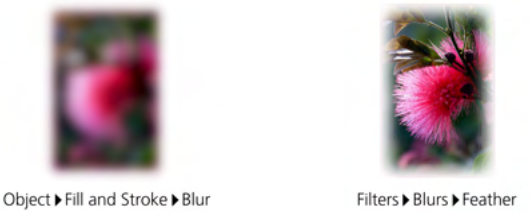


Figure 17-4: Blurring vs. feathering

17.1.1 Blur and Transformations

What happens when you transform a blurred object?

Moving a blurred object moves it as a whole, without affecting the blur in any way. Proportional (uniform) scaling also scales the object as a whole, including the blur radius, so that the blur percentage (in the Fill and Stroke dialog) does not change.

Nonuniform scaling is more interesting. For example, if you squeeze a blurred object vertically, its blur will squeeze with it and become *nonuniform*—now the object is more blurred horizontally than vertically. Such nonuniform blur can approximate real-world *motion blur*—the way the object would look in a photo if captured in fast motion (in this case, horizontal). So, if you want to apply nonuniform blur to a nonsqueezed object, start by stretching it in the opposite direction, blur it, and then squeeze it back into shape (this only works for paths and bitmaps), as shown in Figure 17-5.

Figure 17-5: Motion blur, step-by-step

You can also create nonuniform motion blur by setting different horizontal and vertical blur values in the Filters ▸ Blurs ▸ Blur... adjustable filter dialog, but the stretch/squeeze method described here is usually easier.

> **TRANSFORMED BLUR IN SVG**
>
> At the SVG level, to make blur nonuniform, you need to apply it to an object with a `transform` attribute (A.7), which affects the blur as well. When you stretch a path without blur, this stretching is by default embedded into the points of the path (A.7), but once you apply the blur, reverse squeezing will instead add the `transform` attribute, squeezing the blur as well.

To create motion blur on a group or text object (that is, not a path or bitmap), use this trick. Stretch the object, then group it with itself (Ctrl-G, 4.8), apply a blur to the group, and squeeze it back. The grouping step ensures that in the end, you get an object with the correct proportions but also with the squeezing `transform` attribute affecting the blur.

Similar remarks apply to the way many other filters behave when transformed. For example, squeezing and/or rotating objects allows you to create perspective-distorted variants of texture filters (17.4) for applying to walls and floors in 3D scenes (Chapter 22). Of course, transforming filters by stretching and squeezing objects is not as convenient as dragging handles on canvas would be—but it works.

NOTE *Filters work on the rendered image of the object on your screen—that is, any filter gets as input a bitmap that is created for the current zoom level. This is why filters may be slow to work with: every time you zoom in or out, all the filters in your view are reapplied to the new rendering. This also explains why some filters may look slightly different depending on the zoom. For example, a sharpening filter, when applied to a bitmap object at 100 percent zoom, works on the image features as you would expect. However, if you zoom in closely enough, you will see the pixels of the bitmap object as rectangles (18.2.3), and the sharpening will now apply to the borders of those rectangular pixels!*

17.1.2 Tweaking for Blur

The Blur mode of the Tweak tool blurs the selected objects under the brush more (by default) or less (with Shift pressed). The amount of blur added or removed depends on Force, pen pressure (if you're using a tablet pen), the closeness of the object to the center of brush, and how long you apply the brush.

17.2 Blend Modes

[1.1] Choosing a blend mode affects how the colors of an object *blend* (mix) with those of the background objects beneath it. Like blur, blend modes are easily accessible in the Layers and Fill and Stroke dialogs—where you can change them for layers and for individual objects, correspondingly.

There are 16 blend modes, including the Normal mode that all objects have by default. Each blend mode is basically a formula that calculates the visible color in every point where two or more objects are stacked on top of each other. In the default Normal mode, the color of the background object is taken into account only if the foreground object has less than 100 percent opacity. In all other modes, however, even if the foreground is fully opaque, the background may still show through in some way (depending on the mode and the color of the top object).

You may be familiar with how blending modes work from bitmap graphic editors, such as GIMP or Adobe Photoshop, that let you change how each layer of a drawing blends into the layers below it. In Inkscape, blend modes are also often used for tweaking the look of imported bitmaps with colored "lenses" using various modes, and in this way, they are indispensable—you cannot achieve quite the same effects with Normal-mode colored transparencies. You can adjust the colors of a bitmap using an external bitmap editor (Inkscape will even re-read a linked image that was changed by an external program, 18.2.1), or you can use a predefined composite filter, such as Sepia. In most cases, however, an easier and more flexible approach is to place a flat-color or gradient overlay on top of an image, change the blend mode of that overlay object, and adjust its opacity for the force of the effect.

I will not overwhelm you with details and formulae of all the modes; instead, I will try to meaningfully group and contrast various modes and give some practical tips on how to achieve common useful effects. Refer to Figure 15 in the color insert that demonstrates applying four colored overlays (blue, orange, black, and white) to *Mona Lisa*. The top row shows the original bitmap and two Normal-mode overlays: one with 100 percent opacity (not transparent at all) and

another with 50 percent opacity. All other blend modes below are shown at 100 percent foreground opacity; if you lower it, each mode's effect will be reduced in a way you would expect.

Color Dodge, Color Burn, Hard Light, and Soft Light form two contrasting pairs, with Color Dodge somewhat similar to Hard Light, and Color Burn to Soft Light. In this group, the two modes on the left (Color Dodge and Hard Light) either lighten the background colors or render the foreground opaque; the Color Burn either darkens the background or renders foreground opaque. Finally, Soft Light does something meaningful for all four sample foreground colors; of all the modes, it is the most similar to the partially opaque Normal mode, but it is in fact more useful. Where Normal mixes colors by fading out the background, Soft Light tints the background without dulling its saturation and contrast.

Multiply, Screen, Darken, and Lighten is another group of four modes that fall into two contrasting pairs: Multiply is quite similar to Darken, and Screen is similar to Lighten. Here, the two modes on the left (Multiply and Darken) make the background darker if the foreground is dark, all the way to rendering it black; those on the right (Screen and Lighten) make the background lighter, all the way to white. This means a white foreground has no effect in Multiply and Darken, and a black foreground is invisible in Screen and Lighten. The Darken and Lighten modes just choose whichever of the background and foreground colors is darker or lighter, whereas Multiply and Screen actually combine the two colors.

Overlay is my favorite mode; I often use it with a scattering of elliptic-gradient lenses to deepen highlights (with white) and shadows (with black) in a photo as well as to make it sunnier or gloomier overall. It also works very well for tinting, mixing in the foreground color without fading out the background (similar to Soft Light) but at the same time emphasizing the lights or shades in an image (depending on whether the foreground color is light or dark).

Difference and Exclusion are two modes that, instead of adding, *subtract* the foreground from the background (using slightly different formulae). Thus, a black foreground has no effect (because black is "zero color" in RGB), whereas a white foreground *inverts* the background. Other colors produce various "tinted inversion" effects, more dramatic for light colors than for dark ones.

The four modes in the last group, Hue, Saturation, Color, and Luminosity, are different from all the other modes in that they act not on the RGB channels of the background and foreground colors, but convert them to the HSL model (8.4.3) and combine the HSL channels in various ways.

- The Hue mode combines the foreground hue and the saturation and lightness of the background. Unlike all other approaches to tinting, this one does not use the background's own hue at all; for a flat-color overlay, it creates a purely monochromatic image out of a de-hued version of the background and the foreground hue.
- The Saturation mode applies the foreground saturation to the hue and lightness of the background. This is a natural way to vary saturation of an image, from full desaturation (if the foreground has zero saturation, such as white or black) to psychedelically oversaturated colors if the foreground has fully saturated color (such as bright orange).

- The Color mode takes the hue and saturation from the foreground and combines them with the background lightness. Think of it as a combination of Hue (tints the background) and Saturation (adjusts background saturation) modes.
- Finally, the Luminosity mode takes the luminosity (lightness) of the foreground and combines it with the hue and saturation of the background. If your foreground is a flat color and the background is an image, as in Figure 15 in the color insert, Luminosity renders the image in its own colors but equalizes all of them to the same level of lightness. The result is rarely useful—or even discernible—because it's the relative lightness of pixels, not their hue or saturation, that carries most of the visual information in an image.

BLEND MODES IN SVG

In past versions of Inkscape, a subset of blend modes (applicable only to layers, not individual objects) was implemented via filters using the `feBlend` primitive (which is why they ended up in this chapter). Now, Inkscape uses the `mix-blend-mode` CSS property instead. It is defined in the (still draft as of this writing) *Compositing and Blending Level 1* specification (*https://drafts.fxtf.org/compositing-1/*), which is planned to become part of CSS and SVG.

17.3 Filter Management

A filter applied to an object is part of its style. As such, one object's filter can be copied to another object by copying and then pasting style (Chapter 8). The name of a filter for a selected object is reported in the status bar, for example: *Group of 1 object; filtered (Specular Light)*.

In SVG, an object can have only a single filter applied. However, most filters consist of several filter primitives stacked on top of one another, and you can always combine two or more individual filters by putting one filter's stack of primitives on top of another's. When you apply a preset filter (17.4) to an object that already has a filter, Inkscape is smart enough to combine their primitives for you so that you get the desired combined effect of both filters (for example, a texture overlay and a drop shadow).

If, for some reason, automatic combining of filters does not work for you, use the following trick. Apply one filter, then group the object with itself (Ctrl-G) and apply the second filter to the group. By repeating this operation, you can process your object with any number of filters in any order. You can even double down on a filter by applying it to your object twice (although in most cases, if a filter appears too weak, it is better to go into the Filter Editor, 17.5, and try to figure out which parameters of which primitives to adjust to crank it up).

If the scale of the applied filter is not what you need (for example, if the bubbles or feathered edges are too large or too small for your object) and the filter does not allow you to adjust that with a dialog, you can use this simple trick: scale your object up (or down)—for example, by pressing Ctrl-< or Ctrl-> a few times—then apply the filter and scale the result back down (or up) by the same multiplier.

To remove any filters applied to any of the selected objects, use the **Filters ▸ Remove Filters** command.

[1.1] 17.3.1 Editing the Filter Area

As you've already seen in the section on blurring (17.1), applying a filter may expand an object's visual bounding box. This is true for other filters besides blur that need to paint, scatter, or shift the image beyond the original bounding box. In SVG, any filtered object has an associated *filter area*—the virtual canvas on which the filter is allowed to draw. It is the size of the filter area that Inkscape treats as the new visual bounding box of the filtered object.

Usually, you don't need to worry about this. Inkscape sets the filter area size automatically when you blur an object or choose one of the preset filters (17.4), based on the nature of each filter (for example, extending the pre-filter bounding box by the blur radius). Most of the time this works perfectly—but sometimes, especially when you're combining filters or copy-pasting a filter to a different object, this automatic filter area may cause clipping of the filter's outer fringes so you need to manually fix it.

The easiest way to edit the filter area is to switch to the Node tool (12.5) or any of the shape tools (Chapter 11) and drag any of the two small diamond-shaped handles in the top-left and bottom-right corners of the visual bounding box, as shown in Figure 17-6. Usually you extend the filter area to prevent clipping, but you can also use this approach to clip your filtered object if that's what you need (in that case, however, watch for any other objects that, due to duplication or style pasting, use the same filter—they may end up clipped too).

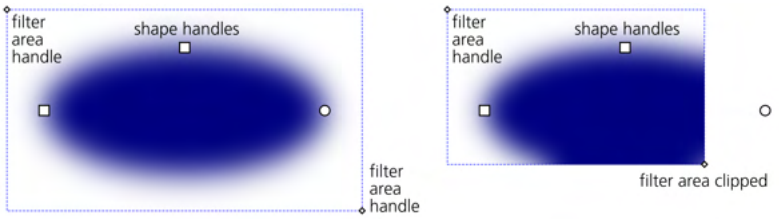


Figure 17-6: Editing the filter area via handles in the Node tool

Interactive editing of the filter area does not work for clones and symbols. You can also edit the area of any filter numerically in the Filter Editor dialog (17.5.4).

17.4 Preset Filters

SVG defines a limited number of filter primitives out of which you can build arbitrarily complex composite filters. Before going that way, however, let's look at an impressive collection of preset composite filters that come with Inkscape.

To apply a preset filter to a selected object or objects, simply choose a command from the submenus of the **Filters** menu. To get a brief description of an individual preset filter (or of any menu command, for that matter), just hover the mouse over its command in the menu and read the status bar, as shown in Figure 17-7.

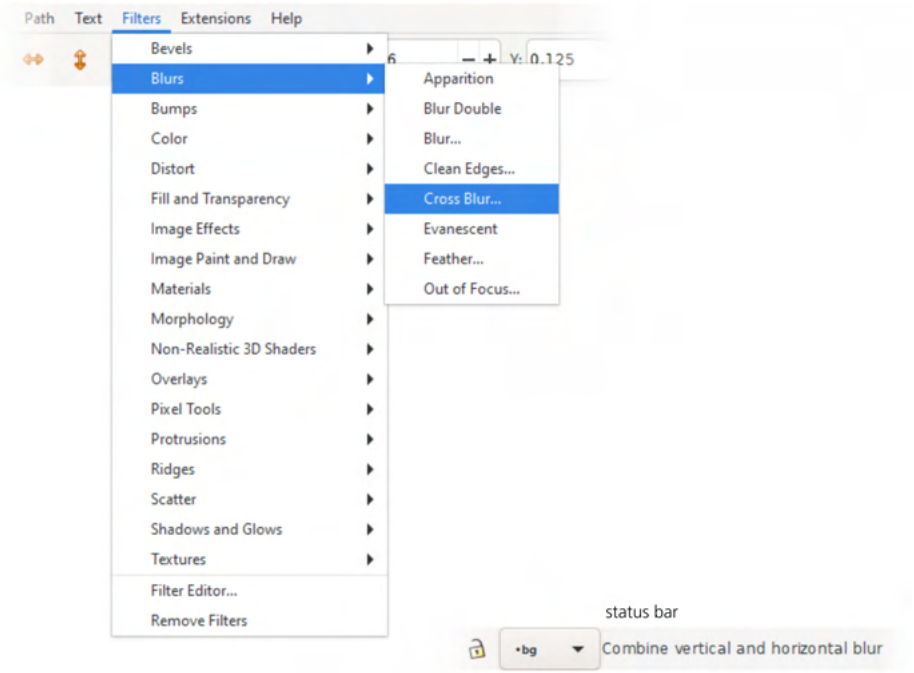


Figure 17-7: Preset filter effects

Most filters will apply immediately. Others—those with an " . . . " at the end of their names in the menu—will display a dialog where you can adjust their parameters. Such a dialog always has the Live preview checkbox that you can turn on to view the result on canvas updated live for any changes of parameters (however, this checkbox locks the canvas so you can't select a different object, nor can you even scroll or zoom). Clicking Apply creates the filter with the current parameters but does not close the dialog (so you can select another object to apply it to); Close cancels the dialog and the preview (no need to undo).

Figure 17-8 shows an example of a pretty complex adjustable effect whose dialog contains several numeric parameters and a color chooser.

I don't describe all preset filters here—there are just too many of them (and more are added with each Inkscape version). However, I do describe each of the submenus of the Filters menu and give brief notes and illustrations of the most notable filters in each submenu (usually the adjustable ones).

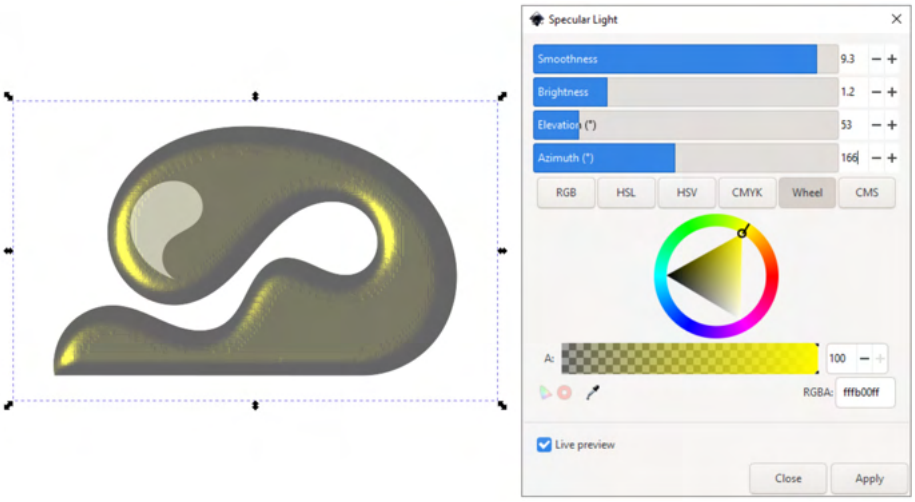


Figure 17-8: Dialog of an adjustable preset filter

NOTE *All the preset filters are stored in the file* filters.svg *in the* inkscape/filters *folder of your Inkscape data folder (look it up in the* Preferences, System *page). You can add your own preset filters to the menu either by editing* filters.svg *or by placing another SVG file with your filters in the same* inkscape/filters *folder. After that, any filters defined in that file will be listed in a* Bundled *submenu under* Filters. *(All such filters will apply immediately; you cannot create an adjustable filter with a dialog.)*

17.4.1 The Bevels Submenu

The Bevels submenu presents many variations on the theme of bevels (pseudo-3D raised edges), as shown in Figure 17-9. Here you will find opaque and translucent materials, glossy and matte finishes, bevels lit by multiple sources of light, with and without shadows, with depressions in the middle and with raised borders, and so on. Most of these filters preserve the original color of the object, though they may make it lighter or darker in places for the 3D effect.

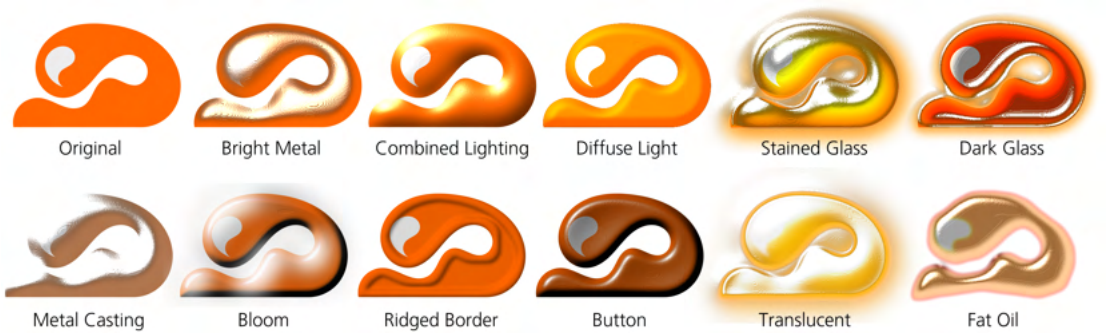


Figure 17-9: Filters from the Bevels submenu

Most of the adjustable filters in this group let you specify how raised the bevel is (Smoothness), the coordinates of the light source (Elevation and Azimuth), as well as the Brightness and color of the highlight. If the filter you've chosen is not adjustable, you can still move the light source if you apply it, open the Filter Editor, select the Specular Light or Diffuse Light primitive in the filter stack applied to the object, and play with the Elevation and Azimuth parameters there.

17.4.2 The Blurs Submenu

The Blurs submenu (Figure 17-10) includes the basic adjustable Blur where you can set horizontal and vertical blur amounts separately. If you want a non-horizontal and non-vertical motion blur, rotate your object to make its motion direction horizontal, blur it with horizontal blur, and then rotate back.

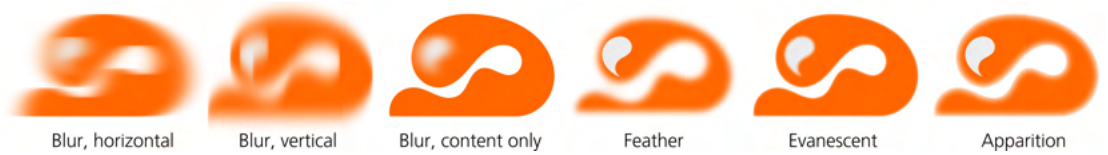


Figure 17-10: Filters from the Blurs submenu

The Blur content only checkbox in Blur blurs only the inside of an object (for example, the image on a bitmap) but leaves its edges (where its opacity drops to zero) unaffected. The Feather effect does the opposite: it blurs the transparency mask of an object but does not affect its content.

Apparition and Evanescent overlay a faded-out blurred copy of an object over a normal copy to make it look softer and a little unreal without blurring it to unrecognizability; of these, Apparition works on the object's edge (like Feather), while Evanescent works on its content. Finally, Out of Focus attempts to model the real-world myopic view of an object's content a little better than a plain blur would.

17.4.3 The Bumps Submenu

The Bumps submenu (Figure 17-11) has filters that add various pseudo-3D textures that emphasize the features in an image. They are similar to the "embossing" effect you may have seen in other software.

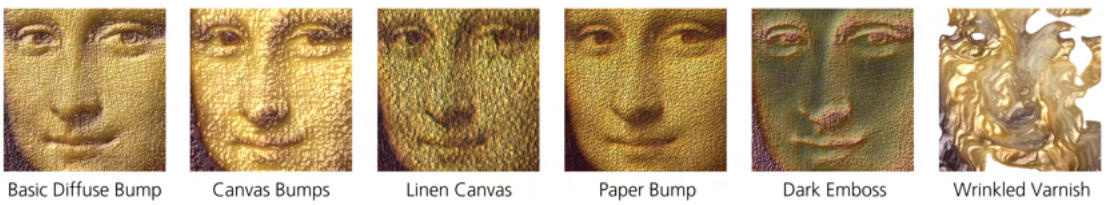


Figure 17-11: Filters from the Bumps submenu

17.4.4 The Color Submenu

The Color submenu collects various ways to adjust or transform an object's colors. They are most useful for imported bitmaps, but you can also use them on vector designs, especially complex ones that you can group and filter as a whole without having to repaint each object separately.

Personally, I prefer to use semitransparent overlays with blend modes (17.2), because with them, I have full control over which parts of an image to affect, but there are cases where blend modes just don't cut it. Still, this submenu includes Simple Blend that emulates any of the blend modes as an effect. You can also:

- Adjust Lightness-Contrast.
- Invert per-channel.
- Convert to Grayscale controlling how much each of the RGB channels contributes to the result.
- Use Color Shift to rotate hues.
- Emulate various types of Color Blindness.
- Render an image in two (Duochrome) or three (Trichrome) colors.
- Adjustably Colorize an image with a chosen color.
- Make colors Fluorescent.
- Extract Channel to get one of the RGB or CMY channels, or Nudge them relative to one another for that "misaligned print" antique look.

17.4.5 The Distort Submenu

The Distort submenu's filters apply random distortions to an object's content, transparency mask, or both (Figure 17-12). This includes the versatile adjustable Roughen where you can set the frequency and amplitude of a random ripple, as well as a number of preset variations on this theme.

Figure 17-12: Filters from the Distort submenu

17.4.6 The Fill and Transparency Submenu

The Fill and Transparency submenu contains several essential filters:

- Channel Transparency (adjustable) makes parts of an object transparent depending on RGB channel values (for example, to make holes in areas where Red is above a certain threshold).
- Fast Crop does not change the look of the object in any way but adds a rectangular crop frame with X-shaped handles in the top-left and bottom-right corners. Drag the handles to nondestructively crop the object.

- Light Eraser renders white or light-colored areas of an object transparent (with the Invert checkbox, it renders everything except light-colored areas transparent).
- Fill Background fills the transparent parts of an object's bounding box (plus extensions, 17.3.1) with an opaque color (this filter is not adjustable, use the Filter Editor to change the color); Flatten Transparency does the same with white.
- Opacity (adjustable) thresholds an object's opacity, making all blurred or transparency-gradient edges crisp.
- Silhouette (adjustable) replaces all colors in an object by black or another color of your choice.

17.4.7 The Image Effects Submenu

The Image Effects submenu contains a few classic bitmap processing algorithms, including Edge Detect (renders a black-and-white version of the image showing the color boundaries in the original) and Sharpen (emphasizes color boundaries by increasing their contrast).

17.4.8 The Image Paint and Draw Submenu

The Image Paint and Draw submenu collects filters that emulate various painting and drawing techniques and materials (Figure 17-13). Not all of these emulations are convincing, but many can be inspiring.

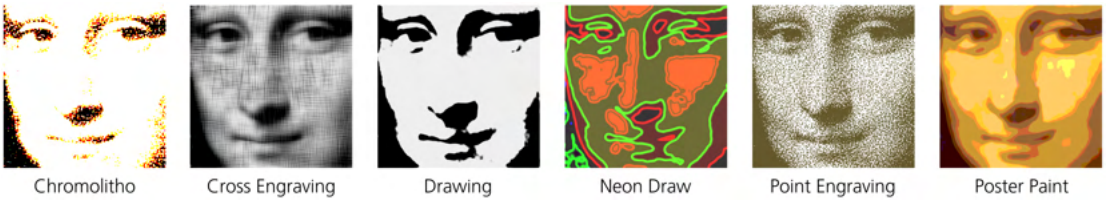


Figure 17-13: Filters from the Image Paint and Draw submenu

17.4.9 The Materials Submenu

The Materials submenu's filters imitate various materials, tinted by the object's color and clipped by its opacity mask (Figure 17-14). Those with "3D" in the name also add a bevel at the edge.

Figure 17-14: Filters from the Materials submenu

17.4.10 The Morphology Submenu

Morphology is Greek for "studying forms"; these filters imitate inset/outset (12.4) and various kinds of filling and stroking, as shown in Figure 17-15.

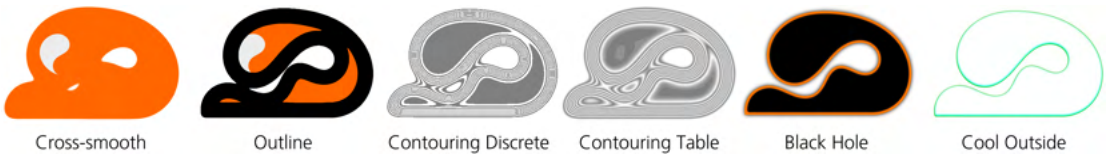


Figure 17-15: Filters from the Morphology submenu

17.4.11 The Non-Realistic 3D Shaders Submenu

The Non-Realistic 3D Shaders, used on flat-colored shapes, are similar to bevels but are thicker, go further in from the edge, and have a much higher contrast between light and shade (hence "non-realistic"), as Figure 17-16 demonstrates.

Figure 17-16: Filters from the Non-Realistic 3D Shaders submenu

17.4.12 The Overlays Submenu

Overlays do not change the object itself but treat it as a pad for displaying some kind of texture, adding it as an overlay on the object's original color (and sometimes fully obscuring it), as shown in Figure 17-17. This group includes the generic adjustable Noise Fill.

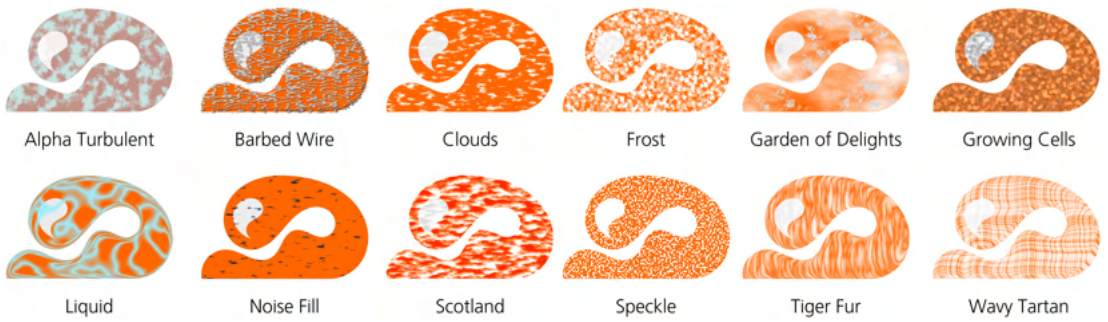


Figure 17-17: Filters from the Overlays submenu

17.4.13 The Pixel Tools Submenu

Pixel Tools has a single filter, Pixelize, which suppresses pixel-level anti-aliasing (see Figure 1-1 on page 2). It does not rasterize vector shapes—but with this filter, at any level of zoom, they have perfectly crisp (and usually visibly jagged) edges.

17.4.14 The Protrusions and Ridges Submenus

Protrusions are fun filters that add various extensions to your shapes, such as dripping liquid or fire. The Ridges submenu filters turn an object into a narrow ridge along the edge and treat this ridge in various ways (Figure 17-18).

Figure 17-18: Filters from the Protrusions and Ridges submenus

17.4.15 The Scatter and Shadows and Glows Submenus

The Scatter submenu filters explode an object into variously shaped fragments—tree leaves, cubes (looking more like squares), random spray splotches—and scatter those fragments randomly around.

The Shadows and Glows submenu is the place to go for your trusty all-purpose Drop Shadow: the adjustable filter lets you select the color and opacity of the shadow (or glow), its displacement relative to the object, and the blur radius (Figure 17-19). This group also includes a few other filters implementing inner and outer shadows and glows and their combinations.

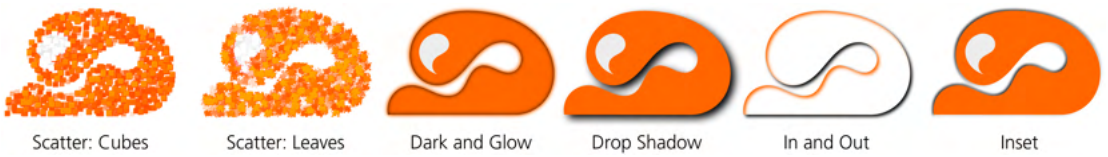


Figure 17-19: Filters from the Scatter and Shadows and Glows submenus

17.4.16 The Textures Submenu

The Textures submenu contains various naturalistic textures: crumpled plastic, jam spread, bark, horizontally striped carpet, and so on (Figure 17-20).

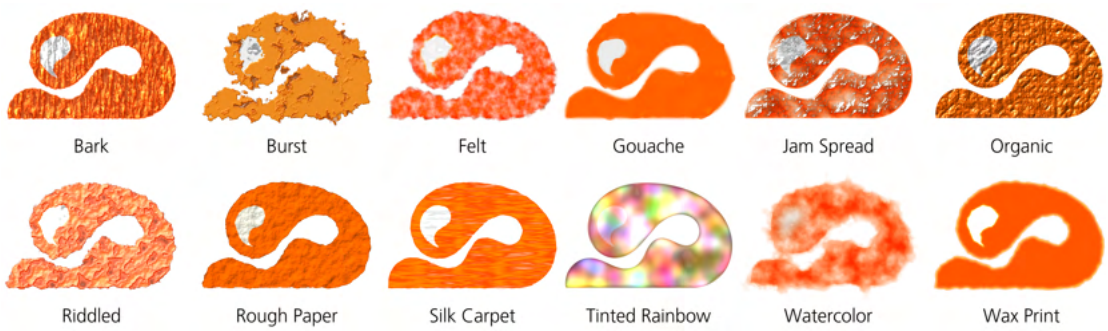


Figure 17-20: Filters from the Textures submenu

17.5 The Filter Editor Dialog

The preset filters that come with Inkscape are useful not only by themselves but also as starting points for your own derivative filters. Choosing one of the preset filters—closest to what you want to get—and working from it is usually easier than starting from scratch. Let's look at the Filter Editor dialog (Filters ▸ Filter Editor... from the menu) shown in Figure 17-21.

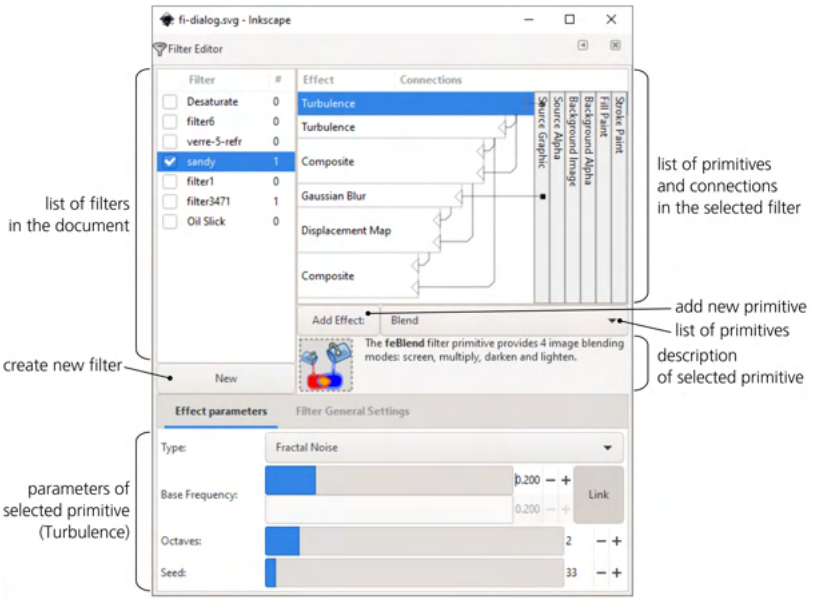


Figure 17-21: Major areas of the Filter Editor dialog

17.5.1 The Filters List

The top left pane of the dialog lists all filters defined in your document; select any of them and edit its structure and parameters in the rest of the dialog. If the currently selected object has a filter applied, that filter will show a check mark in the list—by setting or removing the check mark, you can apply or unapply any filter to any object.

Right-click a filter and choose **Select** to select all objects in the document that use this filter. Filters not currently used by any object remain in the list; to remove unused filters from the document, use the **File ▸ Clean Up Document** command.

Under the list, the New button adds a new filter. If you open this dialog in an empty document, the list of filters is also empty. Click **New** to create a new empty filter—usually called `filter1` (to rename it, just click its name in the list twice and edit the name in place). You can also duplicate or delete a filter in the list by right-clicking it and using the corresponding pop-up menu commands.

Now, create or select some object in the document and check the checkbox for your new filter. Nothing changes—an empty filter does not affect the rendering of an object. For the filter to actually do something, you need to add some primitives to it.

17.5.2 The Stack of Primitives

The area to the right of the filter list is the main filter construction board, where you list, arrange, and connect the stack of *primitives* that constitute a filter. This will be empty until you add primitives to a filter.

The 16 primitives supported by Inkscape are listed in a drop-down list next to the Add Effect button. When you choose a primitive in the list, it displays a brief description and illustration below—read those descriptions to get a rough idea of what each primitive does. A detailed explanation of all primitives is beyond the scope of this book; refer to the SVG specification (*http://w3.org/TR/SVG11/*) for complete details.

Instead, let's look at the step-by-step process of creating a fairly complex filter that uses several different primitives. My filter is called Sandy Blur; I designed it for the background coloring of the rose image (see Figure 25-5 on page 454). This filter aims to imitate blurry watercolor strokes on rough paper.

Plain Gaussian blur would not work for me: it is too smooth, too obviously computer-generated, too boring. For an acceptable watercolor imitation, I needed to model both the small-scale roughness of the paper texture and the larger-scale "blotches" resulting from the watercolor paint flowing and sticking differently in different spots. For both those sources of unevenness, I used the Turbulence primitive that creates random fractal noise at a given scale (Figure 17-22).

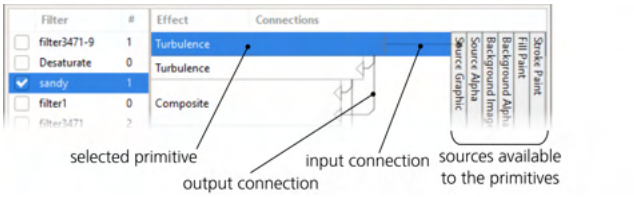


Figure 17-22: The two Turbulence primitives

The two Turbulence primitives are the first two components of the filter—the two topmost boxes in the stack. Each filter primitive has one or more *inputs* and one *output*; in the list, inputs are depicted by lines coming into the primitive box horizontally from the right, and the output is the line going from the box vertically downward. You can wire the connections by dragging, starting from an input triangle at the right-hand edge of a primitive's box.

As you can see, the outputs of some primitives are inputs for others. Generally, information flows top to bottom in the stack of primitives, and the output of the bottommost primitive is what you see rendered in the document window. You can rearrange the primitives by dragging; a right-click menu allows you to duplicate or delete a primitive.

What are the vertically stacked boxes on the right? These are the predefined sources that any primitive can use as input. The most useful of these are Source Graphic and Source Alpha. Source Graphic, as the name implies, supplies the rendered image of the object being filtered, at the current zoom's resolution. The Source Alpha provides a grayscale representation of the original object's opacity (alpha) mask; points that are fully opaque (regardless of color) in the Source Graphic will be opaque black on the Source Alpha image, and points that are transparent will be transparent black.

17.5.3 Parameters of a Primitive

Let's look again at the two Turbulence primitives at the top of the stack. They both take Source Alpha as input and pass their result—random noise—down to other primitives. What's different is the parameters of these filters.

When you select a filter primitive in the stack, its parameters are displayed in the bottom pane of the dialog. Figure 17-23 shows the parameters of the two Turbulence primitives.

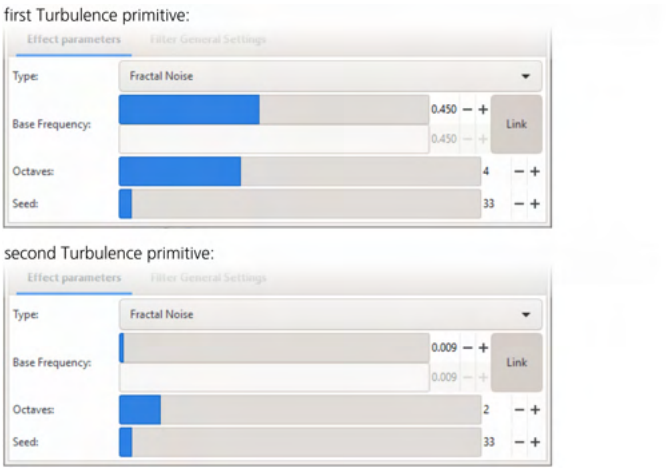


Figure 17-23: The parameters of the two Turbulence primitives

These two primitives have the same Type (Fractal Noise, which looks better in this case than the other option, Turbulence), but they differ in Base Frequency and Octaves. The Base Frequency parameter sets the scale of the turbulence; higher frequency results in smaller, sand-like texture, while lower frequency produces larger clouds. The number of Octaves specifies how deep the recursion is in the algorithm: increasing Octaves produces sharper unevenness with more small details; decreasing this value gives a smoother, more nebulous image.

Now, to combine the two turbulence outputs, I used another primitive named Composite, as shown in Figure 17-24.

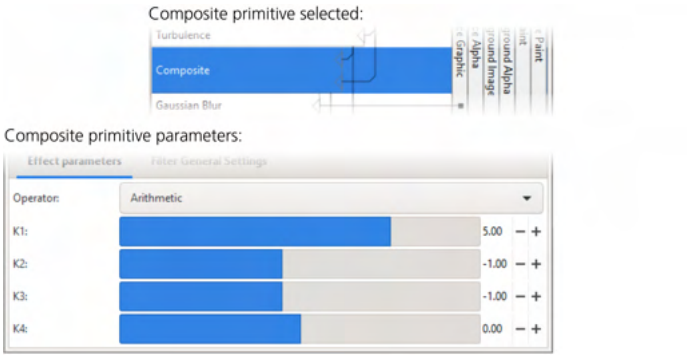


Figure 17-24: The Composite primitive

It has two inputs and combines them, pixel by pixel, using one of a number of methods. Here, I used the Arithmetic method; with the numeric values of the four coefficients (K1 to K4) as shown, it results in the large-scale wave and small-scale ripples being combined into a composite with somewhat increased contrast, as Figure 17-25 demonstrates.

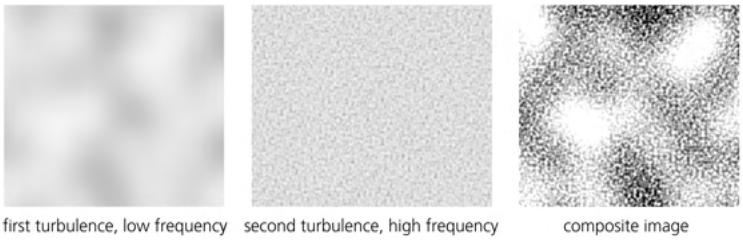


Figure 17-25: The result of compositing two turbulences

So far, I didn't use the image of the object itself to which the filter will apply—that is, I didn't use the Source Graphic. Supposedly, that object would be some kind of a broad brush-like path, likely created with the Calligraphic pen tool (14.2). Of course, the first thing to do to a flat-color, crisp-edged path to make it more like a watercolor stroke is to blur it, as shown in Figure 17-26.

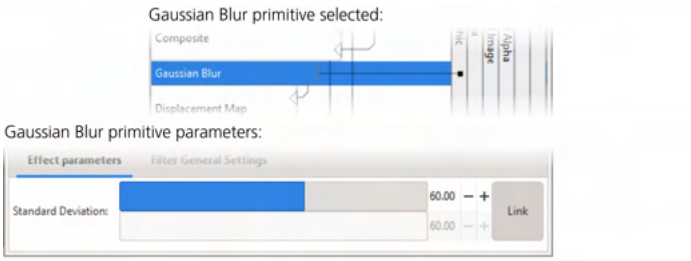


Figure 17-26: The Gaussian Blur primitive

The Standard Deviation parameter of Gaussian Blur is similar to the blur amount you would adjust in the Fill and Stroke dialog (Figure 17-1 on page 354), except that here it is measured in absolute px units (A.6), not as a percentage of the object's size.

How to combine the blurred stroke with the turbulence? Composite won't work here. Any of its modes will result in a smooth blur being *overlaid* with the turbulent ripples, the underlying smoothness not going anywhere (try it). It won't look like watercolor on a rough paper—more like computer-produced blur viewed through a spotted glass.

Consider what happens when you paint with a real brush on a real paper. The blurriness of your stroke results from the softness of the brush—you apply more pressure in the middle than on the edges of the brush. When a brush meets a dimple in the paper, that pressure is changed: if this area of the paper is higher, it will get more paint, as if it were closer to the maximum-pressure point of the brush; if it is a depression, it will get less paint. In other words, the roughness of the paper jitters the blurred stroke *in the plane of the drawing*, as if randomly displacing parts of the stroke sideways. How do you achieve this with filters?

The Displacement Map primitive is a perfect match for the job. It takes its first input and moves its pixels around according to its second input. You can choose which of the channels (Red, Green, Blue, or Alpha) of the second input will move the first one in each of the two axis directions (X and Y), as well as the scale of this displacement. In this case, the first input was the blurred object, and the second input was the composite turbulence field, of which I took the Alpha channel for both axes (Figure 17-27).

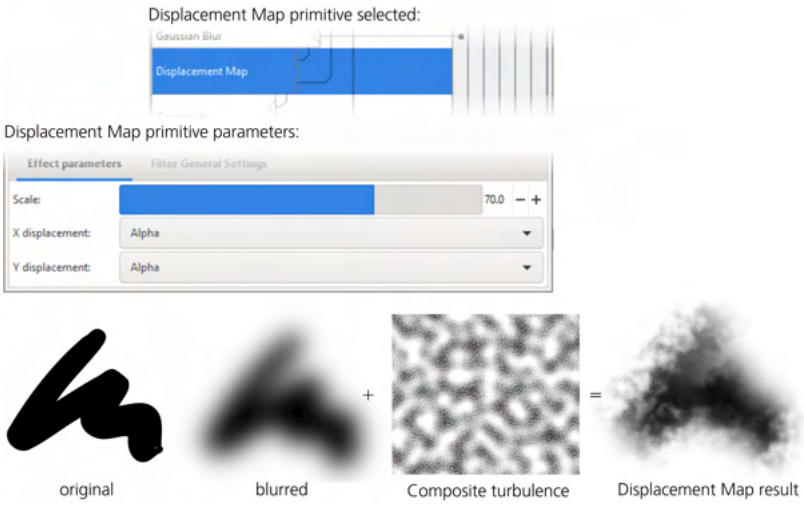


Figure 17-27: The Displacement Map primitive

As an added bonus, the result looks rougher in some areas and smoother in others. This is because in the smooth areas, the displacement map hits the almost-flat inner parts of the blurred stroke—where the high-frequency jitter just moves around pixels of almost the same color, which barely disturbs the

smoothness. On the edges of the stroke, however, different-colored pixels are mixed and jittered, producing visible roughness. All this is additionally modulated by the low-frequency noise, producing a convincing watercolor simulation.

However, if you apply this filter to a light-colored stroke (and not black, as in Figure 17-27), the result is still not perfect. When you're looking at a real rough paper with watercolor strokes on it, what you see is not just the spatial distribution of paint; you also see the roughness itself, which looks like a pattern of shades. Without this shading, light-colored strokes with this filter still look too unnaturally flat. Fortunately, that was easy to fix; I already had the high-frequency turbulence source, which worked nicely as shading once I composed it with the displacement-mapped blur (Figure 17-28).

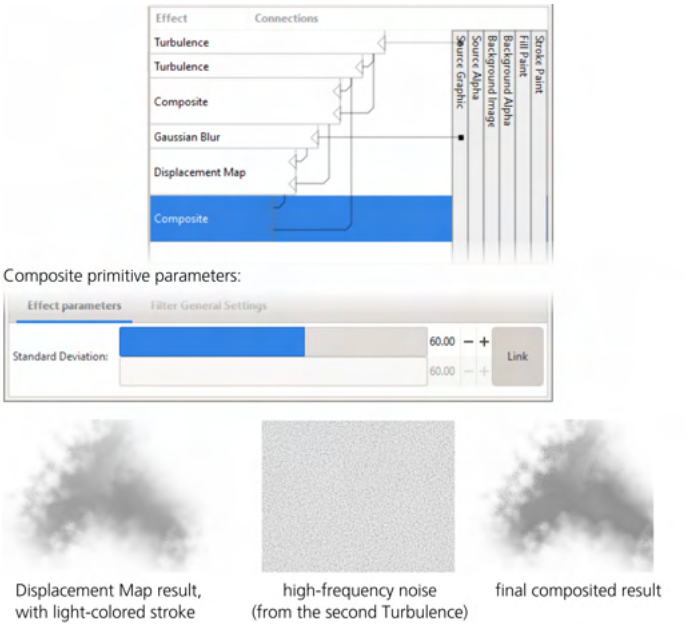


Figure 17-28: The final filter and the parameters of the last Composite primitive that overlays the shading

At this point, the filter looked pretty decent. I could keep improving on it, adding various smears, water leaks, diffusion, and so on; however, since my demo image used this filter only for the background, it was good enough for the purpose.

17.5.4 The Filter Area

The second tab in the filter parameter area in the Filter Editor dialog is called Filter General Settings. It contains parameters that apply to the entire filter stack, not any single primitive. Currently, the only thing you can change here is the *filter area*—the area that the filter will render into, measured in the units of the bounding box of the object to which the filter is applied. You can edit this area interactively on canvas (17.3.1), but the Filter Editor allows you to set it numerically.

The Coordinates widgets specify the top-left corner of the area, and the Dimensions specify the bottom-right corner. All values are measured in the units of the pre-filter bounding box size.

For example, if you set the Coordinates to 0/0 in and the Dimensions to 1/1, the area will be equal to the pre-filter bounding box. This will work if your filter does not reach beyond the object—for example, if it's just a color change with a Color Matrix primitive. However, for anything like blur that paints outside the bounding box, you need to provide sufficient margins for this to prevent cropping. When you apply blur via the Fill and Stroke dialog or use one of the preset filters, these margins are set for you automatically; however, when creating a new filter stack from scratch, you must take care of it yourself. The default is 0.1/0.1 for Coordinates and 1.2/1.2 for Dimensions, which results in 10 percent margins on all sides of the object, as shown in Figure 17-29.

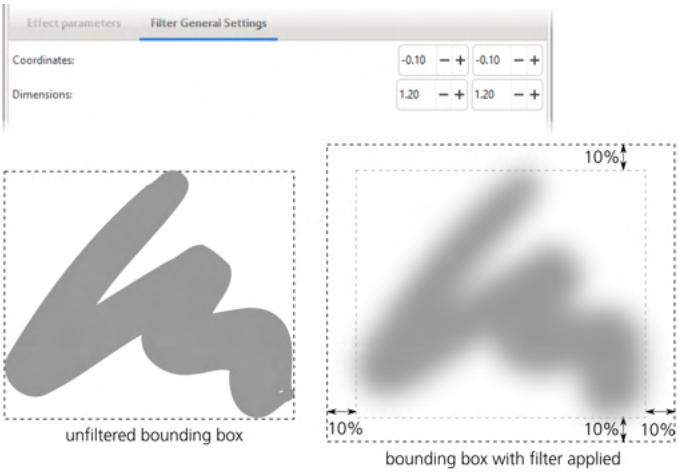


Figure 17-29: Specifying the filter area

If Inkscape uses the visual bounding box (which is the default, 4.3), the bounding box of a filtered object—the frame you see when you select it in the Selector—is the enlarged filter area. Among other things, this means that, when exporting a selected object to bitmap (18.6), Inkscape will make sure the bitmap contains the blurred edges without cropping.

17.6 Filter Rendering Options

The rendering speed of Inkscape filters has been greatly improved over the years. Still, with complex or large-area filters it may still be an issue. That's why Inkscape has a number of preference options to tweak the performance of filters (and rendering in general).

First, you can always switch to Outline mode (3.14) to speed up rendering while working on a document. There is also the No Filters mode, which is the same as normal mode except that filters aren't rendered; this is perhaps the easiest workaround if filters are the main source of slowness for you.

[1.1] Options that affect rendering speed are collected on the Rendering page of the Preferences dialog (Figure 17-30). You can set the Number of threads (simultaneously executing rendering pipelines), four by default; it is often recommended to set this equal to the number of logical processors your CPU has (in my case it is eight). You can also increase the Rendering cache (64MB by default) and enlarge the Rendering tile multiplier (16 by default). If you have modern enough hardware and plentiful RAM, it is worth increasing these values to see how much of an improvement you can get. Also, if you notice slowdowns when the screen is updated during editing (for example, when you're dragging nodes), try switching Redraw while editing from the default Responsive to Conservative.

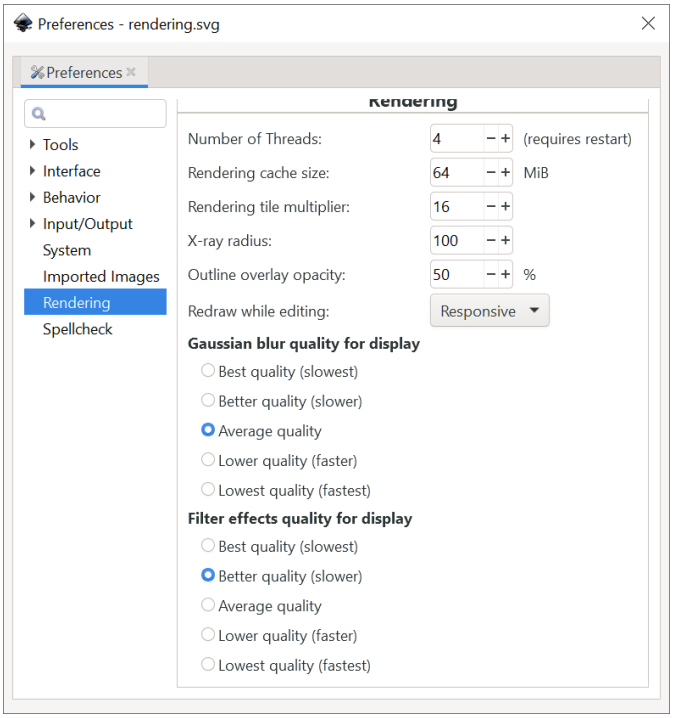


Figure 17-30: The Rendering page of the Preferences dialog

Options that apply specifically to filters include the two groups of radio buttons: Gaussian blur quality for display and Filter effects quality for display—this is where you search for the speed/quality tradeoff that works best for you. The first group affects only blur; the default, Average quality, looks *almost* perfect and renders reasonably fast. At lower quality settings, visible defects appear, but rendering becomes even faster; the higher settings bring little quality improvement but are significantly slower.

The second set of radio buttons applies to all filters (including blur). It also offers to trade an improvement in rendering speed for worse rendering quality, but its speed advantage is less significant, so here the Better quality option is probably optimal.

NOTE *Bitmap export (18.6) as well as rasterization of filters for PS/EPS/PDF export (17.7) always use the highest quality setting, which is why exporting is usually slower than rendering the same image on the screen at the same resolution.*

17.7 Exporting Filters to PS and PDF

PostScript and PDF formats, while being vector, do not support anything like SVG's filters. By default, during export, Inkscape offers to rasterize (convert into a bitmap) any object with a filter applied. This fully preserves the appearance the filter had in Inkscape, but it may increase the file size significantly. In the export options dialog, you can specify the resolution of the bitmaps, as well as turn rasterization off, in which case objects remain vectors but lose any filtering (Figure 17-31).

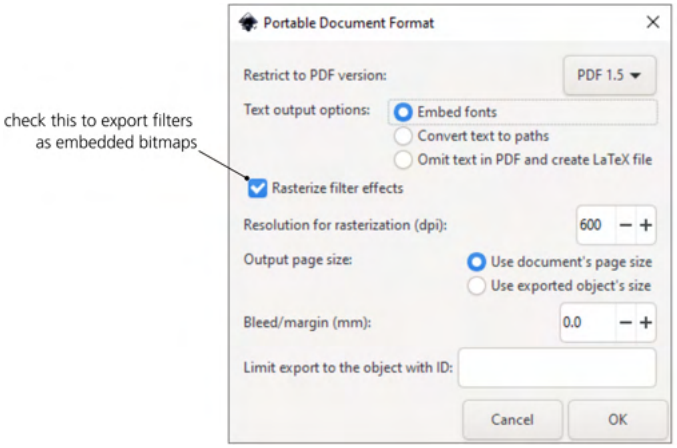


Figure 17-31: Specifying the treatment of filters when saving a document as PDF

When exporting via the command line (C.4), the corresponding options are `--export-dpi` to specify the resolution and `--export-ignore-filters` to turn off rasterization.

18

BITMAPS

Most vector images start or end (often both) their lives as bitmaps (1.1), and Inkscape's SVG documents are no exception. Many vector drawings, artistic as well as technical, are developed from tracings of photos, scans, or other bitmap drawings; at the other end of their lifetimes, almost all vector pieces are exported into bitmap formats for viewing in software that cannot deal directly with vectors. Bitmaps are an important object type in Inkscape, and the techniques for dealing with them are sufficiently complex and interesting to devote an entire chapter to them.

18.1 Bitmap as Object

If you have a bitmap image file, insert it into your Inkscape document with the File ▸ Import command. You can also File ▸ Open any bitmap file as a document in its own right. In this case, Inkscape automatically creates a new SVG document, its page size (3.5.2) matching the pixel size of the bitmap, and places the bitmap on the canvas (into the document root—that is, not in any layer). You can now add other objects to that document and save it as SVG (if you want to get a bitmap

with the result, you need to export it, 18.6), or you can copy and paste the bitmap object from that document into any other.

[1.1] Inkscape can read a large number of bitmap formats, including all of the major ones (PNG, JPG, TIFF, GIF, BMP, but it can't (yet) read the relatively new WebP even though it can export it). Choose **All Bitmaps** in the **Import** dialog to see only the files in supported bitmap formats. Past versions of Inkscape failed when trying to open bitmaps larger than several thousands pixels in one of the dimensions; now you won't have problems with images measuring tens of thousands of pixels (if your computer has enough RAM).

Whether opened or imported, what you end up with in your document is a *bitmap object* (Figure 18-1). In most aspects, this is a regular object that you can transform, duplicate, clone, apply filters to, and so on. In the status bar, it is described as *Image* with its pixel size, for example *640 × 480*. The pixel size of a bitmap is simply how many pixels it has horizontally and vertically; this is not the same as the size of the bitmap object on your canvas—you can scale it to any size in your SVG, but the intrinsic pixel size of the bitmap never changes.

Figure 18-1: A bitmap object in a document

NOTE *Just as you can open a bitmap file with* File ▸ Open*, you can* File ▸ Import *an SVG file into the current document (as a shortcut for opening it, selecting all its objects, and copy-pasting them into your document). Even better, both* Open *and* Import *can fetch files from the internet—just as well as they can open local files: simply paste a URL of the file (for example,* https://upload.wikimedia.org/wikipedia/commons/0/0d/Inkscape_Logo.svg*) into the* File name *field of the file dialog.*

18.2 Bitmap Import Options

Figure 18-2 shows the dialog you see when you open or import a bitmap into Inkscape.

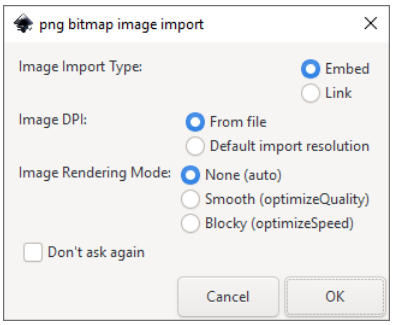


Figure 18-2: Importing a bitmap from a file

18.2.1 Linking vs. Embedding

The first choice you face with a bitmap is whether to *embed* it or *link* to it. By default, any bitmap you import into Inkscape is linked to the document—that means the actual content of the bitmap is always taken from the original bitmap file. The SVG document contains just a reference to that file—its filename and path relative to the location of the SVG file. The status bar description of a linked bitmap object reports its filename (*inkscape.ico* in Figure 18-1).

This way, the SVG file size is kept to a minimum, and multiple SVG documents can reuse the same bitmap file. This also means that any changes made to that linked bitmap file outside Inkscape will be reflected in the Inkscape document immediately. Moreover, you can run an external editor (such as GIMP or Photoshop) on your linked image from inside Inkscape by right-clicking the bitmap object and choosing Edit externally.

On the Imported Images page of Preferences (3.1.1), you can uncheck Automatically reload images to disable the updating of imported bitmaps. It is also where you can choose which Bitmap editor you want to use to edit them via Edit externally (browse to select an executable file).

The biggest disadvantage of linked bitmaps is in how easy it is to disrupt this link. If the bitmap file is deleted or moved to a different location relative to the SVG document, the result is not pretty, as Figure 18-3 demonstrates.

Figure 18-3: What Inkscape shows when it cannot find a linked bitmap file

This is a common problem when you send your art to someone but forget to include the linked images.

What matters for Inkscape is the relative location of the bitmap, because in the `xlink:href` attribute of the `svg:image` object, it stores the relative path from the SVG document location to the image file. For example, if the bitmap is in the *images* subfolder of the folder in which your SVG document resides, you can move that folder along with its *images* subfolder to a different location without a problem.

Inkscape can also store the *absolute path* to the image in the `sodipodi:absref` attribute; if that attribute is present, Inkscape will use it if the relative link in `xlink:href` fails. You can enable this behavior, disabled by default, with the Store absolute path for linked images checkbox on the Imported Images page of Preferences. Using `sodipodi:absref` will restore the images if you move the SVG document to another location on the same computer but leave the linked images behind; it won't help, however, if you move the images, or if you try to open the SVG on a different computer without its associated images.

To prevent any linking problems once and for all, *embed* your image into the SVG document by choosing the corresponding option during bitmap import (Figure 18-2). The status bar description for such an image looks like *Image*

64 × 64: embedded. An embedded image is stored right inside the SVG file, so it will never be lost. On the downside, this increases the file size of the SVG file (by about 1.4 times the file size of the bitmap file, which may be significant). Also, embedded images cannot be edited in an external bitmap editor and cannot be shared by multiple SVGs.

If you imported an image as linked but now want to embed it, use **Extensions ▸ Images ▸ Embed Images**. You can apply this either to the selected bitmap object (leaving all others as they are) or to all bitmap objects in the document, as shown in Figure 18-4.

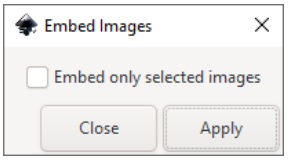


Figure 18-4: Extensions ▸ Images ▸ Embed Images embeds linked images into SVG.

With Extensions ▸ Images ▸ Extract Image, you can reverse this—extract the selected embedded image, or all embedded images in the document, into linked file(s). You will be asked for the path to save the extracted file, as Figure 18-5 demonstrates.

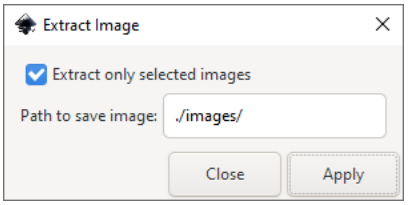


Figure 18-5: Extensions ▸ Images ▸ Extract Image converts embedded images to linked.

18.2.2 Size on Import

What size does an image have when you open or import it?

By its nature, a bitmap does not have a physical size measured in inches or centimeters. What it has is the *pixel size*—for example, 960 by 480 pixels. In addition to that, some bitmap formats specify the suggested *resolution*, also called the DPI (for *dots per inch*, a historic term meaning the same as pixels per inch). For example, if an image of 960 by 480 pixels has the resolution of 96 dpi, then it is supposed to be displayed in a space of 10 by 5 inches. For Inkscape, which is a vector application, this is no more than a suggestion—it treats the pixels of a bitmap as vector shapes that can be scaled to any size.

[1.1] Not all bitmap formats can specify DPI, and not all images have a meaningful DPI value. Still, by default Inkscape uses the image's DPI when it's available to calculate the *initial* size of the bitmap object on canvas. Otherwise, it uses the Default import resolution (on the Imported Images page of Preferences), which is 96 dpi. At this resolution, the size of each pixel square is exactly 1×1 px (A.6). You can force Inkscape to use the same resolution for all imported images, disregarding their own DPI, by turning on the Override file resolution checkbox.

[1.1] 18.2.3 Rendering Options

Bitmaps consist of pixels (1.1), which, to Inkscape, are little different from flat-color rectangles that get scaled, rotated, or skewed when you transform the image. If you zoom in close enough, you can even make out those individual pixels—but only barely, because by default Inkscape tries to smooth them out. If you dislike it, you can turn this smoothing off, when you import an image, by setting the Image Rendering Mode (Figure 18-2) to Blocky (optimizeSpeed). The default None (auto) and Smooth (optimizeQuality) are currently the same, and both smooth out pixels at high zoom levels, as shown in Figure 18-6.

Figure 18-6: Image rendering options

It makes sense to use Blocky, for example, when you work with pixel art where the position of each individual pixel matters. It is also preferable for large images that are not going to be zoomed into—for them, the jaggedness of individual pixels will never be seen, but the Blocky option results in slightly faster rendering.

> **IMAGE SCALING IN OLD INKSCAPE DOCUMENTS**
>
> In older versions of Inkscape, imported images did not get the `preserveAspectRatio` attribute. Unfortunately, by SVG rules, the lack of this attribute has the same effect as its default value, which forces an image to preserve its aspect ratio despite any transformations. If you're trying to stretch an image in an old document but it remains unstretched and just increases the margins, you need to fix such an image by adding `preserveAspectRatio="none"`. You can do this manually in the XML Editor, or use the Set Image Attributes extension to do this for multiple images (make sure Support non-uniform scaling is set).

If you have one or more images imported with wrong settings that you need to change, use the **Extensions ▸ Images ▸ Set Image Attributes** extension. Its Render images blocky option can apply to all images in the document or only to selected images.

18.3 Clipping and Masking

Usually, the first thing you'll want to do to an imported bitmap is *crop* it, removing the unnecessary margins and leaving only part of the image. Inkscape provides several ways to achieve this.

All techniques in this section are shown for bitmap objects—because that's what they are most often used for—but you can apply them to any kind of object, including groups or layers (which, as you remember, are just a special kind of groups).

18.3.1 Clipping

Inkscape allows any object to be *clipped* by a path, so that only part of the object inside that path will be visible. To determine which points are inside and which are outside, the same rules are used as for filling (12.1.2).

Starting from a bitmap object, draw a clipping path or shape over it using any convenient tool, such as the Rectangle or Pen. Then, select both the bitmap and the path/shape and choose Object ▸ Clip ▸ Set. The clipping path disappears (it now resides in `defs`, A.4), but the bottom object is now clipped by it. To edit the clipping path (via its nodes or shape handles) without unclipping, click the corresponding toggle in the Node tool's controls bar, as shown in Figure 18-7.

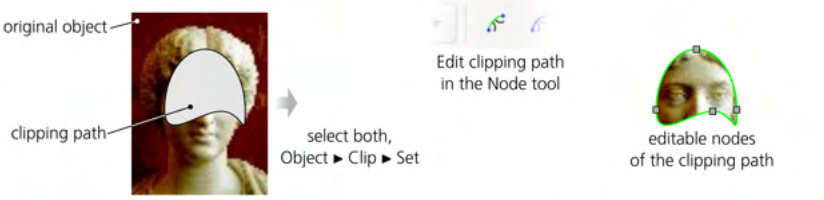


Figure 18-7: Applying clipping to a bitmap object

At any time, you can do Object ▸ Clip ▸ Release to remove the clipping and get the object and its clipping path as two separate objects again.

NOTE *The clipping path does not have to coincide with pixel boundaries—when zoomed in close enough, you can literally clip half a pixel.*

The clipping path can be a group of paths or shapes, not a single path or shape. It makes more practical sense, however, to have a group as the clip target, not as the clipping path. When you clip a group, you can always enter it (4.9.1) and work inside, adding more objects, moving them around, or perhaps replacing the original object you intended to clip with a different one—all without releasing the clip. That's why I recommend, even if you have a single bitmap you want to crop, to group it first with itself and then clip that group, not the bare bitmap.

18.3.2 Masking

Masking is similar to clipping. You select the object and the mask and choose Object ▸ Mask ▸ Set to mask it, or Object ▸ Mask ▸ Release to remove the mask. However, with clipping, an object is limited to the interior of the clipping path;

any properties of the clip, such as opacity, fill color, stroke, or blur, have no effect on clipping. Clipping is *binary*: at any point, the clipped object is either visible or not. Masking, on the other hand, is *gradual*—it may make an object *partially* transparent.

Here's the rule to remember: in a mask, *black* makes the masked object *transparent* (invisible), and *white* makes it *opaque* (visible). A 50 percent gray or 50-percent-opaque white makes the masked object semitransparent, but 50-percent-opaque black works just the same as fully opaque black (makes the object invisible). Any points not covered by the mask, or where the mask is 100 percent transparent, are not visible in the masked object. In other words, to make the object visible, the mask must have *visible lightness*: an opaque white mask gives 100 percent visibility, while anything darker or more transparent produces less than 100 percent visibility.

Obviously, masking is most useful with gradients or blurring. For example, you can blend a photo strip with the background using a linear gradient mask, or feather the edges of a photo cutout using a blurred mask (Figure 18-8).

Figure 18-8: Masking a bitmap object

The Node tool has another button for editing the mask of the selected object. However, this being the Node tool, that button only allows you to node-edit the single path or shape of the mask—you can't change its color, gradient, or blur; for this, you need to release the mask and set it again after changing.

Just as with a clipping path, both a mask and a masked object can be groups. It makes sense to use masking on a group, even if it's a single-object group at first, because you can enter that group and work inside it without removing the mask.

18.3.3 Bitmap as Pattern

If you want your imported bitmap to remain a rectangle and you only want to shave off some margins, you can turn it into a *pattern* (10.8) by pressing Alt-I. This does not change the visible display but converts a rectangular bitmap (it must not be rotated or skewed for this to work properly) into a rectangle object (11.2) with a pattern fill displaying the bitmap.

This method is convenient because you can edit the shape or path and its bitmap fill at the same time (Figure 18-9). If you resize the rectangle via its two corner handles—for example, with the Node or Rectangle tool (11.2.1)—it doesn't affect the fill, which means you can crop it by moving the rectangle handles inward. At the same time, you can use the pattern's three handles to

move, scale, and rotate the pattern, as described in 10.8.2. (Initially, the pattern handles coincide with the rectangle handles; drag the X-shaped handle in the top-left corner to separate them.)

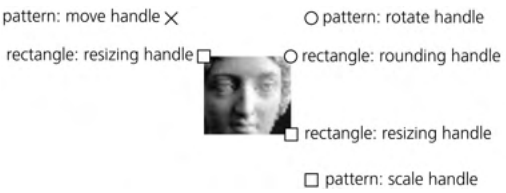


Figure 18-9: A bitmap as a pattern in a rectangle

After you convert your bitmap to a patterned rectangle, you can convert it to path with Shift-Ctrl-C and node-edit the result, or intersect (12.2) the rectangle with another path. Transforming, by default, affects both the shape/path and its pattern fill; see 6.11 for how to change that.

18.4 Retouching and Patching

Simple bitmap editing tasks, such as hiding defects or suppressing unnecessary details, are possible in Inkscape using its vector tools, without resorting to an external bitmap editor. While this approach is limited, it is often surprisingly useful and fast.

Imagine you need to remove a small blemish in a photo. Start by zooming in closely and drawing a calligraphic stroke over it. Then, switch to the Dropper tool and pick a color from the photo nearby to assign it to the newly created path.

In some cases, this may be all you really need—even such a primitive patch may blend well enough to become invisible once you zoom out. More likely, however, the edges of the patch will still be noticeable on at least one side. The next step is, therefore, a gradient: switch to the Gradient tool (10.1), draw a linear or elliptic gradient trying to match the dominant direction of color change in the background, and use the Dropper tool again (8.8) to pick colors for the gradient stops. If the gradient fails to make the patch completely blend into the background, see if a little blurring can help, as shown in Figure 18-10.

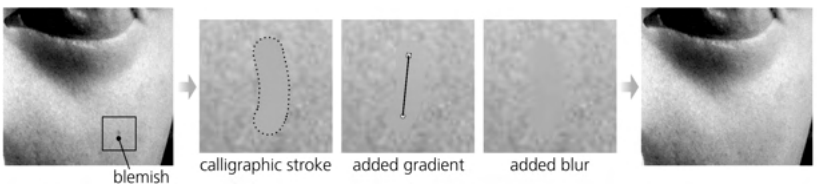


Figure 18-10: Covering a small blemish on the cheek with a vector patch path, made "invisible" by gradient and blur

When these simple methods aren't good enough, try the new Mesh Gradient tool (10.7) with automatic color picking for the mesh nodes (10.7.5) to create a more complex or extensive patch.

When you're done retouching, don't forget to group the bitmap object with all its vector patches so the group can be moved as a single object.

18.5 Tracing

For a vector editor, two crucial bitmap-related capabilities are converting a bitmap to vector objects (*tracing*) and vice versa (*bitmap export*). Inkscape offers rich and powerful tools for these conversions, which the rest of this chapter explores in detail.

18.5.1 Manual Tracing

One way of tracing a bitmap does not involve any tools other than those you already know. Just switch to the Pen tool (14.1.1), zoom in on your bitmap, and do a series of clicks around or along an area that you want to turn into a vector path (Figure 18-11). Use click-and-release in a sharp corner to create a cusp node; for smooth curved edges, make a series of short click-and-drags along the curved line. Vary the density of your clicks depending on how precisely you want to trace a specific area. Double-click, press Enter, or click the starting node to finish the path.

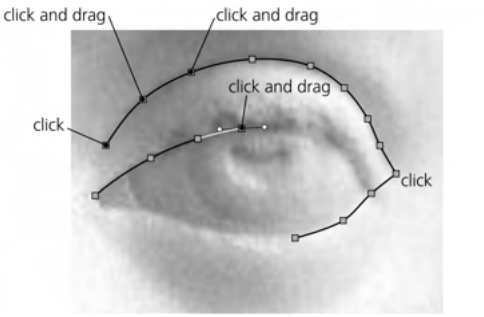


Figure 18-11: Tracing a bitmap manually using the Pen (in the default Bézier mode)

For extra smoothness, you can trace in Spiro or BSpline modes (14.1.4); in those modes, any click creates a smooth node, and for a cusp node, do a Shift-click. Also, in the BSpline mode, remember to click not on the curve itself but a little outward on its curved side, so that the smooth path is inscribed into the polygon you create with your clicks. If you're tracing art without any curves at all, use the Straight lines mode so that an accidental drag does not create a smooth node you don't need.

This technique may seem tedious and time-consuming at first, but once you get the hang of it, you will be able to trace complex art surprisingly quickly. Like any manual technique, its main advantage is the complete creative control—you decide what parts to trace and what to ignore, how to simplify complex shapes, where to diverge from the bitmap, and where to place each node. Depending on your skill, the result may look more satisfying than either an automatic trace or a completely manual drawing.

18.5.2 The Trace Bitmap Dialog

Inkscape's tool for automatic bitmap tracing, based on the stand-alone Potrace open source tracer (*http://potrace.sourceforge.net/*), is very powerful. With it, you can trace anything from a simple black-and-white logo that needs just a few nodes to a complex photo that produces dozens of colored paths with thousands of nodes.

The Trace Bitmap dialog (Shift-Alt-B, Figure 18-12) has two main areas: the options panel on the left and the preview panel on the right. What the preview panel shows is not the traced vector path (that might be time-consuming to create) but the bitmap as it will be fed to the tracer—with all the color reduction and filtering preprocessing as specified in the options panel. To update the preview after the options change, click **Update**. Note that the dialog always shows, and the tracer always traces, a complete bitmap even if it is clipped, masked, or otherwise obscured in the document.

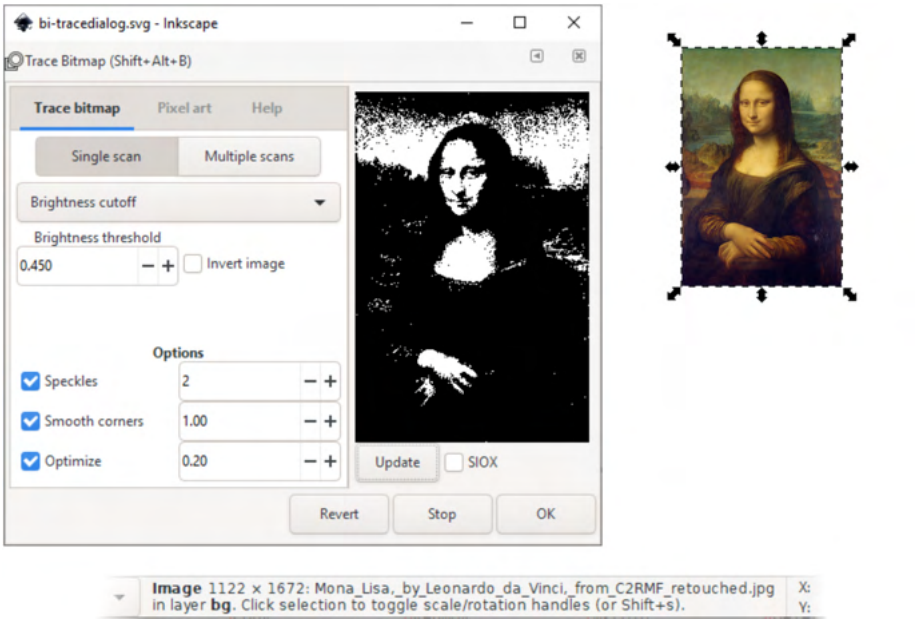


Figure 18-12: The Trace Bitmap dialog

To perform the actual trace of the selected bitmap object, click **OK**. For a large bitmap, this may be slow; watch the status bar for progress messages. You can interrupt tracing by clicking the Stop button; Revert resets the options to defaults.

The tracing tool has several modes of operation. These modes are divided into *single-scan modes* that create a single path from an image and *multiple-scan modes* that create multiple paths (grouped together).

18.5.2.1 Brightness Cutoff

Brightness cutoff is the simplest approach to tracing a path. The resulting path covers anything that is darker than the threshold you set. This trace path, while a single object, can consist of multiple nonoverlapping subpaths (12.1.1). This is the best tracing mode for simple monochromatic shapes such as logos, text, vignettes, and so on.

The Threshold is set as a fraction of the image's complete brightness range (Figure 18-13). For example, at 0.6, the trace path covers the areas that are darker than 60 percent of the image; the Invert checkbox inverts the threshold so the path will cover the brightest 40 percent of the image.

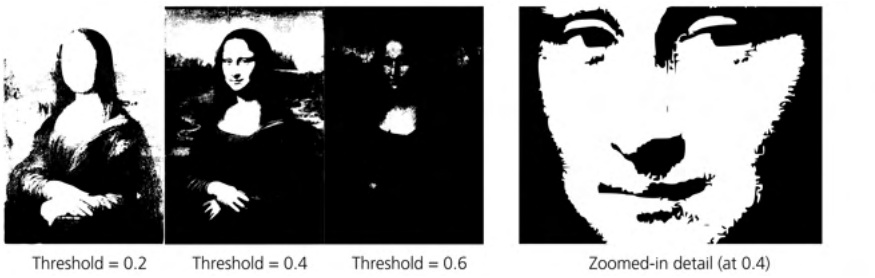


Figure 18-13: Brightness cutoff tracing with different threshold values

The Autotrace mode is another brightness-cutoff mode powered by a different tracing backend, Autotrace, instead of the Potrace backend other modes use. In this mode, you can't set the threshold, but the default might just work for you, as Figure 18-14 demonstrates.

Figure 18-14: Brightness cutoff tracing with different threshold values

18.5.2.2 Tracing Quality

Even if the bitmap you're tracing is itself a rendition of a vector path, the trace will never exactly reproduce that original path. Converting a vector shape into a bitmap always incurs loss of information, and Inkscape's tracer cannot restore this lost information other than by guessing. Generally it's pretty good at it, but there will be cases—especially when tracing low-resolution bitmaps or those containing text—where you will be disappointed by its failure to recognize features (arcs, straight lines, corners) that you can easily see in the bitmap, as shown in Figure 18-15.

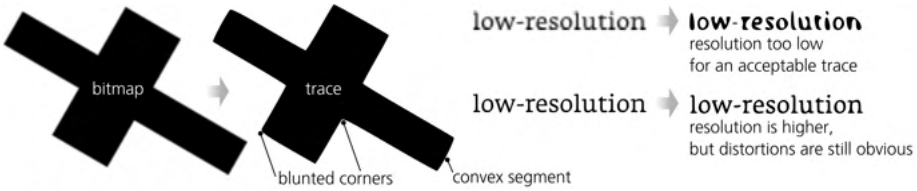


Figure 18-15: Some common quality problems with bitmap tracing

The best piece of advice in this situation is to get your bitmap at the highest possible *resolution*. It's very hard to get a decent trace from a bitmap whose crucial features are several pixels across; tracing a higher-resolution version of an image often makes a huge difference. Also, you can try to adjust the Threshold and experiment with the Options at the bottom of the dialog (these options apply to all modes, both single-scan and multiple-scan):

- The Speckles option removes any color blobs that are smaller than the specified number of pixels across. This suppresses small superfluous subpaths when tracing dirty or dithered bitmaps.
- Increasing the Smooth corners parameter makes the trace algorithm less inclined to recognize sharp corners in the image. This may be useful when tracing a naturally smooth shape from a highly pixelated, low-resolution bitmap where you don't want accidental pixel cusps to become sharp corners in the traced path. Conversely, lowering this parameter is appropriate when you are tracing geometric shapes without any curved lines. When Smooth corners is zero, the resulting path consists almost entirely of straight line segments with cusp nodes between them (but corners may still be chamfered).
- The Optimize paths parameter tries to reduce the number of nodes in the trace path, much like the Simplify command does (12.3). Raising this value decreases the number of nodes you get but also increases the chance of introducing visible distortions or losing the details of your shapes.

18.5.2.3 Other Single-Scan Modes

The Edge detection mode applies the edge detection filter to the bitmap before tracing it. As a result, the trace path will contain narrow strips that follow the color boundaries in the source bitmap. The lower the Threshold is, the more edges will be detected and traced.

The Color quantization mode first quantizes (divides) the image into the given number of areas (Colors), each with its own dominant color, much like when reducing a full-color image to a fixed palette in a bitmap editor. It then traces *every other* such area, which typically makes color gradients appear striped, as Figure 18-16 demonstrates.

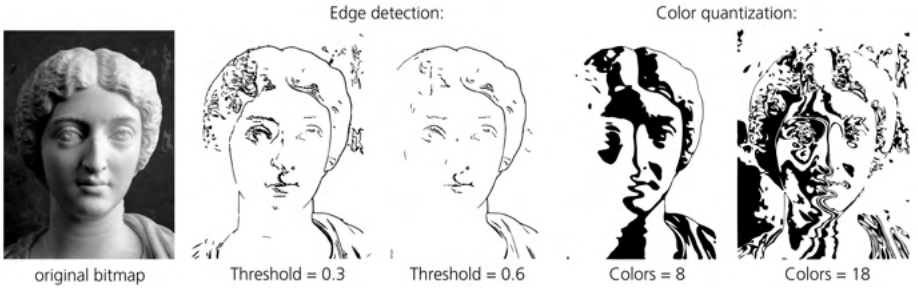


Figure 18-16: The Edge detection and Color quantization modes

Very visually interesting is the Centerline tracing option provided by Autotrace. It's a bit similar to Edge detection but creates a stroked path, not stroke-like strips. In the areas where there are no well-defined edges, this mode creates a charming geometric pattern with a predilection for diagonals, as shown in Figure 18-17.

Figure 18-17: The Centerline tracing (autotrace) mode

18.5.2.4 Multiple-Scan Modes

Like the single-scan Color quantization mode, each multiple-scan mode starts by quantizing the image into the given number of areas (the Scans parameter). It then traces each area separately, assigns an appropriate color or gray level to the trace path, and groups all the paths together. With enough scans, the result for a high-resolution bitmap may look pretty decent—faithfully reproducing the color gradients, blur, natural textures, and so on.

The Brightness steps mode is the best for grayscale images; it ignores any hue or saturation differences and groups pixels into areas based solely on their brightness (Figure 18-18). The Colors mode considers all aspects of the colors when performing quantization, which results in the most faithful reproduction of full-color images (see Figure 16 in the color insert). Finally, the Grays option works the same as Colors, except the resulting paths are painted with approximating shades of gray instead of the original colors.

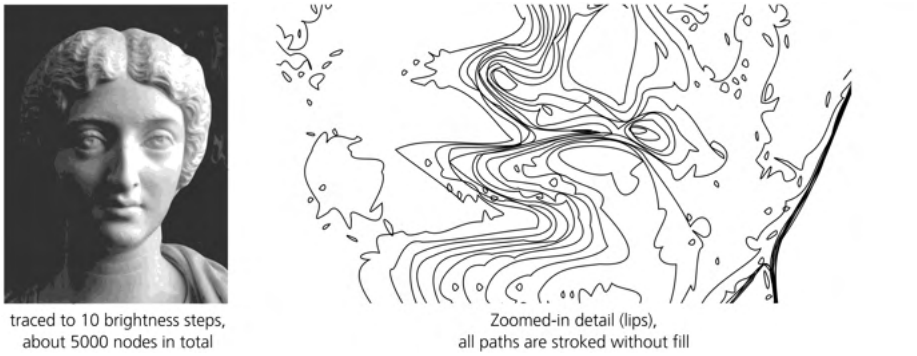


Figure 18-18: Multiple scans: 10 brightness steps

The Smooth option applies a certain amount of blur to the image before quantizing it; this may produce better results in complex photographic images. The Stack scans option is best kept on: it makes sure that each area's path covers not only that area but also all areas below it in z-order, which means there will be no gaps between the scans, as shown in Figure 18-19.

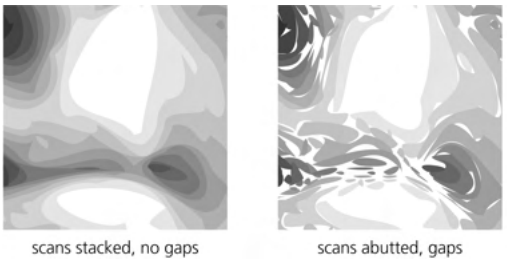


Figure 18-19: Stacking scans vs. abutting them

The Remove background option simply removes the bottommost scan path from the group. This is useful when you are tracing a photo of something on a flat-color background and want to have the vector representation of the object only without the background.

18.5.2.5 Object Extraction with SIOX

SIOX stands for "Simple Interactive Object eXtraction"; this is an algorithm that assists you in separating a foreground object from the background in an image. In Inkscape, it is a preprocessor applied to an image before it is traced (using any mode) when you turn on the SIOX checkbox next to the Update button.

The "Interactive" part of SIOX suggests, however, that it cannot do its magic fully automatically but needs some help. For SIOX to work, you must have two objects selected: the bitmap and, on top of it, a mask path that identifies the foreground object you're interested in. That mask path can be pretty crude, but it must cover the entire foreground object and some minimal amount of background around it. The result of a SIOX scan is rarely perfect (in Figure 18-20, bits of background still need to be cleared away around the cat), but it's a good starting point that you can get to very easily.

Figure 18-20: Using the SIOX option to extract a foreground object during tracing

[1.1] 18.5.2.6 Pixel Art Tracing

Inkscape has yet another tracing backend called *libdepixelize* designed for vectorizing low-resolution bitmaps where meaningful features can be as small as single pixels and there's little to no anti-aliasing. Such images, usually called *pixel art*, were the norm in computer games of the low-resolution era but, surprisingly, remain a popular subgenre of digital art even today.

Inkscape is not the best application for creating pixel art, but the Pixel art tab of its Trace Bitmap dialog makes it one of the best for reusing some old (or new) pixel art in your vector designs. This doesn't mean simple re-creating a bitmap's pixels as square paths; if you need that, just import your bitmap with the Blocky option (18.2.3) and use it as is without tracing. Instead, Inkscape's pixel art tracer attempts to infer and re-create the shapes the pixel art author had in mind—from their pixel approximations. As improbable as that sounds, the results are often very satisfying. For example, where two same-color pixels are diagonally adjacent, the pixel art tracer is smart enough to deduce they are a single feature and melt them together in the output, as demonstrated in Figure 18-21.

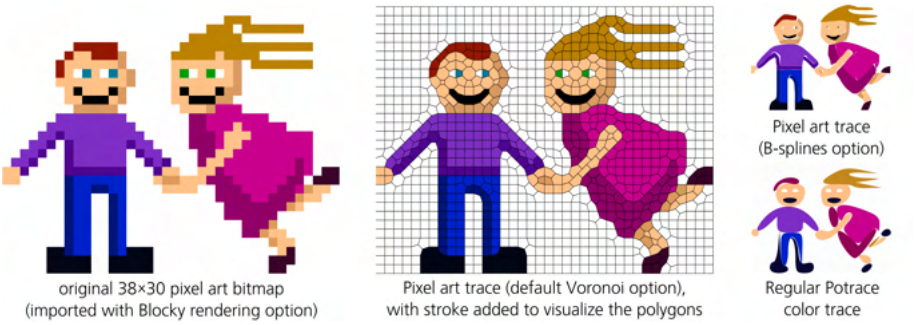


Figure 18-21: Using the pixel art tracer

The default Voronoi option of the pixel art tracer is usually the best; the grouping of polygons it creates is known as a *Voronoi tessellation* (named after a 19th-century Russian mathematician). The other output option, B-splines, tries to further smooth out the shapes of the Voronoi polygons, but the result is usually not as neat. For comparison, Figure 18-21 also shows what the regular Potrace color tracer would do to the same pixel art image.

The options in the Heuristics section are best left to their defaults. If you want to learn more about the algorithm used by the pixel art tracer and about what these options can give you, start with the tutorial at *https://inkscape.org/doc/tutorials/tracing-pixelart/tutorial-tracing-pixelart.html* written by the authors of this feature.

18.6 Bitmap Export

Nowadays, Inkscape's SVG files are often used on the web directly, thanks to web browsers' support for SVG. Still, ours remains a bitmap-dominated world, so the quality of rasterization and the bitmap export capabilities are crucial for a vector editor.

18.6.1 The Export PNG Image Dialog

Use the Export PNG Image dialog (Shift-Ctrl-E) to export your drawing or any part of it into a PNG file at any resolution. The dialog's areas, top to bottom, allow you to select:

- The canvas area you want to export (18.6.1.1)
- The pixel size of the bitmap (18.6.1.2)
- The filename of the export file and its format (18.6.1.3)
- Additional export options (18.6.1.4)
- Export format parameters (18.6.1.5)

18.6.1.1 Export Area

For the area you want to export, you have four main options at the top of the dialog: Page, Drawing, Selection, and Custom (Figure 18-22).

The Page button exports the page of your SVG document, as visualized by the frame on the canvas (2.3), but any objects beyond the edges of the page are excluded. The Drawing button exports the bounding box of the entire visible drawing (hidden layers or objects don't count), which can be smaller or larger than the page; the page frame is not visible in the exported bitmap. When the Selection button is pressed, the dialo'g exports the bounding box of the current selection (again, it can be inside or outside the page frame—this does not matter).

Alternatively, you can click Custom and type your own document coordinates for the top-left corner (x0, y0) as well as either the bottom-right corner (x1, y1) or the width and height of the export area. You can also choose the measurement unit for these values (the default is px, A.6).

The Export PNG Image dialog is not modal—that is, you can keep working on the canvas while it is open. Unless you choose Custom, the dialog responds to changing the selection by switching to the Selection mode and updating the coordinates to match the bounding box of the new selection. If nothing is selected and you are not in Custom, the dialog defaults to the Drawing mode.

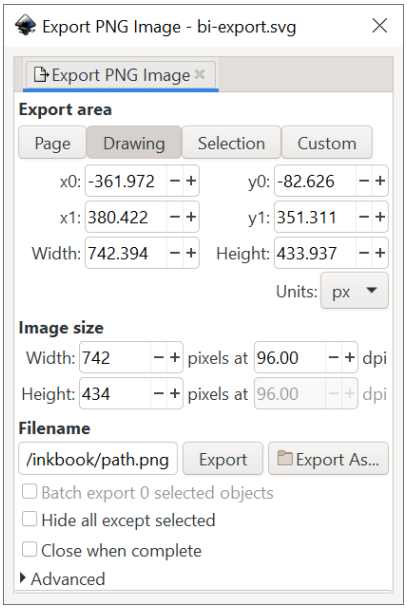


Figure 18-22: The Export PNG Image dialog

18.6.1.2 Image Size

For the Image size, type the width and height in pixels (here, it is bitmap pixels, not px units), or adjust the resolution value of dpi, which stands for *dots* (that is, pixels) *per inch*. The default resolution of 96 dpi results in one SVG px unit translating to one pixel of the rendered bitmap. The horizontal and vertical DPI values are always the same; changing any one of the three editable values (Width, Height, dpi) changes the other two to match.

NOTE *In versions of Inkscape older than 0.92, the default one-px-to-one-pixel resolution was 90 dpi, not 96.*

If your drawing contains some px-sized objects (such as strokes a whole number of px wide) and you want them to export precisely into the pixels of the bitmap, it may not be sufficient to choose the resolution of 96 dpi. For example, a 1×1 px square in SVG might fall on a boundary between pixels in an exported bitmap and thus would end up smeared into four adjacent pixels instead of one.

One way to fix this is by applying the Pixelize filter (17.4) to the exported objects. Another is to suppress anti-aliasing in PNG export options (18.6.1.5). Both those methods work, but you may find that your single-pixel features disappear or get snapped to a wrong pixel position in export. To fix this properly, you need to put in some more work. Enable the grid (7.2) and snap your objects to the default 1 px-sized grid (or use the Pixel Snap extension, 13.4.2.6). Make sure your export area is itself aligned to the grid—that is, it has integer x0 and y0 coordinates when measured in px. After that, exporting at 96 dpi will give you a perfectly crisp image with no unwanted anti-aliasing.

18.6.1.3 Export Filename and Format

For the Filename, you can type or paste a complete path to the file or click the Export As button to access your filesystem and choose a folder and a file. If you've already exported this object or the whole page, Inkscape will try to fill in the filename automatically for you (18.6.1.4). If the file exists, you will be asked if you want to overwrite it.

[1.1] For a long time, PNG was the only export format Inkscape supported. By default, Inkscape still exports to PNG; the PNG format preserves the maximum rendering quality Inkscape is capable of, including alpha transparency, anti-aliasing, and RGB color with 8 bits per channel. Since version 1.1, you can also export to some other formats: JPG (lossy), WebP (lossy or lossless), and TIFF (lossless), as well as optimized PNG (lossy or lossless). Choose one of these formats from the **Save as type** list in the **Export As** filename chooser, or simply append the corresponding extension to the filename (*.jpg*, *.webp*, or *.tiff*; you cannot select optimized PNG by extension).

18.6.1.4 Export Hints

When you're exporting a single selected object, the export filename and resolution are remembered and stored in that object's node in the SVG document (so you may need to save once you're done with the export). These values are then restored into the Export PNG Image dialog whenever you select this object to export again. Similarly, the filename and resolution are remembered when you're exporting the Page or Drawing.

These *export hints* are great time savers when you need to export multiple objects—for example, slices of a web page graphic— to multiple bitmap files. If you've done this once (and saved your document after that), just select your objects with the Export PNG Image dialog open and you will see their saved export filenames in the Filename field. For the same reason—to make it easier to export multiple objects one by one—the dialog stays open even after you click Export (although there's also a Close when complete checkbox under the Filename field).

> **EXPORT HINTS IN SVG**
>
> The export hints are stored in the `inkscape:export-filename`, `inkscape:export-xdpi`, and `inkscape:export-ydpi` extension attributes in the object's SVG element or (for Page or Drawing options) in the root `svg:svg` element.

You can speed up the process even more by selecting all objects you want to export and checking the Batch export all selected objects checkbox. Now, when you click Export, Inkscape will create one bitmap file per selected object. If an object has already been exported before—that is, has the filename and resolution hints saved—those values will be used for it; otherwise, a name for the bitmap file will be constructed from the object's ID (for example, *text2402.png*, see A.9), and the file will be placed in the folder where your SVG document was last saved.

Normally, even if you are exporting a selection, you are actually exporting an area—the bounding box of the selection—which may contain other visible objects. Checking the Hide all except selected checkbox ensures that the exported

bitmap will contain *only* the exported objects, while all others, even if they overlap the export area, will not be rendered.

[1.1] 18.6.1.5 Export Format Parameters

Inkscape allows you to adjust various parameters of export formats. There is no preview window, but if you want to try different parameter values and see the result at once, use this trick. Export your object and then import it back into Inkscape, placing it next to the object or area you're exporting. Now, you can export it over and over to the same file with different options, and Inkscape will reload the newly exported file each time, displaying the effect of your settings immediately.

18.6.1.5.1 PNG

In the dialog's Advanced section (collapsed by default), you can set some format options for the regular PNG export (not optimized PNG).

- Use interlacing creates interlaced PNGs which, when used on a web page, will display a lower-resolution version of the image before it is fully transmitted from the web server.
- Bit depth controls how color is represented in the output PNG. The default value is `RGBA_8`, which means RGB color, Alpha channel (transparency), 8 bits per channel; this is exactly how color is represented in Inkscape's renderer, so this option stores the program's rasterization output optimally. You can trim the output by dropping the alpha channel (`RGB_8`) or by switching to grayscale with (`GrayAlpha_8`) or without alpha (`Gray_8`) and with any number of bits per channel, down to 1 bit for a strictly black-and-white image (`Gray_1`). You can also use full-color representation with 16 bits per channel (`RGBA_16`), but since the renderer is still 8-bit, this will not actually increase the amount of information in the output (for example, if you're experiencing banding in gradients, this option will not reduce it).
- Compression sets the level of compression of the PNG file; the optimal value depends on the content of the file, so if you want to get the smallest possible PNG file, I recommend using external utilities like *pngcrush* on Inkscape's PNGs.
- pHYs dpi is the resolution value recorded in the PNG file; this value is rarely used, so there's no harm in leaving it at 0.
- Antialiasing is actually a parameter of Inkscape's renderer (called Cairo), not of the PNG file. This is where you can either turn off all anti-aliasing (the `CAIRO_ANTIALIASING_NONE` option) or try to improve on the default `CAIRO_ANTIALIASING_GOOD` by choosing `CAIRO_ANTIALIASING_BEST`.

18.6.1.5.2 Other Formats

For formats other than regular PNG, an Options dialog pops up after the progress bar is finished—that is, after Inkscape is done rendering the bitmap.

For JPG, you can set the Quality (1 to 100, default 90) and the Progressive option; when downloaded over a slow connection, a Progressive JPG file will show a low-resolution preview before the complete full-resolution image (this

only makes sense for large images). WebP has a Lossless option and adjustable parameters for Quality and Speed. TIFF has only Quality and Speed parameters.

For the optimized PNG format, you can turn on Interlaced (similar to JPG's Progressive option) and choose how many trials will be performed in an attempt to achieve the best compression for the image. On the Lossy Options tab of the Options dialog, you can enable various reductions (bit depth, color type, palette) that can make the image even smaller but also lower quality.

18.6.2 Exporting via the Command Line

All of the capabilities of the Export PNG Image dialog are also available when you run Inkscape from the command line. This way, Inkscape can be used as a GUI-less utility from scripts or programs to automate various SVG rendering jobs. For example, this is how you export an object with `id="text2402"` at 600 dpi:

```
$ inkscape document.svg --export-filename=img/text.png --export-id=text2402 \
    --export-dpi=600
```

Exporting to formats other than PNG via command line does not work as of Inkscape 1.1. For a complete list of Inkscape's command line switches, refer to Appendix C.

18.6.3 Icon Preview

While there's no preview pane in the Export PNG Image dialog, if you use Inkscape to create icons, one way to preview your work rendered to different icon sizes is via the Icon Preview dialog, which you can call from the View menu (Figure 18-23).

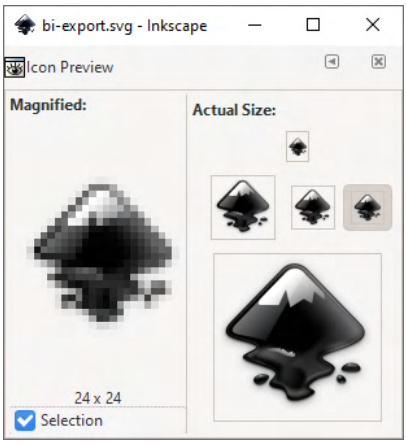


Figure 18-23: The Icon Preview dialog

This dialog contains previews of your document rasterized in a few typical icon sizes, from 16×16 to 128×128 pixels; in the left side of the dialog, one of these renderings is additionally displayed magnified so you can see how your vector objects translate to actual pixels. Click the **Selection** checkbox to switch the preview from showing the entire document to the current selection.

18.6.4 Make a Bitmap Copy

After you export something from your document, you can import (18.1) the bitmap file back to check how rasterization worked. If, however, you need that bitmap in your document and not as a separate file, you can use Inkscape's shortcut: the Edit ▸ Make a Bitmap Copy command.

This command exports the selected objects (without saving the file) and imports the result as an embedded bitmap back to the document, overlaying it on top of the selection. You can set the export's resolution on the Imported Images page of the Preferences; the default is 96 dpi, and for that value, the command will additionally snap the export area to the 1 px grid, making sure the pixels of the created bitmap align exactly with the boundaries of px squares, as shown in Figure 18-24.

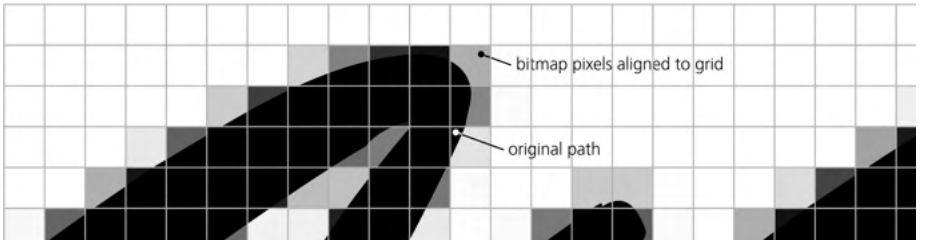


Figure 18-24: A 96-dpi bitmap copy of a path, aligned to the grid

You can use this command for a quick preview of how your art will rasterize (after which, the bitmap object can be deleted). It is also useful when you have some very slow-rendering filters (17.6) but do not want to work in the No Filters or Outline modes; in that case, just make a bitmap copy of the filtered object and hide the slow-rendering original.

18.7 Bitmap Filters and Extensions

You can apply Inkscape's filters (Chapter 17) to any kind of object, but they are perhaps most useful when you have a bitmap that is not editable with the standard vector tools. I especially recommend blend mode overlays (17.2) and the preset filters (17.4) in the Blurs, Color, and Image Effects submenus. You should use filters whenever possible because they are nondestructive and do not change the original bitmap file you imported.

Sometimes, however, you really do need to change the bitmap data itself. As mentioned above (18.2.1), with a linked image, you can just run your favorite bitmap editor on it, and you can even launch that editor from within Inkscape. This capability, however, is not available for embedded images. In those cases, extensions (Chapter 19) can help; a number of common bitmap processing algorithms have been implemented as extensions. Here are those that are most useful—and not (or not quite) achievable via filters—from the Extensions ▸ Raster submenu (Figure 18-25):

- Adaptive Threshold shifts each pixel to extreme values in each of the RGB channels, depending on whether that pixel is above or below a threshold. That threshold is calculated from that pixel's local neighborhood (whose size is defined by the Width and Height parameters). This is a valuable transformation, because it is similar to the way humans perceive images—we judge a point "light" or "dark" relative to its local area, not relative to the average lightness of the entire image.
- Add Noise gives you a choice of various types of noise to overlay on your image; most types look more or less like scattered sand.
- Crop is a way to reduce the size of an SVG with an embedded image—unlike clipping, it actually removes part of the bitmap data from the edges (you can specify how much to crop on each side).
- Despeckle, Enhance, and Reduce Noise implement various noise reduction algorithms.
- Dither randomly scatters pixels in a bitmap, with Amount specifying the radius of scattering.
- Equalize applies histogram equalization to the image.
- HSB Adjust adjusts the hue (in the range of –360 to 360), saturation (–200 to 200), and brightness (–200 to 200).
- Implode, Swirl, and Wave smoothly distort an image more or less as their names suggest. Wave uses a horizontal sine wave with the given amplitude and wavelength.
- Level blackens pixels that are darker than the Black Point, whitens pixels brighter than the White Point, and scales those that fall within this range to the full color range. When the Black Point is greater than 0 or the White Point is less than 100, the extension increases the contrast of the image. Gamma Correction specifies additional brightness correction (1 means no change).
- Level (with Channel) is the same as Level but for a single channel only.
- Median paints each pixel with the median color of its circular neighborhood; the result is somewhat similar to blurring, but Median preserves sharp boundaries between colors that are too different.
- Normalize increases contrast by expanding the color range of pixels to the full range of color (for example, if the image has no reds, all colors will be tinted toward red to compensate for that).
- Oil Painting is similar to Median but additionally melts similar adjacent colors in a way that's reminiscent of paint strokes.
- Resample changes the pixel size of the bitmap without scaling it. Resampling *up* does not change the appearance of an image, but it may be useful if you plan to apply some other effect and want it to work with higher resolution. Resampling *down* makes the image lose detail without changing its dimensions in the document; like Crop, this reduces the file size of an SVG document with an embedded bitmap.
- Sharpen and Unsharp Mask are the classic algorithms for sharpening an image.

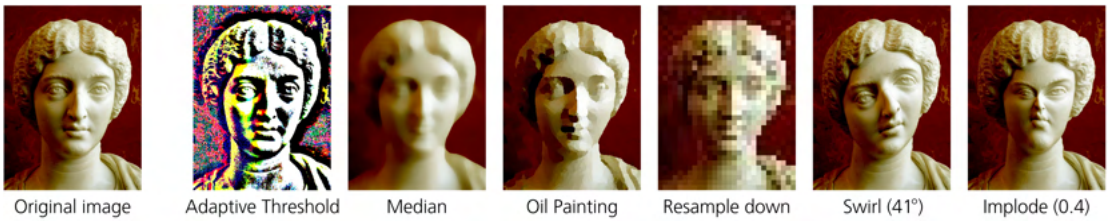


Figure 18-25: Examples of extension effects from Extensions ▸ Raster

Unlike filters, extensions make permanent changes to the bitmaps they process. You can undo an extension but you cannot, for example, readjust its parameters after it is applied. Also, unlike filters, these extensions can work *only* on bitmap objects—you cannot, for example, apply a bitmap effect to a clone of a bitmap. If the bitmap is linked, these effects embed it and work on the embedded copy, not on the external linked original. Refer to 19.1 for general tips on working with extension effects.

18.8 Color Management

Color management aims to ensure that the colors in your artwork are correctly translated between different output devices, typically from screen to print. The ranges of colors that can be reproduced are different for different devices, and some color distortions are unavoidable. Color management allows you to preview, control, and thereby minimize such distortions.

Inkscape has never claimed to be the best in class when it comes to color management. You can use screen-proofing to preview output colors, but you can't, using Inkscape alone, prepare a color-separated file for print with a color profile embedded. However, in most cases, you can get the result you need by employing some additional software, as described in this section.

On the other hand, in the years since the first edition of this book, the importance of color management has certainly diminished. These days, a lot more graphics go from screen to screen than from screen to paper. Also, consumer-grade computer screens and desktop color printers are now much more uniform in quality—and most desktop printers will accept and print the same RGB data as is displayed on your screen, performing their own color conversion without you needing to do any extra work.

It's only when you intend to print your design on high-end commercial printers, usually by sending it to a print service provider, that you will need to look into Inkscape's color management facilities. Even then, some providers can perform color management on your documents for you if you ask them to. If you have sufficient control over this process (for example, can review the print proofs), this is usually the best option, because the print service staff knows what works best with their equipment.

18.8.1 ICC Color Profiles

An ICC color profile is a file that describes the color capabilities of an output device. If you want to prepare your document for outputting on a specific device, you must first obtain the ICC profile that exactly corresponds to this device and the output media (for example, the paper type used for printing). Sometimes, you can find an appropriate profile file on the internet (for example, on the website of the printer hardware's manufacturer), but typically, you would request it from the print service provider you will be using.

Once you get the destination profile, you need to install it into your operating system. On Windows 10, it's as easy as right-clicking an *.icc* file and selecting Install Profile. On Linux, you need to have *gnome-color-manager* or *colord-kde* installed; then, double-click the *.icc* file and click Import.

18.8.2 Screen Proofing

Since the color range of a typical printer is narrower than that of a computer display, Inkscape can proof (preview) the print output on your screen by emulating the printer colors. This is called *screen proofing* or *soft proofing*. For this, you need to have two ICC color profiles: one for the printer you're going to use and another for your screen.

Ideally, you should have your display *calibrated* using a special hardware device called *colorimeter*; this calibration creates a custom ICC profile of your display. Unless your quality requirements are truly demanding, however, you can probably make do with a generic RGB profile, such as the one that comes installed with Windows.

In **Preferences**, go to the **Input/Output ▸ Color management** page. Choose the display profile (if you have more than one) in the **Display adjustment** section. Then, in the **Proofing** section, check **Simulate output on screen** and choose the **Device profile** of your target device (that is, the printer on which you're planning to print your artwork). After that is set up, toggle the color-managed view via **View ▸ Color-Managed View** or the little toggle button in between the vertical and horizontal scroll bars in Inkscape's editing window.

For both screen and target device profiles, you can also choose Device rendering intent. The default Perceptual is the best choice in most cases; if you want the output to look as color-rich as possible (for example, when printing simple business graphics), try Saturation.

If a screen color is "out of gamut"—that is, cannot be rendered on the output device at all—you can make it immediately visible by designating an out-of-gamut marker color. For example, if your design has no reds, check **Mark out of gamut colors** and choose red for **Out of gamut warning color**. Then, wherever you see red in your drawing, you'll know you need to change the actual color of that object (which is shown in the status bar or in the Fill and Stroke dialog—the red mark is only in the drawing) if you want it to print without gross distortions.

18.8.3 Separating and Embedding

Screen proofing is helpful, but it may not be enough—you may be required to produce a file already converted to the target color system. Such files are often called *color-separated* because they contain the separate color channels (usually CMYK, 8.4.2) corresponding to the inks of the output device. Such a file may also have the target color profile embedded into it. The most commonly used formats are PDF (vector) and TIFF (bitmap); both can contain color-separated data and embed ICC profiles.

Although it can export PDF and PNG, Inkscape cannot do color separation or profile embedding. You need some other software to do the job for you, such as Adobe Photoshop (for TIFF) or Illustrator (for PDF); both can import Inkscape's SVG format directly.

You can also use open source software. The Scribus page layout program (*http://scribus.net/*) will import SVG and create color-separated PDFs; starting from Inkscape 1.0.1, you can set up PDF export via Scribus from Inkscape's Save dialog (see the Inkscape 1.0.1 Release Notes for details). The Cyan plug-in for GIMP (*https://github.com/rodlie/cyan/*) will take Inkscape-exported PNG bitmap and convert it into a color-separated TIFF with screen proofing for complete control.

19

EXTENSIONS

Most long-lived software projects owe a large part of their success to extensibility. If you manage to attract and keep outside developers with their unique needs, perspectives, and approaches, your software product has acquired good insurance against oblivion. Inkscape's ecosystem of extensions is not very large, but it is an important part of the program's impact. Almost 200 extensions ship with Inkscape 1.0, and you can find many more online.

In this chapter, I first look at extensions from the user perspective. I explain the extensions UI (19.1) and describe, with examples, the notable extensions included with Inkscape (19.2). After that, I put on my developer's hat; I explain the architecture of Inkscape extensions (19.3) and guide you through a simple example of an extension created from scratch (19.4). As Inkscape is an open source application, everyone can (and is encouraged to) contribute to it; writing an extension may be the easiest way not only to solve your unique problem but also make your solution useful to others.

19.1 Working with Extensions

An *extension* is a piece of code you can run to perform some actions on the selected objects in an Inkscape document. Within Inkscape, you can access most extensions via commands in the Extensions submenus. Unlike path effects (Chapter 13) or filters (Chapter 17), an extension is a one-off operation; it is typically destructive and changes objects without preserving the originals. The only way to reverse an extension run is via Edit ▸ Undo.

A typical extension has a number of parameters that you set in its dialog before running the extension. The dialog may have more than one tab, and some of the most developed ones have an About or Help tab that describes the extension and how to use it, as shown in Figure 19-1.

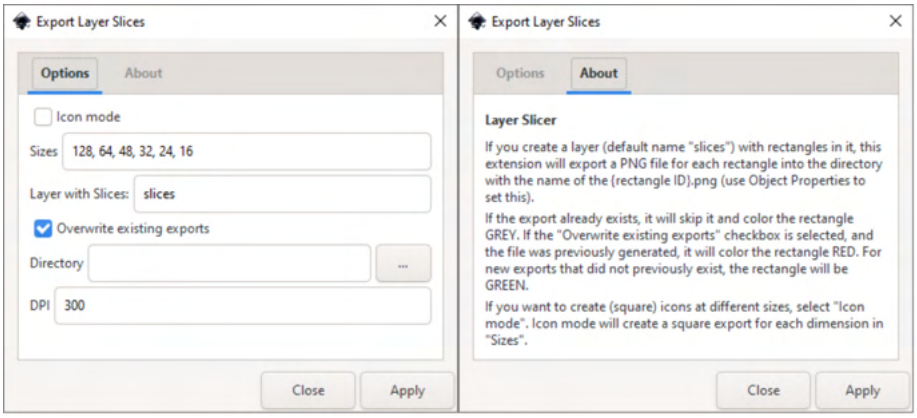


Figure 19-1: The Export Layer Slices extension: the two tabs of the parameters dialog

Most extension dialogs have a Live preview checkbox; when checked, it shows your document as it would look after the extension is applied with the current parameters. This way, you can experiment with parameter values without having to undo and run the extension again. To apply the parameters as final, click **Apply**; to cancel without running the extension, click **Close**.

When the Live preview is off, the extension dialog is not modal, so you can pan the canvas and select different objects; as soon as you turn Live preview on, the dialog locks the rest of Inkscape so you can only change the parameters in the dialog and see their effect. For some extensions, updating can be slow, especially when your document is big and complex.

The Previous Extension command in the Extensions menu re-runs the last extension command without showing the parameters dialog. The Previous Extension Settings command opens last extension's parameters dialog from which you can run it again.

19.2 A Guide to Inkscape Extensions

Many extensions have already been mentioned or described throughout this book. In some (but far from all) cases, an extension is an alternative to some other, more convenient or integrated method to achieve the same result; when

an extension duplicates functionality available in the core of the program, it's usually because the extension was created first. Still, many extensions are unique and indispensable tools in Inkscape's toolbox. Let's go through the Extensions menu to review what is available:

- The Arrange submenu has Restack for z-order manipulations (4.4) and Deep Ungroup for ungrouping multiple levels of grouping (4.8.1).
- Color extensions change the colors of selected vector objects (but don't work on bitmaps), as detailed in 8.10.1.
- Document has two extensions that convert between the old Inkscape SVG documents that assume 90 dpi resolution and the new ones that are based on 96 dpi (see the note at the end of 4.2 for details).
- Export extensions implement various ways to export a document, such as sending it to a plotter device (Plot) or exporting multiple slices of a web page mockup (Export Layer Slices, which is intended as an improvement on the standard batch export functionality in the Export PNG Image dialog, 18.6.1.4).
- Gcodetools is a collection of extensions to prepare and export your document to the G-code language that various computerized machine tools understand, so that you can, for example, cut or engrave a metallic badge out of your Inkscape design.
- The Generate from Path, Modify Path, and Visualize Path submenus provide a number of path-processing extensions described in the chapter on path effects (13.4).
- The Images submenu contains utilities for embedding and extracting images (18.2.1) as well as for changing image attributes (18.2.3).
- JessyInk extensions insert Javascript snippets into a document that turn it into an interactive presentation when viewed in a browser. With these snippets, you will be able to scroll the pages, make objects respond to clicks, use transitions, and even insert videos.
- Raster (bitmap-processing) extensions were reviewed in 18.7.
- Stylesheet ▸ Merge Styles into CSS is the tool to use if you have many objects with the same style and want to store that style in a single place so you can update all those objects at once. Recall that Inkscape supports selectors that apply a shared style to multiple objects (8.1); this extension creates such a selector from the selected objects and then switches them from their inline style properties to that selector.
- Text extensions were described in the chapter on text (15.7).
- Typography is a collection of extensions that aid in creating SVG fonts (15.8).
- Web extensions once again remind you that Inkscape's documents are usable as web pages because modern web browsers support SVG (see also JessyInk above). The Web ▸ JavaScript ▸ Set Attributes extension allows you to change an object's style property in response to a given event—typically a click of some other element. The Web ▸ JavaScript ▸ Transmit Attributes is similar but copies the clicked object's property to another object. You need to select both objects, the click target first and the property-changing object second, before calling these extensions.

19.2.1 The Render Submenu

The Render submenu contains extensions that generate entirely new objects—not based on anything in your document. Most of them haven't been mentioned, so here's a list of the notable ones:

3D Polyhedron
: Creates a projection of any of a number of three-dimensional shapes, from a cube to a great stellated dodecahedron; you specify the spatial angle and the styling of the shape's faces.

Barcode
: Extensions in this subsubmenu create a Classic linear barcode (supported formats include EAN8, EAN13, Code39, Code128, and others), a two-dimensional Datamatrix barcode, or a square QR Code with the given text.

Calendar
: In case you ever need it, this extension fills your page with a fully formatted calendar for a given year, with many layout, styling, and localization options.

Draw From Triangle, Triangle
: Similar to the path effects for geometric constructions (13.3.15), Draw From Triangle draws any of a number of geometric shapes (such as the circumcircle or orthocenter) based on a triangle defined by the first three nodes of a selected path. The Triangle extension creates a triangle from any unambiguous combination of sides and angles.

Function Plotter, Parametric Curves
: These two extensions draw function graphs. The Function Plotter creates a graph of a single function (for example, altitude against time of a moving body), whereas Parametric Curves draws a curve where both X and Y coordinates vary as functions of an independent *parameter* variable conventionally called t (for example, X and Y may represent coordinates of a body as they change with time).

  Before calling one of these extensions, draw or select a rectangle that will define the scale of the graph, and use the numeric parameters to define the ranges of the X and Y coordinates. The Multiply X range by 2*pi checkbox is convenient for trigonometric functions; with it, for example, the range of 0 to 2 is treated as 0 to $2 \times \pi$. In the Function Plotter, the Use polar coordinates checkbox transforms the graph into a circle whose center is in the center of the selected rectangle, with the X range mapped to the angle and the Y to the radius.

  The Samples parameter sets the total number of times the function(s) will be sampled; since each sampling produces a node, this is also the total number of nodes in the generated path. The higher this number, the more precise the graph. The optimum number of samples depends on the nature of your function; for example, a periodic function with n periods in the X range would need at least several samples per period to reproduce with any fidelity.

The function or functions are written in the Python programming language. You can use a number of built-in mathematical functions such as `sin(x)`, `log(x)`, or `sqrt(x)`; refer to the Functions tab for a full list.

In the Function Plotter (Figure 19-2), you can give the *first derivative* of the graphed function; the value of the derivative determines the angle of the Bézier handles at each node. Either ask the extension to Calculate first derivative numerically or uncheck that checkbox and provide the derivative function analytically, using the same Python syntax and built-in functions—for example, the first derivative of `sin(x)` is `cos(x)`.

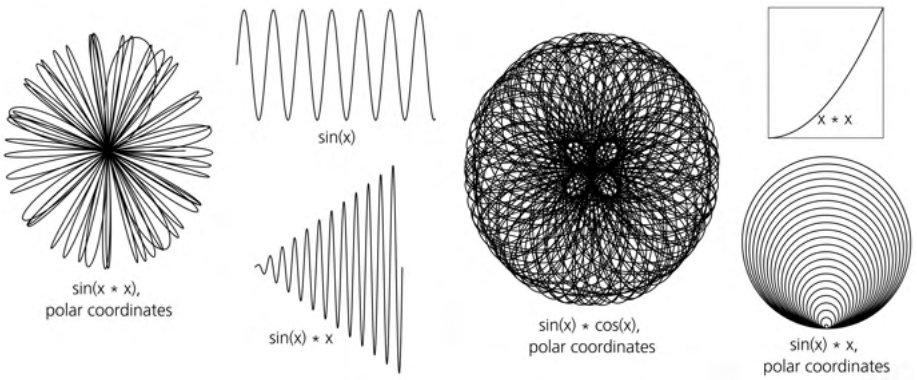


Figure 19-2: Function Plotter examples

Gear

These two extensions create a circular gear or rack gear, with a given number of teeth, pitch, and contact angle.

Grids

This family of extensions is for creating grids of lines—rectangular (Cartesian, such as the grid in Figure 13-60 on page 290), isometric, or polar. The options for the number of lines and their spacing are self-explanatory; major and minor lines use different stroke width. Lines can be spaced logarithmically. Each line is a separate path object.

Guides Creator

This extension divides your page into any number of rows and columns by adding guidelines (7.1).

L-system

This extension implements *Lindenmeyer systems*—a simple graphic language with recursion that can produce complex sequential or tree-like structures (Figure 19-3).

Programs in this language are built out of simple commands like "draw one step forward" or "turn left." You start with an Axiom and apply the substitution Rules to it recursively, limited by the Order parameter. The rest of the parameters determine the length and angles of the lines produced by the primitive commands; you can also randomize these values to render a

more natural result. For a complete list of the commands recognized in the axiom and rules, refer to the Help tab.

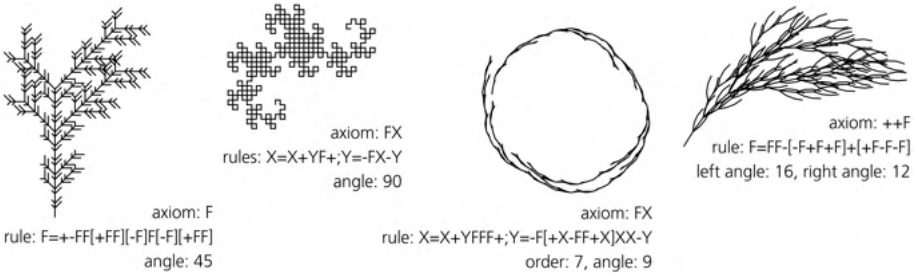


Figure 19-3: L-system examples

Mathematics ▸ LaTeX
: This extension runs `pdflatex` (which needs to be installed on your computer) to convert a LaTeX string into a vector formula image inserted into your document.

NiceCharts
: This extension can create bar charts, stacked bar charts, or pie charts based on data from a file or a string. Parameters determine sizes, spacings, and styling of the chart's elements.

Random Tree
: Similar to the L-system but more primitive, this extension draws a random branching tree where the first segment of the trunk is Initial size long and each subsequent branch is progressively shorter, until the Minimum size is reached and the drawing terminates. The bigger the difference between the initial and minimum sizes, the more complex the tree.

Spirograph
: A spirograph is a toy where a small circle, holding the pen, rolls along the inner edge of a larger circular hole. This extension is an implementation of the same idea that you can use to produce a huge variety of smooth centrally symmetric curves.

Wireframe Sphere
: This is the extension to use whenever you need a geometrically correct globe of parallels and meridians.

19.3 Extensions Architecture

The rest of this chapter is more technical. Read on if you're interested in learning how Inkscape's extensions are implemented, why you might want to make one of your own, and how to proceed if you do. Familiarity with the basics of programming in general and the Python language in particular is a plus.

Figure 19-4 gives an overview of Inkscape's extension architecture.

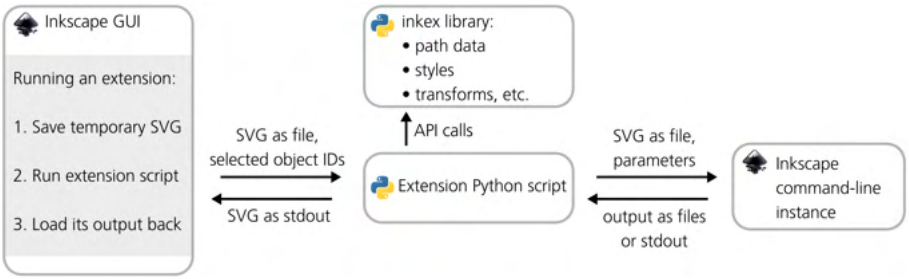


Figure 19-4: Inkscape extensions: under the hood

Unlike, for example, Adobe products, Inkscape does not have "first-class" plug-ins that would load into the main program's memory space and have the same level of access to the document's tree and the program's API as Inkscape itself. Instead, an Inkscape extension is just a script written in Python that Inkscape runs on a (temporary) saved copy of the current document—and then loads the script's output back, replacing the document with it and restoring object selection if possible. Inkscape's extension-running machinery compresses all this saving, processing, and loading back into a single undoable action that is transparent to the user. What's more, when you turn on the Live preview checkbox in an extension's dialog, every change of the parameters causes another save-process-load cycle—which is automatically undone if you change a parameter again or close the dialog without clicking Apply.

It is the extension script's responsibility to not touch anything in the document except what it is supposed to touch. Along with the reference to the saved temporary copy of the document, it is passed the list of currently selected objects as well as, of course, the parameters the user has set in the extension's dialog. The script must load and parse the SVG file, find the selected objects in it, do its thing, and output the changed document back as serialized SVG (via the standard output).

If this whole approach strikes you as clumsy and inefficient, that's because it is. However, such a radical decoupling of extension code from the core of Inkscape has its advantages, too. The biggest advantage is the low barrier of entry for new developers. You don't need to know much about Inkscape itself in order to start; all you need is basic familiarity with SVG and an understanding of how to deal with the XML tree of the document in Python. Existing Inkscape extensions are very easy to study, modify, or fork for the same reason. Another advantage is that any extension is easy to reuse as a stand-alone script that you can run not only from Inkscape, but also in all kinds of automated processing scenarios.

While you don't have any access to Inkscape's internals when you're in an extension, that doesn't mean you're completely on your own. Inkscape ships with a Python library called `inkex` (for INKscape EXtensions) that extension scripts are supposed to use. It has facilities for a lot of tedious and complex tasks that you don't need to do in your code, such as parsing the path data (the `d` attribute), parsing style strings and calculating an object's effective style (taking inheritance and CSS selectors into account), manipulating transform matrices, and so on. You can write much more powerful algorithms with `inkex` taking care of the low-level drudgery for you.

Sometimes, however, even `inkex` is not enough. You may need something done that only Inkscape itself can do—for example, create one or more PNG exports from the source document. For this, you need to run Inkscape itself—a *second copy* of Inkscape called, possibly repeatedly, via the command line. In older versions, even getting the correct bounding box of an object required a command line Inkscape call (using the `--query` parameters, C.5); as of this writing, `inkex` is capable of providing a correct bounding box for a path object, but it still can't do so for text (because of the complexity of correctly rendering arbitrary fonts and text layouts).

19.4 Creating an Extension

The remainder of this chapter walks you through creating an extremely simple—but still rudimentarily useful—Inkscape extension from scratch. Hopefully, seeing how easy and natural that is, you will feel encouraged to automate your own tedious and repeated manual tasks—and perhaps, eventually, share your solution so that others may benefit from it too.

The example extension, called *Make Initial*, creates a simple initial in a text object by taking its first character and increasing its font size. It has a single parameter: the size of the initial letter, given as percent of the font size of the rest of the text.

19.4.1 The .inx File

Before getting to the Python code, we need to create an *.inx* file (from INkscape eXtension) for the extension. This is an XML file that describes, to Inkscape, what this extension is, how to run it, and how to present it to the user. Here is the *makeinitial.inx* file for the Make Initial extension:

```
<?xml version="1.0" encoding="UTF-8"?>
<inkscape-extension xmlns="http://www.inkscape.org/namespace/inkscape/extension">
  <name>Make Initial</name>
  <id>com.kirsanov.text.makeinitial</id>
  <label>Enlarge the first character of each selected text object.</label>
  <param name="initialsize" type="int" min="0" max="1000" gui-text="Size of initial, percent">200</param>
  <effect>
    <object-type>text</object-type>
    <effects-menu>
      <submenu name="Text"/>
    </effects-menu>
  </effect>
  <script>
    <command location="inx" interpreter="python">makeinitial.py</command>
  </script>
</inkscape-extension>
```

The root element, `inkscape-extension`, is just a container. Note the obligatory `xmlns` attribute that sets the file's namespace, without which Inkscape won't recognize it. The `name` is the name of the extension as it will appear in the menu; `label` is the intro text to be displayed in the extension's dialog. The `id` can be any string so long as it's unique for your extension.

The single `param` element (there may be any number of them) describes the extension's parameter. Called `initialsize`, it is an integer (as this is a percentage, we don't need floating-point precision) in the range from 0 to 1000. The `gui-text` attribute provides the label it will have in the dialog, and the content of the element (200) is the initial value (after the first use, Inkscape remembers and restores the value you last used for all parameters).

The `effect` element identifies this one as an *effect extension*; other extension types are `input`, `output`, and `template` (see 19.4.2 for details). Inside, the `object-type` identifies what kind of selected objects this extension accepts; possible values are `all`, `path`, or `text`. The `effects-menu` element places this extension inside the Text submenu of the Extensions menu.

Finally, the `script` element tells Inkscape how to run this extension. In the `command` element, the `location="inx"` instructs it to look for the executable files in the same folder as the *.inx* file, and the `interpreter` attribute identifies Python as the interpreter to use (the version included with Inkscape 1.1 is Python 3.8.9). The content of that element (*makeinitial.py*) refers to the extension's main code file.

19.4.2 The inkex Base Classes

Gone are the days when Python was a quick-and-dirty scripting language with little structure. You now can—and are encouraged to—write fully object-oriented Python code. Inkscape's `inkex` library is as good an example as any: the first thing you do after `import inkex` is create a class that will hold your extension's code. Your class should extend one of the *base extension classes* of `inkex` so that it inherits all of the utility methods it needs for loading the SVG document, parsing the parameters, and outputting the changed document back.

The `inkex` library includes several base extension classes designed for various kinds of extensions, from general to specialized. Choose one that best fits your case.

- `EffectExtension` is the most general type of extension that, per Figure 19-4, takes the input document and the list of selected objects, does something to them, and returns the changed document back. You need to implement the `effect` method that does the actual processing. This is the base class we will use for our example.
- `GenerateExtension` is for an extension that doesn't care about the source document or its selection; all it does is generate some new object(s) to be added to it. The base class handles adding the generated objects to the document, placing them by default in the center of view. You need to implement the `generate` method.
- `InputExtension` is an extension that is not available in the Extensions menu; instead, it adds a new item to the list of file formats in Open and Import dialogs. To support a new file format through an input extension, you need to implement the `load` method that reads an input file from a stream and, optionally, the `effect` method that transforms the result. Needless to say, you have to return valid SVG.

- `OutputExtension`, similarly, is for output extensions that manifest themselves as file formats in the Save and Save As dialogs. You need to implement an optional `effect` method that transforms the document before saving and the obligatory `save` that performs the actual output to a stream.
- `CallExtension` is for an extension designed to be a simple interface between Inkscape and some external SVG-processing application. You need to implement the `call` method that calls your application on an input file and produces an output file. For such tasks, this base class is more efficient than the generic `EffectExtension` because it does not even parse the SVG in the Python code.
- `TemplateExtension` is an extension that provides a new document template. Like input and output extensions, it's not listed in the Extensions menu; instead, it adds an item to the list of templates in the New from Template dialog (Figure 3-3). You can implement the `get_template` method that returns the template's ready-to-use SVG code (for example, by reading it from a file). Alternatively, you can implement the `get_size` method (returns the template's size) and/or `set_namedview` method (creates the `sodipodi:namedview` element where you set up guides, page borders, zoom, and so on).
- `ColorExtension` is a specialized effect extension for adjusting colors in selected objects. All you need to do is implement the `modify_color` method that takes an input color and returns the output color; optionally, you can add `modify_opacity` that does the same for opacity. The base class does all the rest (such as parsing styles, tracking down clones, dealing with gradients, and so on).
- `TextExtension` is a specialized effect extension for editing a document's text content. All you need to do is implement the `process_chardata` method that takes an input text string and returns its edited version.

In addition to the obligatory methods for each base class (listed above), any extension class may also need to implement the `add_arguments` method if your extension has some user-adjustable parameters.

19.4.3 The makeinitial.py File

After these preliminaries, we can finally dive into the actual Python code. Here's the *makeinitial.py* file in full:

```
import inkex
from lxml import etree

class Makeinitial(inkex.EffectExtension):

    def add_arguments(self, pars):
        pars.add_argument("--initialsize", type=int)

    def effect(self):
        texts = self.svg.get_selected(inkex.TextElement)
        for text in texts:
            self.__makeinitial(text)

```

```
14      def __makeinitial(self, node):
15          if node.text != None and node.text.strip() != "":
16              char = node.text.strip()[0]
17              charAt = node.text.index(char)
18              newNode = etree.Element("tspan")
19              newNode.attrib["style"] = "font-size:" + str(self.options.initialsize) + "%"
20              newNode.text = char
21              newNode.tail = node.text[charAt + 1:]
22              node.text = ""
23              node.insert(0, newNode)
24          elif len(node) != 0:
25              self.__makeinitial(node[0])
26
27  if __name__ == '__main__':
28      Makeinitial().run()
```

Naturally, the first thing we do is `import inkex`. Since we will be manipulating the document's tree (to create a new `tspan` node for the initial), we also need to import the `etree` component of `lxml`, which is Python's library for dealing with XML (if you're unfamiliar with it, read the tutorial at *https://lxml.de/tutorial.html*).

We create the class for the extension by extending `inkex.EffectExtension` (19.4.2). The first obligatory method to implement in our class is `add_arguments`. In it, we call `pars.add_argument`; the first string argument of that method is the command line parameter (therefore starting with `--`) that Inkscape passes to our script, and it must match the name of the extension's parameter (`initialsize`) as defined in the *.inx* file. It is also the name by which we will access its value later on.

The `effect` method shown in line 9 does the actual work. The call `self.svg.get_selected(inkex.TextElement)` returns the list of elements that the user has selected before calling the extension, filtered to contain only text elements. We then go through all these elements and call our own method, `__makeinitial` (the double underscore in the name is Python's convention for private methods), on each one in turn.

In `__makeinitial` (line 14), things get a little more complicated. We need to get the first character of this text element's textual content. However, in SVG, the `text` element usually does not contain text under it; instead, it contains `tspan` elements for the lines that, in turn, may contain further `tspans` for styling. We therefore check if this node has any text at all (`node.text != None`) and if its text is not all whitespace (`node.text.strip() != ""`). If that is not so, we call `__makeinitial` recursively on its first child, if it has one (lines 24–25); in `lxml`, elements are lists of children, so the check for presence of any children is `len(node) != 0`, and then the first child is simply `node[0]`.

If our current node does have text, we extract its first non-whitespace character via `node.text.strip()[0]` (line 16). Since it may not be the first one due to whitespace, we calculate its index (line 17). Then, we create the new `tspan` node in line 18 (note for XML geeks: don't worry about namespaces here because the SVG namespace is the default one in Inkscape's SVG documents). We retrieve the extension parameter's value via `self.options.initialsize` and, in our new node, construct the `style` attribute that uses it in the `font-size` property (line 19).

Per CSS rules, the percentage value in the property refers to the font size of the parent node (that is, the `node` we're processing now).

We now need to take care of the text. The initial character gets assigned to the new `tspan` node, as its `text` property, in line 20. As for the rest of the text, however, we need to cull it out from the current `node` (line 22) and assign it to the `newNode` as its `tail` (line 21). That's because in `lxml`, the `text` property of a node holds only that part of its textual content that comes before its first child node. After that, each of the child nodes holds the fragment of text that comes after it, if any, in its `tail` property.

Finally, the `newNode` is ready, and we insert it into the current `node` (line 23).

The last two lines of the script (27–28) are a bit of black magic required for it to run. We check whether this script is run as a stand-alone program (as opposed to, for example, being imported into some other script) and, if so, run our class's `run` method (supplied by its `inkex` parent class).

19.4.4 Deploying and Testing

Both the *.py* file with the extension's code and its *.inx* file are placed into Inkscape's extensions folder. In Preferences ▸ System, you can look up the Inkscape extensions folder—this is where the extensions that ship with Inkscape live and where you search for inspiration and code to reuse. Your own extensions, however, should go into the User extensions folder; on my system, it's in *C:\Users\dmitry\AppData\Roaming\inkscape\extensions*. This way, your code will not be lost when you upgrade to a newer version of Inkscape.

Copy your files over, restart Inkscape, and voilà! The new extension shows up in Extensions ▸ Text menu and can be used on any text object, as shown in Figure 19-5.

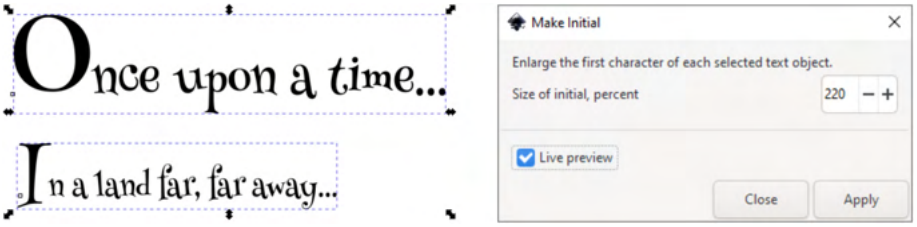


Figure 19-5: The new Make Initial extension in action

20

TUTORIAL: DESIGNING A BUSINESS CARD

Business cards have a role of their own even in the age of virtual meetings. They reflect your personality or that of your organization and present what you most want others to know or remember about you. Like in a three-line haiku, you have a limited space to make a clear, original, and memorable statement that may well have a longer life than your current website.

Simplicity, limited space, and the need for the design to stand out: those criteria make a business card a good test project for a vector editor as a design tool. Inkscape's toolset makes it perfect for the job—even though it is not the best piece of software for creating designs to be printed on paper (18.8).

This tutorial shows two different business card designs. I'm not inviting you to follow the steps exactly (unless you just want to learn the techniques). If you plan to create a business card, look at more examples than these two, and play with Inkscape to combine the best design tricks shown here with something original. Creativity cannot be taught, but it can be inspired. Inkscape will serve you whether you're being artistic or creating a perfectly traditional design.

20.1 Design 1: Simple Graphics

To start creating a business card, choose **New from Template** from the **File** menu, select **Business Card** in the list, select the preferred size in the **Business card size** list, and click **Create from template**. This opens a new Inkscape window with the properly sized business card canvas. The examples in this tutorial use the 90×50 mm size.

Switch to the Text tool and create text objects for all the text lines you will have on your card: name, position, address, phone, and so on (Figure 20-1). Make them all independent objects (click and type each one separately) because you're going to move them around a lot, trying different layouts. If you want to display a logo on your card, import the logo file (18.2). If your logo is only a low-resolution bitmap, you can trace it (18.5) and manually clean up the resulting paths, then delete the bitmap.

Figure 20-1: Preparing the workspace and adding text objects

20.1.1 Choosing Fonts

The next step is choosing font(s) for your text objects. Use a font you like to help personalize your design. If you're interested in graphic design, you likely have a library of favorite fonts; otherwise, a few basic (but good) fonts usually come with your operating system, and you can find many high-quality (and inexpensive) fonts online.

I've chosen the free Gentium font[1] for this design, as it's aesthetically pleasing and features broad Unicode coverage. After assigning the same font to all the text strings, play with their relative sizes using the Selector (Figure 20-2).

Figure 20-2: Fonts and sizes

[1] Available at *https://software.sil.org/gentium/*.

Do different sizes of the same font look good together? If they don't, try using different fonts for different elements, but keep in mind that using more than two fonts may look disorganized in such a small design.

To make Inkscape see a new font, just install it as you usually do in your operating system (for example, on Windows, right-click the font file and choose **Install for all users**) and restart Inkscape. The new font will be listed in the Text and Font dialog as well as in the Text tool's drop-down list (15.3.2).

20.1.2 Layout

When making a template for multiple business cards (or other design items), you need to leave extra space for variable-length names and addresses. However, I'm making a card only for myself, which frees me from this requirement and allows me to position and align all text objects precisely. I was able to push all the address bits closely against the name, creating an asymmetric composition tightly bound together by its alignments (the address is aligned with the start of the last name, the email with the top of the name, and so on), as shown in Figure 20-3.

Figure 20-3: Laying out the text

This layout looked interesting, but perhaps a little too rectangular. An obvious thing to try was to select all (Ctrl-A) and rotate (press [once). Much better! The design now has a flair of constructivism—a short-lived but influentional design movement of the 1920s that espoused bold contrasts and text running at an angle. Let's develop this style further: add three black corners intruding into the composition from the edges and place a big red circle in the center of the composition (Figure 20-4). Constructivists loved simple geometric forms in black and red!

Figure 20-4: Rotating and adding shapes

To look their best, most text objects require letter spacing adjustments (uniform spacing between all letters in the text) and kerning (intervals within specific letter pairs). In the Text tool, use the controls bar (**Spacing between letters** and **Horizontal kerning** numeric values) to adjust those parameters; for kerning, you could also use Alt-arrows to change the interval at the text cursor (15.4.4). Generally, large text objects look better with tighter letter spacing, while small type needs increased letter spacing for readability.

20.2 Design 2: Artistic Drawing

My first business card design was mostly inspired by the text lines' layout, with graphic elements coming secondary to support and reinforce that layout. You could do it the other way around, starting with a graphic and building the design around it.

For a company business card, the obvious starting point is the logo. If you're designing a personal business card and want to it to be more personable, add a photo that you can additionally trace (18.5) and then tweak as vectors. You could also use a piece of clipart (1.3) as a focal point of your design.

For the second demo card, however, I chose another approach: artistic initials. I switched to the Calligraphic pen, set the Angle to 90 degrees with a Fixation of 100 and drew a couple of intertwined letters. I finally got the letter shapes more or less right, but the result was at best mildly interesting. I tried to improve it by unioning the path objects and then simplifying, insetting, and outsetting it a few times, as Figure 20-5 demonstrates.

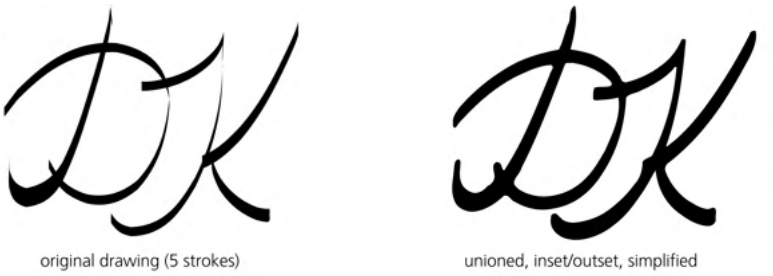


Figure 20-5: Creating initials

Now the letters looked more natural, but I wanted to make them even more interesting. "Don't be too neat" is a viable design strategy; if you can smear, distort, or damage your art in a creative way, go ahead and do it. With this approach in mind, I selected a somewhat narrower pen nib, maximized Tremor to 100, and danced wildly around the letters with my tablet pen (Figure 20-6). Using a mouse for this effect would work almost as well. The result appeared unattractive at first because I forgot to apply the usual Simplify/Inset/Outset magic.

Figure 20-6: Damaging the letter shapes creatively

20.2.1 Layout

This piece of distortion art deserved the central place on the card, with the rest placed symmetrically around it. I used a plain, slanted, very light sans serif font (Helvetica Condensed Light) that doesn't distract from the artwork in the center. The horizontal layout made the card seem cramped so I rotated everything on its side (by pressing Ctrl-[) to give the art ample space and allow the text to float to the edges and not interfere with the artwork, as shown in Figure 20-7.

Figure 20-7: Laying out the card around the initials

20.2.2 Texture and Color

A pleasing design was starting to emerge, but it was far from ready. The card looked too empty, flat, and hostile with the irregular blotch of ink in the center. To fix this, I decided to add some background gradients.

As explained in 10.6, the default gradient from opacity 1 to opacity 0 of a given color (for example, blue) looks crude, because its boundary is clearly visible upon white even when you use a pale color. To improve the look of a gradient over a white background, paint the transparent end of the gradient white, instead of the same color as the other end.

Here, I added four rectangles with irregularly slanted bluish green gradients at the edges of the card to achieve a softly blending, naturally bending, asymmetric look. In Figure 20-8, the dashed lines are the bounding boxes of the four rectangles with gradients, and the gradient lines show each gradient's span and direction. I painted the initials dark blue and added a blurred, 50 percent opaque drop shadow to the letters (Filters ▸ Shadows and Glows ▸ Drop Shadow).

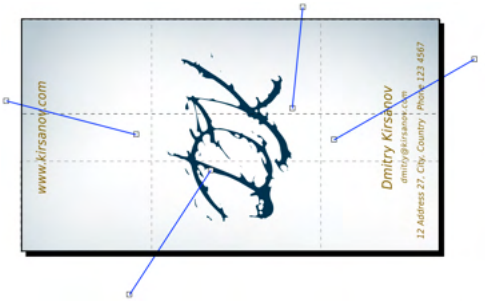


Figure 20-8: Adding gradients and shadows

To make it even prettier, I decided to add texture to the card by overlaying a regular grid of semitransparent lines (Figure 20-9).

Figure 20-9: Adding a striped texture. The card is ready.

To add texture, I drew a rectangle over the entire card, opened Fill and Stroke (Shift-Ctrl-F), switched the fill to pattern, and chose the Stripes 1:1 white stock pattern. Using pattern adjustment handles in the Node tool (10.8.1), I rotated the stripes and scaled them down. Finally, I stepped the rectangle in z-order above the gradients and the initials but below the text, and reduced its opacity to 20 percent.

20.3 Export and Printing

If you've created a business card design and want to print it, first you'll convert the file to PDF.

Inkscape's PDF exporter (B.3) is mature and sophisticated enough to render any design, be it this tutorial's first card (no transparency or gradients, a collection of fully opaque shapes) or the second one (opacity, gradients, and even filters for the drop shadow on the initials). What needs to be rasterized will be rasterized, and text objects will either have their fonts embedded or be converted to paths on export.

If even PDF doesn't cut it, you can always just export an entire design as a bitmap. Inkscape can export only as PNG, but any number of other programs will convert a PNG to a different bitmap format, such as TIFF (old but still popular in the print world).

20.3.1 Using Device Colors

Sometimes a print service will require device-specific CMYK or spot colors (for example, a mandated spot color for your company logo), rather than a PDF or TIFF file. Although Inkscape provides some support for color-managed display, you can't export anything except the RGB screen color space into any output format. You'll need to use external software to perform the color separation (18.8.3).

20.3.2 Tiled Output

If you plan to print the business cards on an office or home printer, you will probably use A4 or Letter paper with multiple copies per sheet that you cut into separate cards. To prepare a printable file, group all of your card's objects, then use the **Create Tiled Clones** dialog (16.6) to create, for example, a 2×5 grid of clones of the group that will fit on your printable page (Figure 20-10).

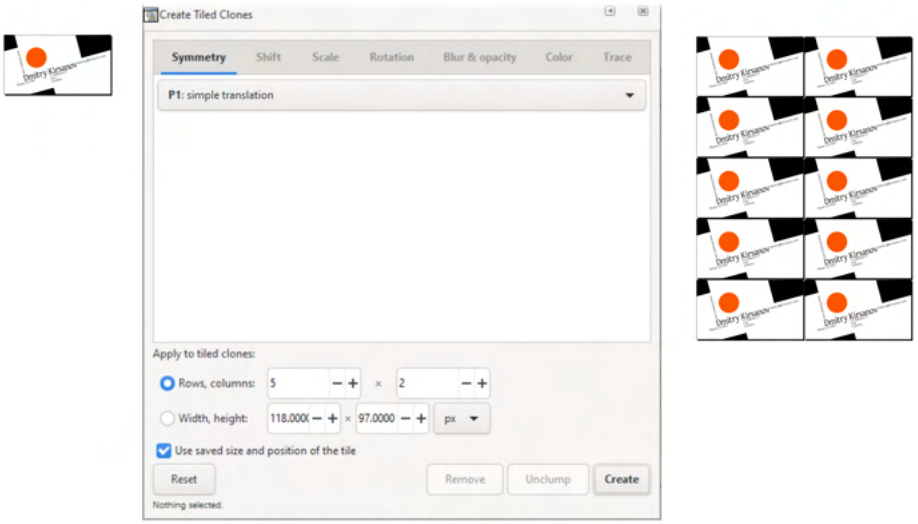


Figure 20-10: Tiling the card to fill the output page

21

TUTORIAL: CREATING AN ANIMATION

From the start, SVG was meant as a language for both static and animated vector graphics. It supports both scripted and declarative animation. For better or worse, however, animated SVG on the web never really caught on, and now it seems like the days of vector animation of any kind (such as Flash) are behind us. HTML + JavaScript or pure video are now what mostly move things online.

Inkscape does not support animated SVG documents (although you can add animation attributes manually via the XML Editor) and can display SVG documents only statically. Still, you can use Inkscape to create static frames and then combine them into an animated GIF or a video—although without timeline control, it's difficult to work on anything longer than several frames long.

21.1 Creating the Template

The easiest way to create animation frames in Inkscape is by putting them on separate layers (4.9). By toggling adjacent layers to be visible, you can see how your frames stack up and control what changes from one frame to the next.

Creating many layers manually is tedious, so I wrote a simple Python script that creates a 200×200 px document with 100 empty layers:

```
print ("""<svg width="200" height="200"
  xmlns:svg="http://www.w3.org/2000/svg"
  xmlns:inkscape="http://www.inkscape.org/namespaces/inkscape">""")
for i in range(100):
  print ('<g inkscape:groupmode="layer" display="none" id="%03d"/>' % (i + 1))
print ('</svg>')
```

All layers are created hidden (that's what `display="none"` is for), so in Inkscape, you will need to unhide them one by one (4.9.3) to draw on them. You can change the number of layers that the script creates (`range(100)`) as well as the artboard's dimensions (`width="200" height="200"`). Save the script into a file named *generate-layers.py*, and at a command prompt, run it and capture its output to an SVG file. (You will need Python installed on your computer; get it at *https://python.org/*.) Then, run Inkscape on this file:

```
$ python generate-layers.py > ani.svg
$ inkscape ani.svg
```

You can also put the resulting file (*ani.svg*) into your *~/.inkscape/templates* folder, where it will work as a template, so the next time you will be able to create an empty 100-layer file by choosing it from the File ▸ New list. Figure 21-1 shows Inkscape's Layers dialog (4.9.4) with the multilayer file loaded.

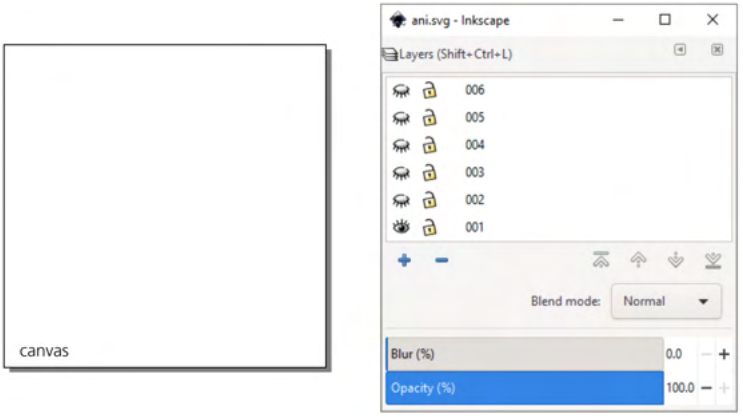


Figure 21-1: A 100-layer animation template loaded into Inkscape

21.2 Creating the Character

Let's make an animation of a dancing man, which is fun yet simple enough not to require much drawing skill and fits the animation's small format. My hero's claim to fame will be the way he moves, so I don't need to make him look too sophisticated. A simple stick figure will do (Figure 21-2).

Figure 21-2: The stick figure, showing its nodes in the Node tool

It consists of three simple paths (arms, legs, and body) and one ellipse (head). Use the Pen tool (14.1.1) with Ctrl to draw horizontal/vertical straight lines; use the Ellipse tool (11.4) to create the head. To facilitate interpolation, do **Path ▸ Combine** on the body and limbs so they become one path, and use Ctrl-Alt-click in the Node tool (12.5.3) to add nodes in his elbow and knee joints.

21.3 Tweening

Now make a copy of the character by duplicating it (Ctrl-D with both head and body selected), move it to the right, and play with its nodes (in the Node tool) to give our man a funky dancing pose (Figure 21-3). This provides the two *keyframes*, and the entire animation could be as simple as alternating between them.

Figure 21-3: Two keyframes

Let's add some intermediate frames to make the transition between the keyframes smoother. Animators call this *tweening* (derived from *between*). Select both bodies (that is, the two path objects) and do **Extensions ▸ Generate from Path ▸ Interpolate**. Specify the number of **Interpolation steps** (say, 4), select **Interpolation method 1**, and, if desired, use a nonzero **Exponent** value to make the movement speed up or slow down nonlinearly. Then, create the heads for each of the tweened bodies by interpolating the two ellipses, using the same number of steps and the same Exponent (Figure 21-4).

Figure 21-4: Interpolating the keyframes

If the tweening steps don't look right to you, undo the interpolation, tweak the keyframes, and re-interpolate until you get what you like. Interpolation method 1 matches the nodes that are at the same position along the path, so it works best when one keyframe path was created by tweaking the other one without adding or removing nodes (as in our case). If two paths are of different origin and have incompatible nodes, method 2 is better.

21.4 Compositing

Let's place our animation in its proper place on the canvas and distribute the frames across layers. Remove the tweening and move the second keyframe right over the first. (Note that the dancer's right foot coincides in both keyframes because it rests on the floor.) Select both overlapping figures and place them on the canvas, scaling if necessary and leaving space for whatever other elements you plan to add (such as a text heading). Finally, interpolate the bodies and the heads again—in place, as shown in Figure 21-5.

Figure 21-5: Interpolating in place

The next task is a bit boring. You need to ungroup the interpolation paths and manually place each one on its own layer, from 002 to 005, with the keyframes occupying layers 001 and 006. Then, duplicate the frames in the reverse order: frames 005 to 001 will go into range 007 to 011 to make the dancer smoothly return into the original position.

Use Shift-Page Up and Shift-Page Down to move the selected objects one layer up/down, and watch the status bar that shows in which layer your selected objects are located. Alternatively, you can cut (Ctrl-X) an object, switch to the destination layer, and paste it in place (Ctrl-Alt-V). Finally, make sure all the layers you've used are visible, and save the SVG file.

21.5 Exporting

You can export the layers manually, but that would be tedious—especially if you need to do it again after viewing the result and making changes to the source. Fortunately, Inkscape has convenient command line parameters, so I wrote another Python script to automate the export:

```
import os, sys
for i in range(int(sys.argv[2]), int(sys.argv[3]) + 1):
  os.system("""inkscape --export-filename=%s-%03d.png --export-id=%03d \
    --export-id-only --export-dpi=400 --export-area-page \
    --export-background-opacity=1 %s""" % (sys.argv[1], i, i, sys.argv[1]))
os.system("convert -loop 0 -delay 10 %s-*.png %s" % (sys.argv[1], sys.argv[4]))
```

The script takes as parameters the name of the source file, the numbers of the start and end layers to export, and the name of the resulting GIF. It calls Inkscape to export each layer separately; then, it calls the `convert` utility from ImageMagick to combine the frames into an animated GIF file that can then be viewed in any web browser.

Save it as *produce-gif.py* and run:

```
$ python produce-gif.py ani.svg 001 011 stick.gif
Exporting only object with id="001"; all other objects hidden
DPI: 400
Background RRGGBBAA: ffffffff
Area 0:0:100:100 exported to 444 x 444 pixels (400 dpi)
Bitmap saved as: ani.svg-001.png
Exporting only object with id="002"; all other objects hidden
...
Exporting only object with id="011"; all other objects hidden
...
```

View the output at *http://www.kirsanov.com/inkscape-animation/stick.gif*.

NOTE *Instead of the `convert` utility from ImageMagick, you can use the open source Gifsicle program (*http://www.lcdf.org/gifsicle*) for composing frames into an animated GIF file. Its advantage is that it can optimize the animation to reduce its file size.*

21.6 Freehand Drawing

So far, our stick-figure animation looks at best mildly engaging. This smooth, vectory style is good for technical animations, such as demonstrating the workings of a machine, but it's not too inspiring for an animated dance.

To improve the stick figure, hide all layers except `001`, select the figure, and lower its opacity. Then arm yourself with the Calligraphic pen (14.2), select a **Width** of 20 with **Tremor** of 40, and draw freehand strokes over the stick figure. Try to make it more random, funky, and personalized; add more pronounced feet and fists and more human-like body forms (Figure 21-6).

Figure 21-6: Humanizing the figure with Calligraphic pen

When done, delete the original skeleton figure. It has served its function of a blueprint and is no longer needed. The result looks a bit foreign in its roughness upon the immaculate white background. To fix this, reduce the width of the pen to 1 and add some thin random strokes around the dancer, hinting at his limbs' motion and shadows on the floor, as shown in Figure 21-7. Don't

worry if this looks *too* random—in the moving figure, this randomness will come alive and seem natural.

Figure 21-7: Skeleton removed, motion noise added

The main rule with this kind of project is *don't copy*. Freehand roughness cannot be recycled. No matter how similar one frame is to another, you need to sketch each frame entirely from scratch, using nothing but the stick figure as your guide. Duplicating the freehand strokes—even if you move or scale them—instantly kills the rough, natural feel and makes your animation wooden and dull. Don't be lazy; the more you draw the easier it gets. Check out the complete hand-drawn animation at *http://www.kirsanov.com/inkscape-animation/rough.gif*.

You can use the same technique to trace with the Calligraphic pen over an imported bitmap manually. Make the bitmap half-transparent and sketch on top of it, trying to highlight the most important features and ignore the rest. In an animation, the source bitmaps might be frames of a video, still photos, or rendered 3D images.

21.7 Adding Text

Let's move the entire animation down to free up some space for a text heading above it. Unhide all layers, select all objects on all layers (Ctrl-Alt-A), and move them downward.

The banner on our animation will be just a single word: "dance!". Can we do something more interesting than copying the same static text object into each frame?

We could use the Calligraphic pen to draw ruffled handwritten letters over some text object used as a guide—rendering the entire animation in the same style. However, for the purpose of demonstration, let's try something different: make the text banner wave smoothly as if on a flag. We could use the Envelope Deformation path effect (13.3.3), but it's probably easier to apply some node sculpting instead (12.5.7.2).

To begin, create a text object using a nice-looking font, convert it to path (Shift-Ctrl-C), ungroup (Ctrl-U), union (Ctrl-+), switch to the Node tool (F2), select all nodes (Ctrl-A), and Alt-drag one of them. The entire shape will smoothly bend and stretch, as Figure 21-8 demonstrates. If the letter shapes become too distorted, undo the drag and press Insert a couple times, each time

doubling the number of nodes—this usually helps make the path you're sculpting behave more naturally.

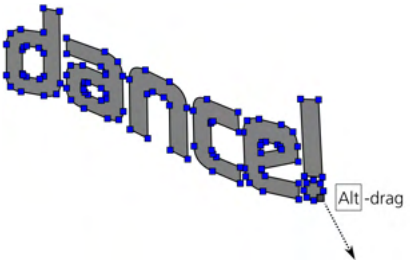


Figure 21-8: Node sculpting on a text banner

Make two copies of the text object, sculpt each one differently—so they look like two shots of a banner floating in the wind—and interpolate between them. Then, just as we did for the dancer figure, distribute the interpolation steps into layers, placing them above the figure on the canvas (Figure 21-9). Don't worry about precise placement; slight banner wobbling isn't a problem.

Figure 21-9: Interpolating the banner

To see where to place each object relative to the previous layer, use the **Opacity** control on the **Layers** dialog to make the previous layer temporarily half-transparent. Compose the animated GIF again using the *produce-gif.py* script. Figure 21-10 shows the frames 001 to 005.

Figure 21-10: Half of the animation's frames (forward movement)

Visit *http://www.kirsanov.com/inkscape-animation/with-banner.gif* to see the result so far.

21.8 Adding Color

The only thing not to like about our animation is its total lack of color. Let's add a different colored background to each frame—for a real stroboscopic dance floor effect. To automate the random color selection, create a rectangle with an unset color and use the **Create Tiled Clones** dialog (16.6) to multiply it with some hue variation.

Paint the "dance!" banner with a contrasting bright color, also varying from frame to frame. Finally, to make the dancing man stand out from the background, add an elliptic gradient "spotlight" behind him; randomly move, scale, and rotate the spotlight on each frame for an additional energizing effect. The final version is at *http://www.kirsanov.com/inkscape-animation/final.gif*. Enjoy!

22

TUTORIAL: DRAWING A 3D-CORRECT CARTOON

Inkscape's 3D Box tool (11.3.1) is not intended as a replacement for specialized 3D design applications. Inkscape is a drawing program, and its 3D Box tool is best used as a *drawing aid*, which is how we'll use it in this simple tutorial.

A *3D-correct* drawing is one that satisfies the rules of perspective that have been known to artists for centuries. It does not need to have perfectly realistic shading and texturing; it just needs the lines and shapes to be approximately correct with regard to angles and dimensions.

Intentional perspective distortions may well have an artistic value of their own, and some drawing styles don't use perspective at all. When necessary, many artists are capable of drawing 3D-correct art without any technical aid. However, quite often you will notice perspective errors—from barely noticeable to embarrassing—in otherwise well-done pieces of art. I would love to see more artists—not only beginners—use Inkscape's quick and easy way to set up the perspective for a drawing without tedious measurements and helper lines.

22.1 The Room

Let's draw a simple scene of two people meeting in a room. Switch to the 3D Box tool (X) and drag in the middle of the canvas to draw the box for the room. To see all sides of the box at once, enter it as a group (Ctrl-Enter), select all sides inside (Ctrl-A), and set the **Opacity** slider in the status bar (marked O:) to 50 percent, as shown in Figure 22-1. Then, leave the group (Ctrl-Backspace).

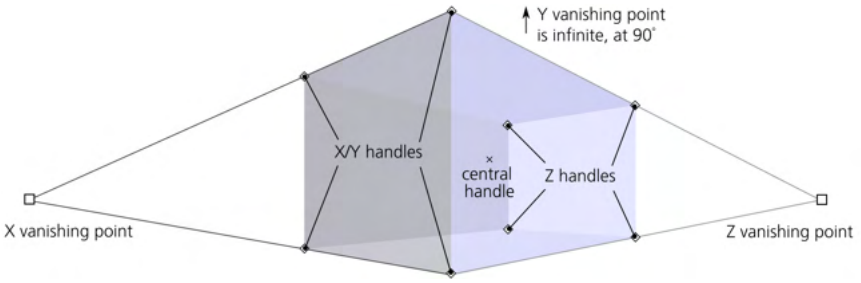


Figure 22-1: Creating the room

Don't worry about getting the shape right on the first try; keep dragging the box's handles and, if necessary, the vanishing points to adjust what you've drawn. Use the corner handles to resize your room. The four X/Y handles move in the X/Y plane by default and along the Z axis when you press Shift. The four Z handles, conversely, move along the Z axis without Shift and in the X/Y plane with Shift. The X-shaped central handle will move the entire room, again in the X/Y plane without Shift and along the Z axis with Shift. When moving a handle in X/Y, press Ctrl to restrict its movement to one of the axes (only X or only Y). Dragging the vanishing points will reslant and resize the entire room to fit the changed perspective (11.3.3).

You can create a third vanishing point on the Y axis in addition to those in X and Z. In other words, you can make the Y vanishing point *finite*. Do this if you want your drawing to have a more dramatic or exaggerated perspective—as if viewed by a spider in the corner of the room. Conversely, you can make the X and/or Y vanishing points infinite if you want a more detached, "scientific" view from far away, as shown in Figure 22-2.

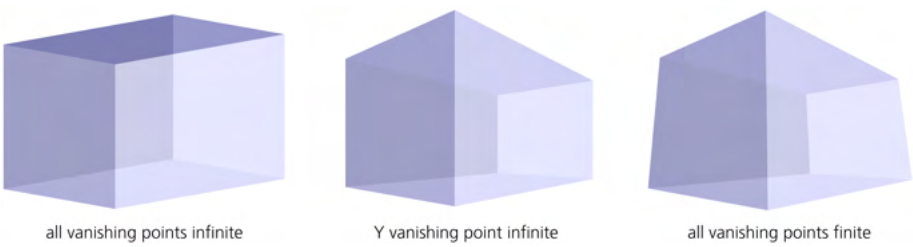


Figure 22-2: Different perspectives

22.2 The Furniture

Let's add some furniture—for example, a sofa.

When creating 3D box compositions, using snapping (7.3) is a convenient way to make boxes align precisely. On the snap controls bar (7.3.1), make sure you have the following toggles on: Enable snapping (topmost), Snap nodes, paths, and handles, and Snap to paths.

Now, draw a smaller box inside the larger one. (Here, "inside" refers to perspective; in z-order, the boxes can be in any order so long as their sides are semitransparent and you can see the edges of all boxes you created.) The new box will use the same perspective (the same vanishing points) as the room box. Grab the lower-left X/Y handle and drag it so it snaps to the lower-left Z-edge of the room. That will be the base of the sofa; use the other corner handles to give it the correct height, width, and depth (Figure 22-3).

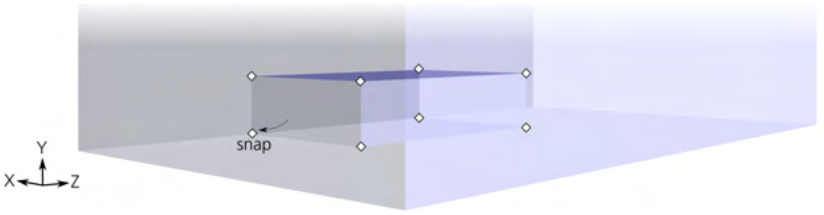


Figure 22-3: The base of the sofa

For the back of the sofa, draw a new box and snap it into alignment in a similar way; or, even easier, duplicate the base (Ctrl-D) and drag its X/Y handles to resize it to the correct shape, as shown in Figure 22-4.

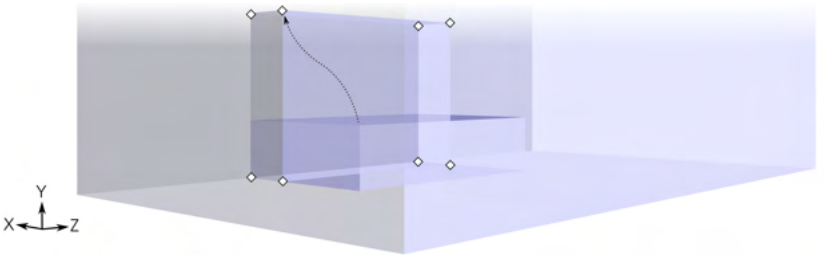


Figure 22-4: The back of the sofa

Let's add two small basement windows on each side of the sofa. A couple boxes flattened in the X dimension will make perfect windows. However, how do you place them properly, given that no side of a window aligns with a side of any other box?

Again, the duplicate-then-resize trick is the easiest way to achieve this. Duplicate the largest box you have—the room itself. Ctrl-drag its front X/Y handle backward (along the X axis) to squeeze it into a thin layer; then, drag four handles on the X/Y plane to resize the window into place. For a second window, duplicate the first one and Shift-drag its central handle in the Z direction; this

way, the second window will be an exact perspective-aligned copy of the first one. Using the same duplicate-and-flatten trick, add a doorway box on the Z-most wall (that is, the wall closest to the Z vanishing point), so that it protrudes outside of the room box, as Figure 22-5 demonstrates.

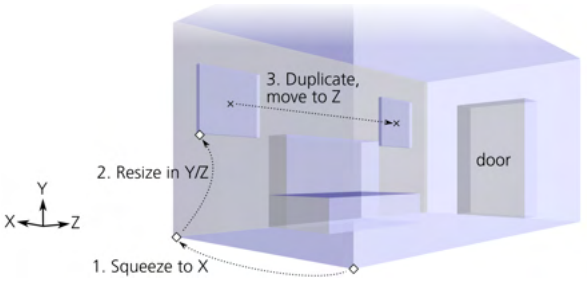


Figure 22-5: The windows and the door

22.3 People

Let's draw two people, one standing by the sofa and the other in the doorway. How can the 3D Box tool help with drawing humans? I'm not going to model robot-like constructions out of parallelepipeds, but I still want the height and bulkiness of each figure to be correct in the drawing's perspective, and an easy way to ensure this is with the 3D Box tool.

Here's how you do it. First, add a tall, narrow box in the doorway. This will be human 1. As before, duplicate the doorway box, squeeze it on the sides in the X/Y plane, and resize it for a realistic proportion of the figure height and doorway height. (Ask someone to stand in a real doorway to get an idea.)

Then, duplicate the human 1 box. Using its X-shaped handle in the middle of this new human, Shift-drag it to the front, then Ctrl-drag it horizontally a bit along the X axis. This will be the second human standing beside the sofa (Figure 22-6).

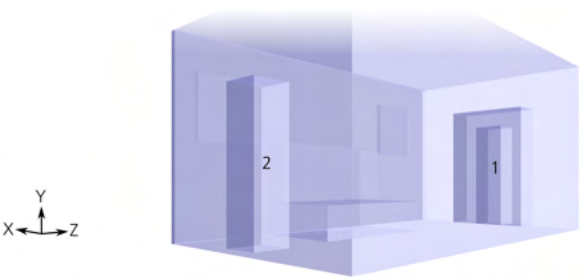


Figure 22-6: The two people

This way, both people boxes have the same perspective-adjusted height and will serve as useful guides for actually drawing the human figures. If I were trying to achieve even better anatomic fidelity, I could use a stack of three boxes for the legs, body, and head for each human. (Unfortunately, you cannot drag several selected boxes in 3D space *at once*; you have to drag them one by one.)

22.4 Sketching and Coloring

Enough boxes! Put away the 3D Box tool, arm yourself with the Calligraphic pen, and start sketching on top the 3D blueprint. Create a new layer (**Layer ▸ Add Layer**) so you can show and hide the 3D boxes and the drawing separately.

Don't try to trace the edges of the boxes precisely. For nongeometric objects, such as the sofa and humans, this is hardly possible, but even the walls and windows will benefit from some hand-drawn roughness and imperfections. Use the edges of the 3D boxes as general guides, but otherwise, *draw freely* (Inkscape's motto) in your own manner and style (Figure 22-7). You certainly don't need to trace *all* of the lines; you can omit some and hint at others with short partial strokes.

Figure 22-7: Sketching over the 3D blueprint

You will discover that the 3D boxes make it much easier to keep in mind the overall composition and space relationships in your drawing while working on the details. I won't go into details of the drawing process; you've seen my sketching style with the Calligraphic pen in another tutorial (21.6).

Once you're done with the basic outlines, turn off the 3D layer to see how your image stands on its own merits, as shown in Figure 22-8.

Figure 22-8: When the sketch is complete, hide the 3D layer.

Finally, colorize the drawing using wide blurred colored strokes underneath the dark crisp outlines, as Figure 22-9 demonstrates.

Figure 22-9: Adding color

23

TUTORIAL: ARTISTIC DRAWING

Among vector editors, Inkscape is one of the best for sketching and freehand drawing, thanks to its versatile Calligraphic pen tool (14.2). Far from being geometrically perfect and lifeless like other vector tools, the Calligraphic pen has character—but it is still a vector tool that produces always editable vector objects.

In this tutorial, we'll go through the process of creating a simple drawing entirely in Inkscape. I'll draw a funny, cartoon running horse that might be a mascot or a comic character. I've always found it challenging to draw anything from scratch, and I imagine many readers can sympathize. Yet with Inkscape, I was able to make a drawing I actually liked.

23.1 The First Sketch

Start by switching to the Calligraphic pen tool and setting a **Width** of 0.05 and a **Thinning** of 0.2 in the tool's controls bar (14.2). If you have a graphics tablet, enable the pen pressure for varying the stroke width (14.2.1.1). The Angle and Fixation parameters make sense only for calligraphy, not freehand drawing, so

set the **Fixation** to 0—this effectively turns a fixed-angle pen into a round brush that has no orientation (14.2.2).

Believe it or not, I've never drawn a horse before, and my first several strokes make this brutally obvious. With a beginning like this, I imagine many would have been tempted (as was I) to give up immediately. But patience and hard work pay off—especially in Inkscape. Just keep throwing strokes onto the canvas, undoing, tweaking, and stroking again. Sooner or later, something not *entirely* awful will flash through the tangle of bad lines (Figure 23-1).

Figure 23-1: The first strokes

For most people, drawing objects requires visual aids. You may find it hard to visualize on your own how a horse's body curves or which way the legs bend. I found that photos of real horses were of little help; much more inspirational were stylized *drawings* of horses—where other, more capable artists have already done the hard work of abstracting and amplifying the core equestrian features. A Google image search will provide you with plenty of reference material. After much sketching, undoing, and emphasizing (by adding pen pressure), I had the sketch shown in Figure 23-2.

Figure 23-2: The first sketch

23.2 Inking

A traditional comic or cartoon artist's workflow has two main stages: *sketching* (typically with a pencil) and *inking* (with a pen) over the rough sketch. You can follow the same process in Inkscape. As soon as the drawing looks more or less like a horse, start inking *over* it, on a new layer, to develop it further. This way, you can amplify what is good in your sketch and build on that without the risk of destroying it with too much tweaking. Select all strokes (Ctrl-A) and assign them an opacity of 0.05 so they become barely visible; this effectively hides the thin strokes, making it easier to concentrate on the bold ones. Then, lock this layer (using the lock toggle button in the status bar), create a new one (**Layer ▸ Add Layer**), and draw the same horse *again* using the sketch as a guide, as shown in Figure 23-3.

Figure 23-3: Inking the sketch in a new layer

Is this the same horse? Not quite. The horse in the first sketch looked like a real horse—perhaps a bit too real. What we want is a cartoon, not photorealism. In cartoons, creatures' heads (especially the eyes) and feet (or hands or paws) are often enlarged, out of proportion. In the first inking layer, let's do just that: keep the body the same but enlarge the hooves and the head, adding a pair of oversized cartoonish eyes (Figure 23-4).

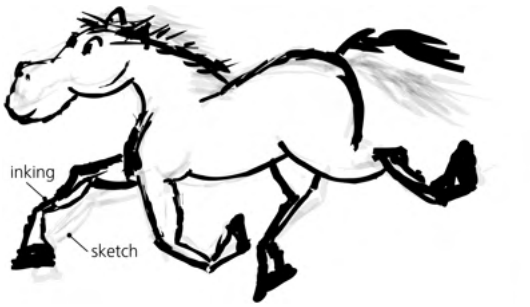


Figure 23-4: Making the horse more cartoonish

23.3 Tweaking

A vector editor's unique advantage is that all strokes remain independent objects, which makes nudging, scaling, or rotating parts of a drawing easy. To make such adjustments on the horse drawing, with the Selector tool, drag around the leg or the head to select all its objects and use either the mouse handles or the convenient keyboard shortcuts: Alt-arrows to move, Alt-< or Alt-> to scale, or Alt-[or Alt-] to rotate. Often, you'll make a good stroke but at a wrong scale or in the wrong location—instead of deleting, transform it to fit. Even if it already looks acceptable, playing with a character's limbs or facial features can sometimes make it more expressive than you thought possible (Figure 23-5).

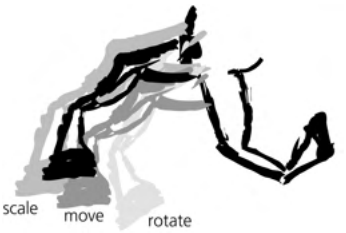


Figure 23-5: Adjusting parts of the drawing

You are not limited to just one inking layer. At any time, you can hide the previous sketch layer, lock and fade out your current layer to make it a new sketch, and create a new inking layer on top. With every such transition, a drawing usually will look less and less like a pencil sketch and more like an ink drawing with smooth strokes and elaborate details. Figure 23-6 shows my third inking attempt over the previous sketch layers; making the head look like a horse rather than a hippopotamus was especially difficult.

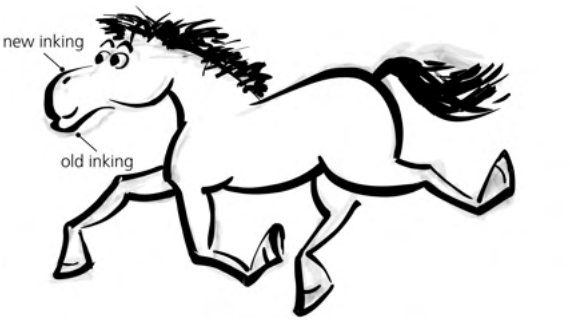


Figure 23-6: Another inking layer

23.4 Coloring and Smoothing

With the outline almost complete, start thinking about coloring the drawing. Inkscape can emulate many different coloring styles. For example, you can imitate a painting by overlaying many random calligraphic strokes and painting each one with a different color and opacity (using the Tweak tool, 8.9.1).

We will try something simpler and more traditional: a flat color fill with some lighting and shading overlays (white and black ellipses with elliptic opacity gradients, 10.1.2). Use the Paint bucket tool (14.3) to create the interior shape; apply a little blur to it (about 2 percent) for extra smoothness (Figure 23-7).

Figure 23-7: Coloring options

Let's return to the outline (hide the coloring layer for now). You can make it smoother and more attractive if you select all the strokes in part of the figure (such as a leg or the head), union them (Ctrl-+), and press Ctrl-L a few times. Performing those steps melts the sharp corners and welds the joints for a much more natural and integrated look (Figure 23-8). Another useful trick is to make all strokes thicker by pressing Alt-) to outset them and then pressing Alt-(to make them thinner (12.4).

Figure 23-8: Welding and melting

Multiple insets and outsets on a path have an effect similar to simplification. Too much insetting may make some parts of a path disappear altogether, but this is not necessarily bad—trust the overall impression and your artistic sense.

23.5 Drawing Hair

For many new artists, drawing realistic-looking hair is hard. Striking the right balance between regularity and chaos, as well as between tidiness and untidiness is critical for the impression your character will make. At this stage of the horse drawing, the tail and especially the mane weren't impressive. I drew them over and over, but all my attempts looked either too lumpy or too bushy (or both). I was finally able to produce a decent wavy hair by using a high value of Tremor (14.2.4) in the Calligraphic pen (Figure 23-9). Maximizing this parameter makes the calligraphic brush strokes nicely blotchy and much more uniformly uneven than I was able to do manually.

Figure 23-9: Making the horse's mane look more realistic

The last touch is to try some scaling, rotating, and skewing of the entire drawing, which can sometimes cause significant improvements without needing to do much work.

Overall, the result was much better than I expected. I'm pretty sure I wouldn't have been able to achieve this level of quality by drawing on paper. Of course, you can do other styles of drawing with Inkscape; experiment to find the techniques that are most natural for you. I have grown to especially like the Calligraphic pen with a high Tremor value, which provides an almost "natural media" drawing tool, as shown in Figure 23-10.

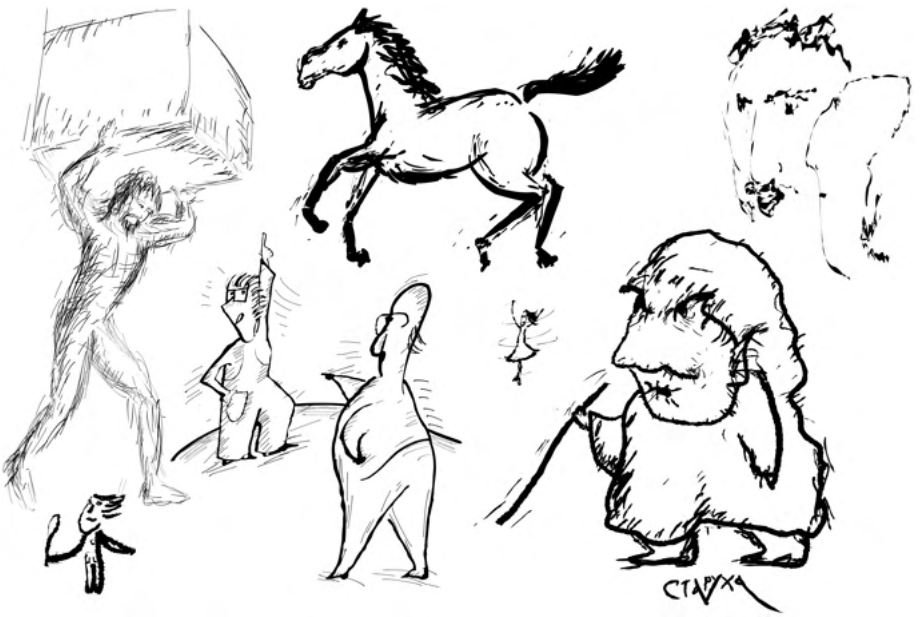


Figure 23-10: More freehand drawing examples

NOTE *While the best device for drawing on the computer is a graphics tablet (14.2.1.1), you can create decent Inkscape drawings with a mouse. A hand with a mouse cannot move equally naturally in all directions; horizontal strokes are usually easier than vertical strokes and much easier than diagonal ones. Rotating the canvas (3.13) is indispensable when you're drawing or hatching with a mouse.*

24

TUTORIAL: TECHNICAL DRAWING

You can use Inkscape to create technical drawings, too. Inkscape is not a CAD (computer-aided design) application, but it works very well for moderately complex technical drawings. If you already know how to use Inkscape, using it for this kind of work is a logical choice. As an example, let's draw an isometric image of an engine part's cross-section (Figure 24-13 on page 449).

The key to creating technical drawings in Inkscape is the use of grids (7.2), snapping (7.3), transforming by numbers (6.6), and shapes, especially rectangles (11.2). Those features allow you to create a 3D view of a complex piece quickly and precisely, without ever having to approximate or adjust anything by hand. Every object, handle, and node just snaps into its exact place. Once you get the hang of it, producing such drawings will be almost addictively easy.

24.1 Setting Up the Grid

An *isometric* image is one where all three coordinate axes are separated by equal angles (60 degrees) and have the same scale. For example, in an isometric cube, all visible edges have the same length.

To start a grid to create the engine part drawing, open the **Document Properties** dialog (7.2.1), choose the **Grids** tab, and create the default axonometric grid with both X and Z angles set to 30 degrees (measured from the horizontal), as shown in Figure 24-1. You may want to set the Spacing Y (the grid being isometric, it applies all three axes) to the minimum measurable distance in your drawing. For example, if all dimensions in your drawing are in millimeters and precise to the first fractional digit, set the grid spacing to 0.1 mm so you never have to place anything in between grid lines.

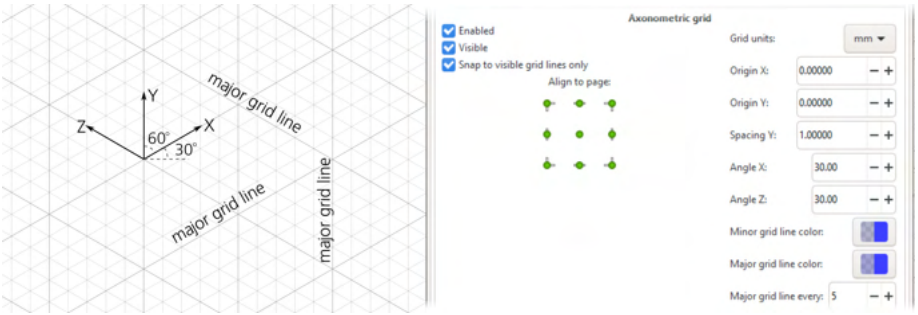


Figure 24-1: Setting up the isometric grid

You can also change the major (darker) grid lines' spacing; by default, they appear every five regular lines. When you zoom out, the regular grid lines disappear first.

24.2 Making the Box

Let's create the object's bottom box. With the grid (and, by default, snapping to grid) on, use the Rectangle tool to draw a rectangle—any rectangle. Notice that the corners of the rectangle snap to the grid line intersections (pay attention to the snap indicator and tips, 7.3.2), but the rectangle is not isometric.

To skew it into the isometric projections, open the **Transform** dialog (Shift-Ctrl-M, 6.7), and on the **Skew** tab, specify 60 degrees for **Horizontal** skew and 30 degrees for **Vertical** skew. After you click **Apply**, the rectangle transforms to fit the grid; note that the rectangle's resize handles now also move in the isometric projection (11.2.1). Now, all you need to do is snap them to the corresponding grid intersections, as Figure 24-2 demonstrates.

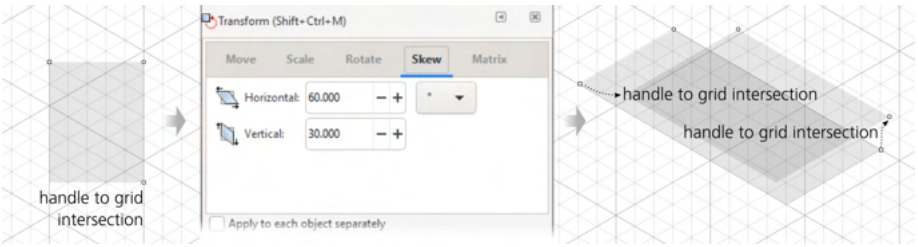


Figure 24-2: Creating, skewing, and snapping the top of the box

The box's other two sides are even easier to create. For the front side, you need only 30 degrees of Vertical skew, with Horizontal skew set at 0. The left side of the box is simply the front side duplicated (Ctrl-D) and flipped (H). After skewing and flipping, snapping the corners of all three rectangles to form a precise, solid, gapless 6×8×2 box is super easy. The X and Z units need to be an even number, so that the two grid lines intersect in the center of the top side (Figure 24-3).

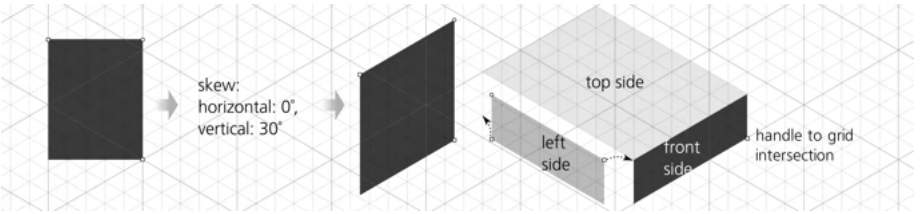


Figure 24-3: Creating the front and left sides of the box

Now that you have one box, you can quickly build up complex architectures by duplicating its sides and re-snapping the corners to new positions. However, this is not what we are going to do; the next step is rounding the box's corners.

24.3 Rounding Corners

To round a rectangle's corners (11.2.2), grab the circular handle on a corner and drag it along the side. In the double-skewed top side of the box, rounding works as expected; the rectangle becomes rounded in its plane with appropriate projective distortion. While Ctrl-dragging one of the rounding handles, snap it to the nearest intersection so that the rounding radius is equal to one grid unit. Then, on the front and left sides, move the sizing handles to make them narrower by one grid unit on each side (Figure 24-4).

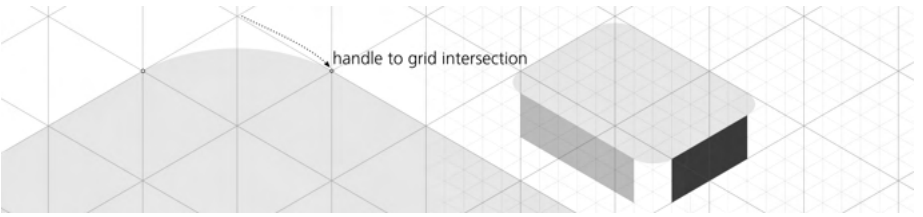


Figure 24-4: Rounding the corners on top of the box

To fill the gaps in the box's corners, create a vertical cylinder and put copies (or clones) in all three of the box's visible corners. You'll also reuse it for the top cylinder of the engine part.

To make a cylinder, start with an ellipse, but you don't even need the Ellipse tool to make it. Instead, just duplicate the box's top side and resize it to an isometric square of 2×2 grid units. Since resizing preserves the rounded corners, each with the radius of one unit, this produces a perfectly isometric ellipse (Figure 24-5).

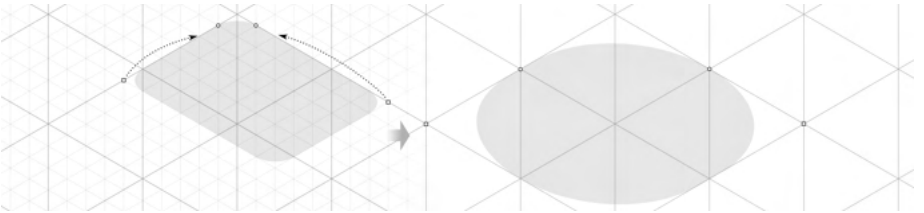


Figure 24-5: Creating an ellipse out of a rectangle

To create a cylinder, you could duplicate this ellipse, move it down, and draw a flat non-isometric rectangle between them. Although workable, this method isn't "clean," because the ellipses' extremities have no grid lines to which to snap the rectangle's left and right edges. Such an approach requires turning off snapping and using manual tweaking with its inevitable speed/precision tradeoff. Let's try another method.

Convert a copy of this ellipse-like rectangle to path (Shift-Ctrl-C) and switch to the Node tool (F2). You will see what looks like four nodes on the path—but in fact there are eight: each visible node is actually two nodes, contributed by the two rounding arcs that meet there. To fix this, drag-select around each of these duplicate nodes in turn (the status bar should say *2 of 8 nodes selected*) and click the **Join selected nodes** button on the Node tool's controls bar. Do this four times, so you end up with a path of four nodes.

However, the ellipse's left and right extremities don't have nodes where you'll need them. Select all nodes (Ctrl-A) and insert new nodes between each of the two selected nodes by pressing Insert (12.5.3), as shown in Figure 24-6.

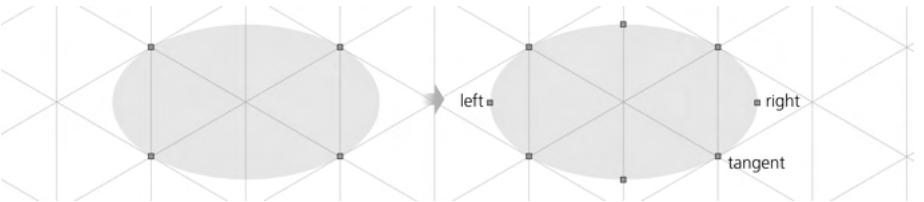


Figure 24-6: Adding nodes at the ellipse's extremities

Now, select only the two nodes at the horizontal extremities (marked *left* and *right* in Figure 24-6) and duplicate them (Shift-D). Add the three nodes in the bottom half of the ellipse to the selection by dragging around them with Shift. Grab the node that is tangent to the grid line (marked *tangent*) and Ctrl-drag it and the rest of the selected nodes downward until it all snaps at a level two units lower, creating the ideally precise cylinder outline. After that, all you need to do is add a copy of the original ellipse back into its place on top, and draw a horizontal gradient (Figure 24-7).

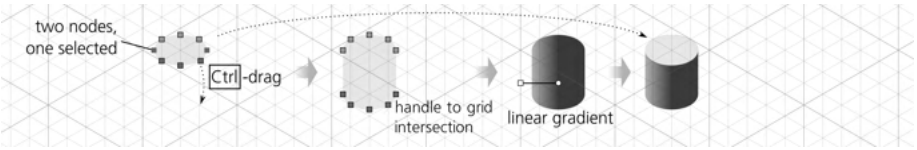


Figure 24-7: Pulling out, shading, and capping a cylinder

Group the cylinder, create three copies of it, and move each one to fill a gap in the rounded box. The cylinders will fit snugly into their places. To make the box look solid, just sort out the z-order and match the colors, as shown in Figure 24-8. Use the Dropper tool (8.8) to copy colors from the flat sides to the gradient stops on the cylinders (or vice versa).

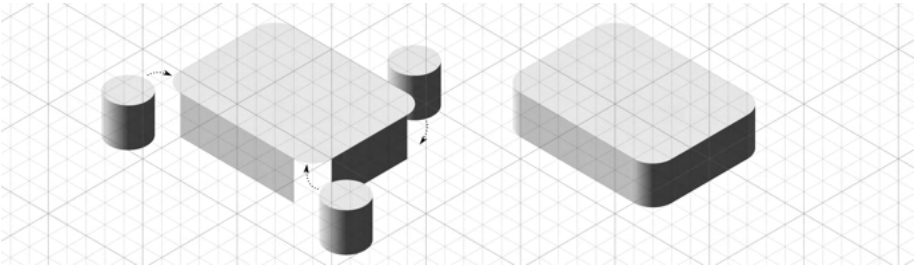


Figure 24-8: The box is ready.

24.4 Making the Top Cylinder

To create the top cylinder, make another copy of the corner cylinder and scale it up twice by pressing Ctrl->, then snap it into place on top of the center of the box.

Duplicate the top ellipse and scale it down to 50 percent in place (Ctrl-<). Now, you don't even need to move it; it's already precisely where it must be to imitate the hole in the top cylinder. All you need to do is make it *look* like a hole by adding a horizontal linear gradient in the opposite direction of the cylinder. While you're at it, make three more copies of the hole, snap each one into the base box's corners, and scale down by pressing Ctrl-< again. These will be the holes for the bolts to fasten the detail in place, as shown in Figure 24-9.

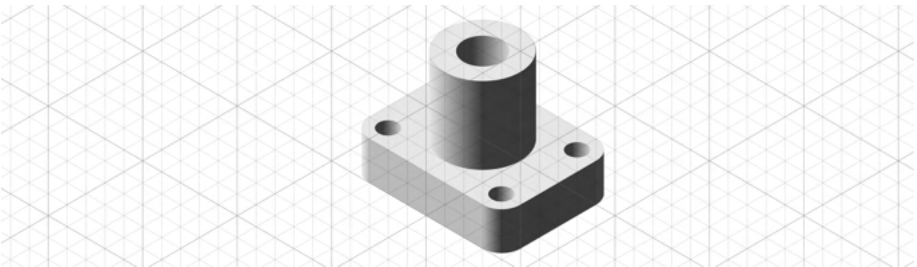


Figure 24-9: Adding the top cylinder and the holes

24.5 Making the Cutout

A full isometric view of the object is now ready. To demonstrate some additional techniques, let's create a cutout of the object showing two perpendicular cross-sections.

Since everything you've done so far is snapped to the grid, adding the cutout shape is very easy. Switch to the Pen tool (14.1.1) and click near the correct grid intersections to create a closed path for the left side of the cross section. Then, while pressing Shift, create a second subpath of the same path, clicking through the right side's corners (Figure 24-10).

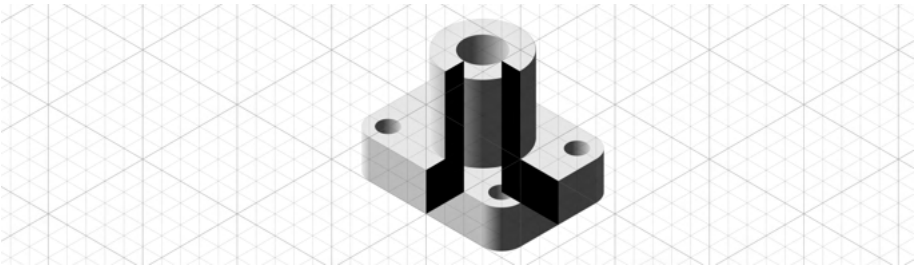


Figure 24-10: Adding the cutout shape

Cutouts in technical designs are often filled with a pattern symbolizing the object's material. Plain stripes are used for metals, and you can use one of the preset patterns to indicate that an object is metallic. Open the **Fill and Stroke** dialog (8.2), click the **Pattern** button on the **Fill** tab, and choose the **Stripes 1:8** pattern. The problem with this pattern is that it shows black stripes on a transparent background, and you need black stripes on a white background. Just duplicate the shape, paint it white, and move the white copy under the striped one in the z-order (Figure 24-11).

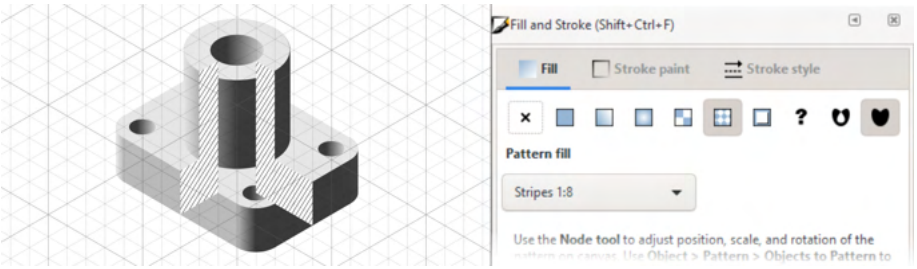


Figure 24-11: Painting the stripes

Now you need to remove the parts of the object in front of the cutout. Select and delete the frontmost rounded cylinder. Using the rectangle resize handles, contract the the box's right and left sides to snap them to the cutout's edges. As for the rest of the objects—the top side of the box and the top ellipse of the cylinder—you'll actually need to *cut* them.

Again using the Pen tool, snap-draw a triangle covering the area you want to cut out. Subtract this triangle from the shape by selecting both and pressing Ctrl-–, 12.2), as shown in Figure 24-12.

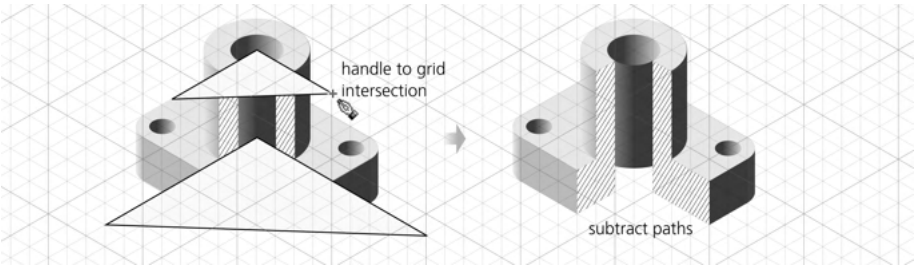


Figure 24-12: Cutting out unnecessary pieces

Now, create the inside of the hole visible through the cutout. Duplicate the cylinder shape, scale it down (Ctrl-<), and step it down in the z-order so it's under the cutout shape and the hole ellipse. This will be the hole's inner surface. Using the Node tool, select and pull down that shape's bottom nodes. Then, duplicate the hole ellipse, move it down to the level of the bottom edge, and subtract it from the inner surface shape.

Finally, you need to paint the inner surface. Select the ellipse at the top of the hole and copy it to the clipboard (Ctrl-C). Then, select the inner surface path and paste the style to it (Shift-Ctrl-V). This will paint the inverted horizontal gradient of the top ellipse on the entire inner surface of the hole, as shown in Figure 24-13.

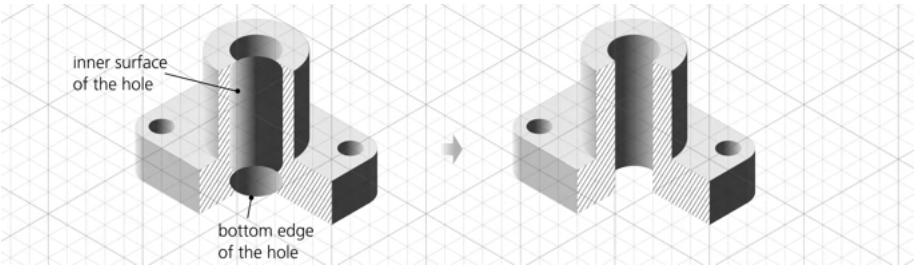


Figure 24-13: Creating the hole's inner surface

25

TUTORIAL: THE ROSE

The complex image of a rose (see Figure 17 in the color insert) was created to showcase Inkscape's versatility. A single drawing of a rose—traced from a photo—is rendered in a different way in each of the six square windows as well as in the background. A single tutorial showing the stages of creating this image is like seven different tutorials on ways to draw a rose—or anything else, for that matter.

I started with a photo, but if you can draw well, you don't need to do that. Just draw whatever you like on an empty canvas. Tracing a photo, however, is a useful technique by itself.

Go to **File ▸ Import** (or press Ctrl-I) to import the photo, and scale and position it as convenient. Rename the layer with the image to background. You will need to hide and show the photo several times, which is much easier to do when it's in its own layer. Reducing the photo's opacity and locking its layer so you don't move it accidentally is a good idea.

Then, create a new layer called ink and start to trace over the outlines and the boundaries between colors in the photo (Figure 25-1).

Figure 25-1: Importing the image and setting up tracing using the Calligraphic pen

The best tool to use for tracing an image depends on the nature of the image and the result you want to get. If you're tracing geometric shapes that can be rendered as straight lines and Bézier segments, use the Pen tool. If you need more freeform and artistic shapes but want to minimize the number of paths and nodes, the Pencil tool is a better choice. Finally, if you want an expressive, artistically untidy style, use the Calligraphic pen. I wanted my drawing to be minimal but artistic and natural, and I had no reason to save on the number of nodes, so I used the Calligraphic pen, as shown in Figure 25-2.

Figure 25-2: Tracing complete: 106 path objects in layer ink

I used a pressure-sensitive tablet pen, allowing me to vary the width of the brush stroke by how hard I pressed the pen. You can get a similar result with a regular mouse, though. In fact, you may even prefer using a mouse, because controlling the stroke thickness with pen pressure, although nice in theory, requires a lot of practice and rarely produces the exact result you desire. A better approach is to draw with a constant width in the Calligraphic pen and then apply the Tweak tool's Shrink/Grow mode (12.6.4) to make your strokers thicker or thinner where appropriate, as Figure 25-3 demonstrates.

Figure 25-3: Tweaking the drawing with the Tweak tool

The black and white outline is ready; now let's add color to it. With a crisp ink outline, I wanted a soft washed-out coloring. To do that, hide the background layer, create a new layer called watercolor, and put it below the ink layer. After that, with a wide calligraphic brush, make a few wild strokes: red under the rose, green under the stalk and leaf, and various shades of blue around those. Finally, blur those strokes by a large enough radius so they almost blend together (Figure 25-4).

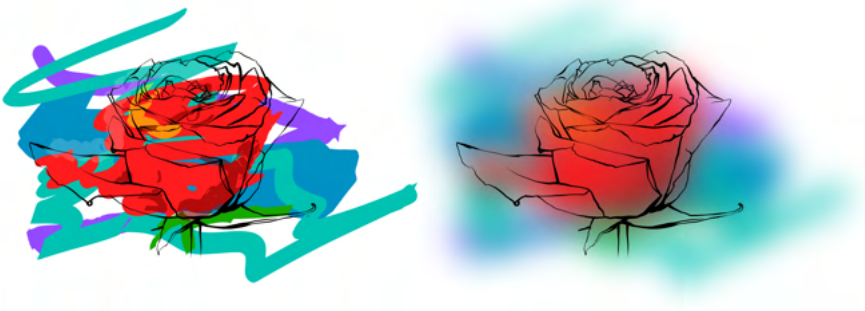


Figure 25-4: Adding a blurred background

This already looks nice, but not nice enough. The smoothly blurred blotches are *too* smooth—and, frankly, boring. To make them more natural, I expanded the simple Gaussian Blur filter applied to the background by adding more primitives to it. First, I added a Turbulence primitive, composited with the blur, to imitate paper grain. This looked better but was still too uniform—too computer-generated. I then added another Turbulence component with a much larger period, which I used twice: once via a Displacement Map, for making the overall watercolor more blotchy and splotchy, and then via a Composite operator, for modulating the small-scale paper grain to imitate paper more realistically, where some areas are smoother and others are more grainy, as shown in Figure 25-5.

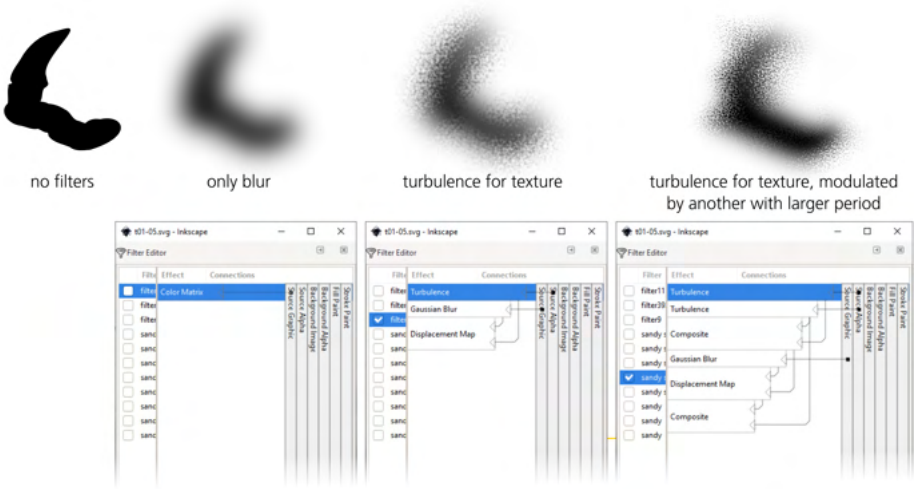


Figure 25-5: Building up the "realistic watercolor" filter for the background

The final composite filter, which I called *Sandy Blur* (see 17.5.2 for more details), looked a lot more realistic but was slower to render. I had to switch to the No Filters mode (17.5.4) so I could work without the annoying rendering delays.

As a last touch, you can duplicate the content of the ink layer and blur it to form a shadow, or (as I did) soften the ink lines slightly by outsetting (thickening) duplicates of the ink paths and making them semitransparent. The rose drawing's background is done.

25.1 Treatment 1: Engraving

With the ink and watercolor layers finished, I went on to create a series of treatments—demos on how to render the same image differently. I overlaid the rose with six randomly scattered squares and limited each treatment to one square.

The first rendition is an engraving made of lattices of curved variable-width pen strokes. For this treatment, I chose an area of the rose where the curving and shading of the petals was especially deep. Engraving is an intricate art; I do not claim that my attempt is any good, but I was surprised by how easy it was to do using Inkscape's tools.

When you have an outline and want to turn it into an engraving, the first step is planning. Think about the most natural directions for the engraving lines in each part of the drawing. Create a new layer and draw several wide-spaced test strokes with the Calligraphic pen, trying to capture the changing curvature of each area while maintaining the best directional contrast at the boundaries where the areas of different curvature meet (Figure 25-6).

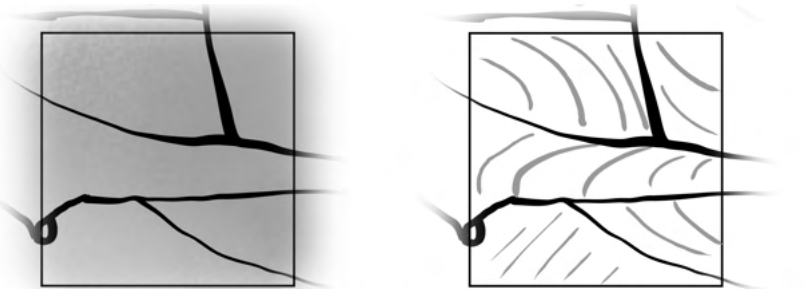


Figure 25-6: Choosing the engraving square and doing test strokes

Once you get an idea of how you want the engraving strokes to go, move the test strokes aside (but keep them for reference) and use the Calligraphic pen's guide tracking feature (14.2.7) to fill each area with evenly spaced uniform-width strokes (Figure 25-7).

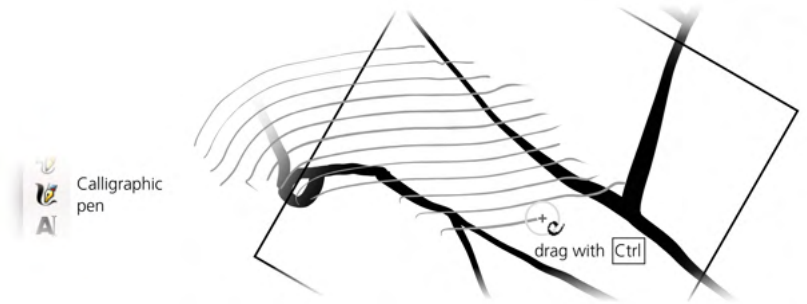


Figure 25-7: Creating an evenly spaced lattice with the Calligraphic pen

Don't worry if the direction or curvature of your strokes drifts off somewhat; you will be able to fix that later. What matters more at this stage is the uniformity of spacing. You can rotate the canvas (Ctrl-Shift-wheel) to make it easier to stroke. Draw beyond the edges of the area you're filling; it's much easier to trim strokes that are too long than it is to extend short ones.

You'll use the Tweak tool to do the rest of the work (Figure 25-8).

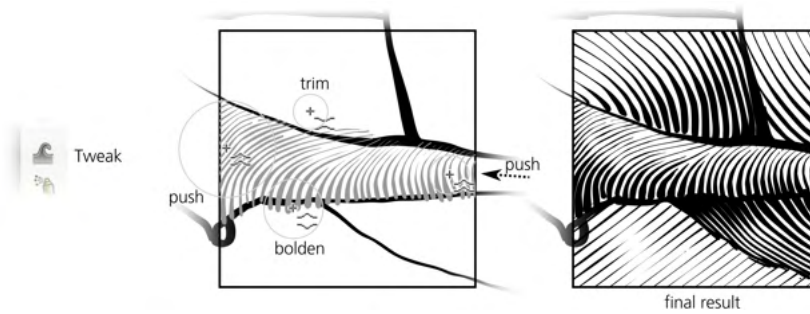


Figure 25-8: Finalizing the engraving with the Tweak tool

Use the Shrink mode to trim line ends and thin them, the Grow mode to thicken, and the Push mode to move and curve the entire lattice (select all of the lattice's strokes to bend them together). Don't worry about the trimmed strokes' unsightly butts. In the final result, the ink layer's wide boundary strokes will cover them.

25.2 Treatment 2: Tessellation

This treatment is easy to do because it's mostly automatic. The only tricky part is creating the tessellation itself—that is, a pattern of complex interlocked tiles covering the entire plane without holes or overlapping.

Draw a shape—any shape—and create a pattern of tiled clones from it (**Edit ▸ Clone ▸ Create Tiled Clones**, 16.6) with any nontrivial symmetry group; I used P3. At first, the clones do not form any kind of tessellation; it is your task to reshape the source shape—with its clones already in place— until the clones meet and smoothly interlock. This step is easier than it might seem. The fact that the clones immediately reflect any change in the source shape makes the task almost trivial. Just add nodes and move them to grow appendages or make depressions in your shape. Each of the clones will mirror those changes. Keep sculpting the shape until the clones precisely interlock and cover the entire plane, as shown in Figure 25-9.

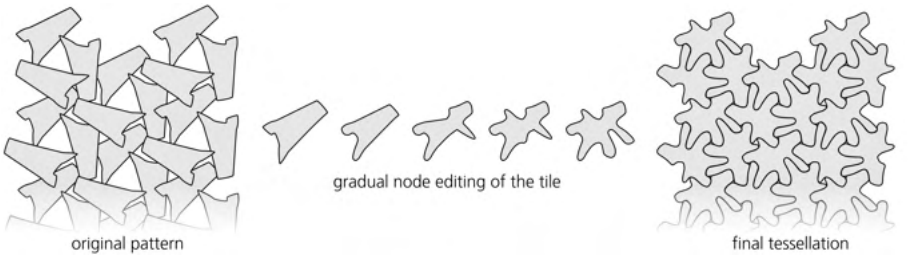


Figure 25-9: Creating a tessellation

Now, let's make the pattern reflect the background colors, so that the rose drawing shows through. Once set up, the process is mostly automatic. Make sure the **Use saved size and position of the tile** checkbox is on, delete the existing tiling (**Remove** button), and set the **Width** and **Height** for the pattern to cover all of the area you want covered. Unset the fill color of the original shape (right-click the **Fill:** swatch in the status bar and choose **Unset**). Then, go to the **Trace** tab, enable **Trace the drawing under the tiles**, pick **Color**, and apply the picked value to the clones' **Color** as well (Figure 25-10).

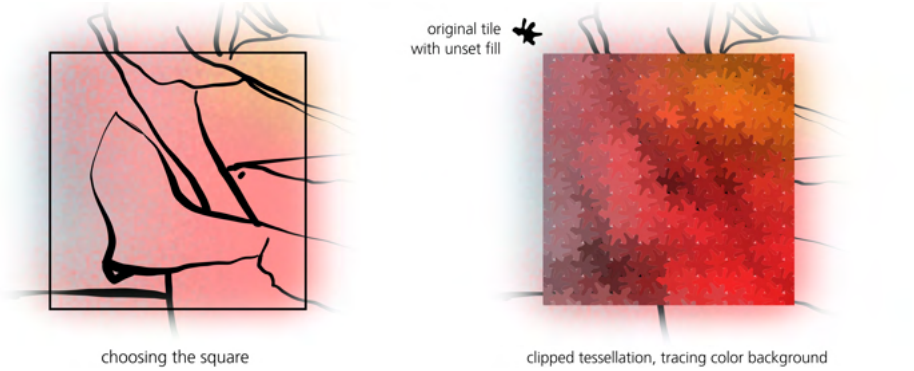


Figure 25-10: Color tracing with the tessellation

As a final touch, let's add slight glossy highlights to the tiles using filters (Chapter 17). One problem with this is that the gloss filter is not symmetric—it has one special direction from which the light is cast. However, if you apply such a filter to rotated *clones* in the pattern, the filter will be rotated together with the clones and all highlights will appear as if lit from different directions. To work around this, simply group all tiled clones (Ctrl-G) and apply the **Button** filter from **Filters ▸ Bevels** to the group, as shown in Figure 25-11.

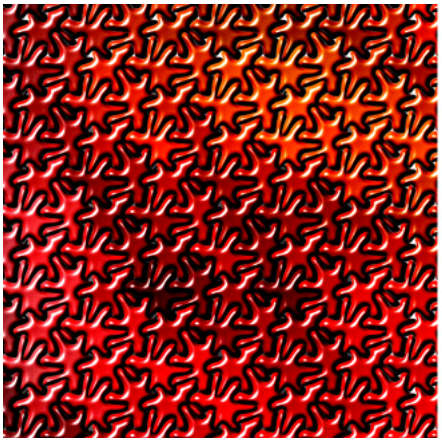


Figure 25-11: Adding highlights

25.3 Treatment 3: A Field of Cubes

Like the color-tracing tessellation, this treatment uses the Create Tiled Clones dialog. In other respects, however, this design is the direct opposite. It's randomized instead of regular, pseudo-3D instead of flat, and uses three different original objects—and thus three intermingled patterns—instead of a single one.

Start by drawing a cube with the 3D Box tool. Convert it into a group of paths (Shift-Ctrl-C) and duplicate it (Ctrl-D). In one copy of the cube, enter it as the group, select all six sides and union them (Ctrl-+), then ungroup. This turns a cube into a single cube-shaped path. Unset its fill—in the clones, this one will take the color of the background. For the second copy of the cube, use 50 percent opacity and either white or black on each of the sides without unioning. You can also add an ellipse with a white-to-transparent elliptic gradient on the foremost corner of the cube as a highlight (Figure 25-12, left). This is the shading layer that makes each box look like a box despite a change in overall color.

Finally, position the shading cube exactly on top of the background shape and group them together. This is your cloneable object that can both pick the background color with its unset-fill shape and present a pseudo-3D appearance with the semitransparent overlay. Repeat all these steps for two more differently oriented cubes, so there are three overall (Figure 25-12, right).

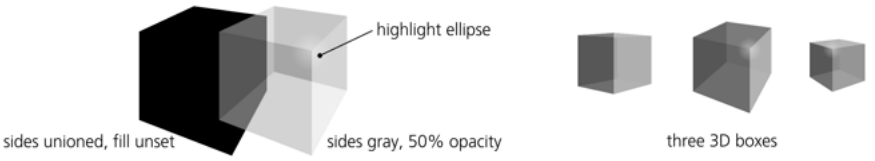


Figure 25-12: Original objects for the cube scattering

Now, select a prepared cube and create a P1-symmetry scattering over the area you want to cover, using color-to-color tracing in the Create Tiled Clones dialog (16.6.6). Randomize their rotation (slightly, so as not to destroy the perception of all boxes being in a common perspective), their size, and their position. To imitate the effect of a 3D field, set scaling and row spacing to increase per row from top to bottom, as shown in Figure 25-13.

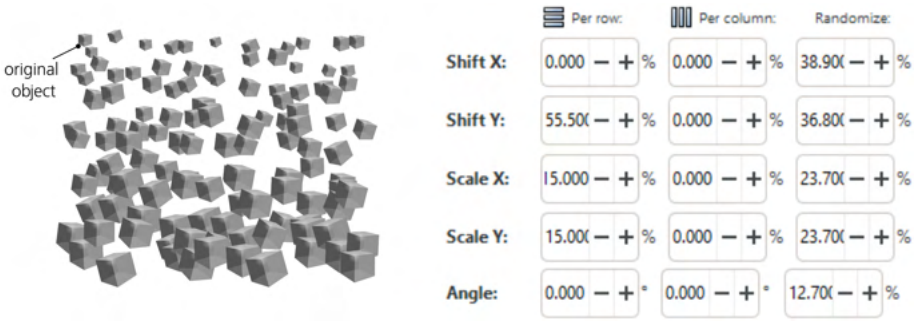


Figure 25-13: Scattering the cubes

Repeat this procedure for the two other cubes you've prepared, covering the same area. This combined field of cubes doesn't yet look like a *field*, because the z-order is wrong. The cubes that are closer to the bottom and larger are supposed to be closer to us and therefore on top of the others, but they are not.

To fix this, use the **Restack** extension (from **Extensions ▸ Arrange**) to sort the z-order of the boxes so they stack from top to bottom. Then, do a single **Unclump** action from the **Align and Distribute** dialog so the arrangement of cubes has fewer gaps (Figure 25-14).

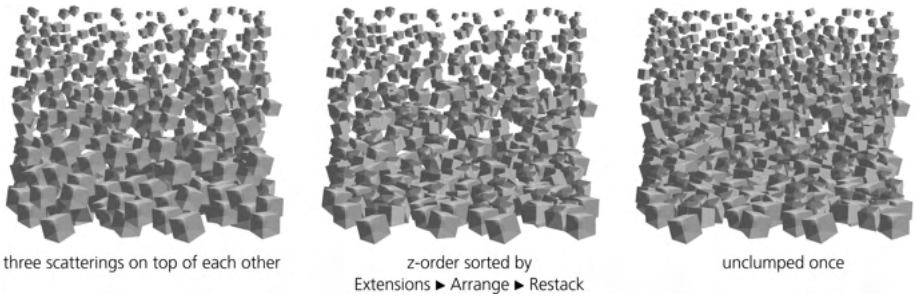


Figure 25-14: The field of cubes: z-order sorting and unclumping

The goal is to reproduce a square area of the rose's background layer. However, if you use automatic color-to-color background tracing (as done in 16.6.6) for all three patterns placed over the target area, you may find the colors resulting from that tracing are quite drab. That is because each cube samples color from a relatively large area that it covers, averages it, and makes it less saturated than the original background image. Also, the 3D shading overlay in each cube washes out the color of the background shape.

To fix this, you can simply paint the cubes manually, as shown in Figure 25-15.

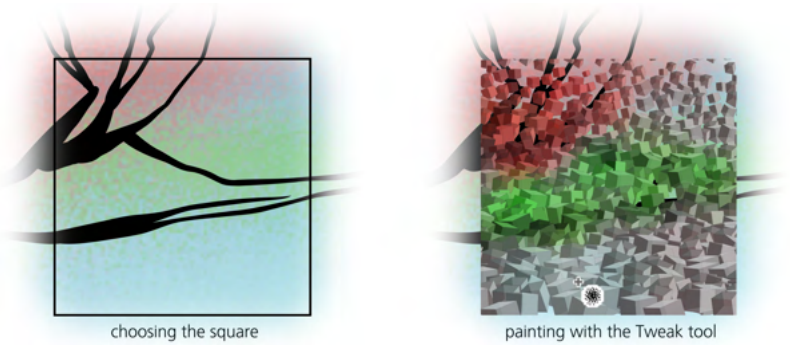


Figure 25-15: Painting colors on the cubes

Just select all the cube clones, choose the Tweak tool's Color Paint mode (8.9), choose a bright red color (or pick it from the background image with the Dropper, 8.8), and paint over the image's red area. Then, do the same for the other colored areas, including the black ink lines. (If it doesn't work, you may have forgotten to unset fill in the bottom path in each cube, see Figure 25-12.) Such a thoroughly bitmap-like paint job is a fun way to edit a vector drawing!

Finally, group all the cubes and clip (18.3) the group to the chosen square.

25.4 Treatment 4: Photorealistic Drawing

This photorealistic rendition is the most time-consuming of all. I wanted to re-create the original photograph as closely as possible in this part of the drawing. Absolute photorealism, of course, is neither possible nor desirable, but you can strive for the characteristic "vector photorealism" look that is attractive in its own way. The key here is a smart use of shapes, gradients, and blurs.

First, locate the square in which you're going to work and reveal (but leave locked) the layer that has the source photo. Start by dividing your drawing into areas, each approximated by flat color or a single gradient. Use the Pencil tool (14.1.2) to create these areas and the Gradient tool (10.1) to paint and stretch gradients across them, as shown in Figure 25-16. Use the Dropper tool (8.8) to pick the exact colors from the photo layer.

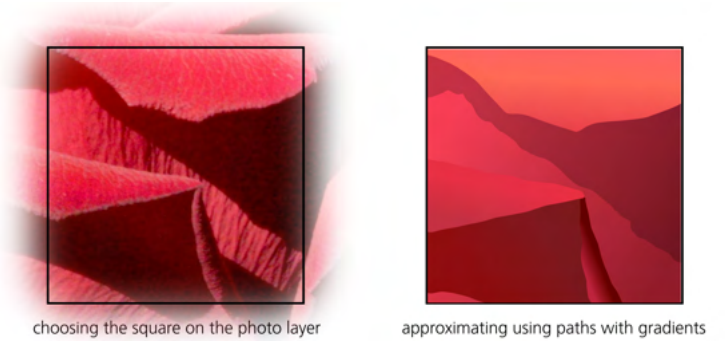


Figure 25-16: Approximating areas of the image using paths with gradients

To soften the edges, blur the shapes a bit (no more than 2 percent) using the **Fill and Stroke** dialog. To apply some naturalistic texture, draw a 40 percent gray, 10 percent opaque rectangle over the shapes and apply **Filters ▸ Overlays ▸ Speckle** to it. In the **Filter Editor** dialog, increase the frequency of the random texture (**Base Frequency** in the **Turbulence** primitive), as Figure 25-17 demonstrates.

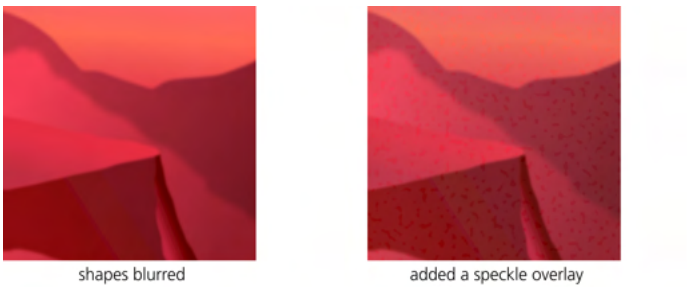


Figure 25-17: Blurring and texturizing the background

What exists so far is just a layer of rough background paint, upon which you'll overlay the fine details—reproducing the characteristic wrinkles and texture of the rose petals. You'll primarily use the Calligraphic pen (14.2) with relatively small Width, zero Fixation, and some Tremor. Typically you would draw with semi-transparent black or white for shades or highlights, respectively. You can also try barely saturated versions of complimentary colors (for example, pale green for red areas) that can often make your drawing look more natural—in real life, color tints are never perfectly uniform.

Of course, you'll need to blur these semitransparent brush strokes as well. Don't apply blur to each stroke separately—this would be tedious to do (unlike opacity, blur is not retained in the tool's style and therefore not applied to the next created object automatically) and slower to render. Instead, draw a single stroke, group it (Ctrl-G), blur the group, and then enter the group (Ctrl-Enter). Now, each new object you create with Calligraphic pen or any other tool gets added to the blurred group and becomes blurred along with its siblings (Figure 25-18). When finished drawing within the group, press Ctrl-Backspace to leave it.

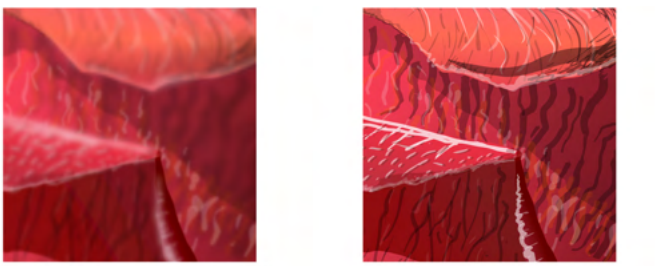


Figure 25-18: Adding hand-drawn foreground strokes

Pay special attention to the edges of the objects on your drawing—properly emphasizing edges with dense highlights and shadows is the key to an attractive "painterly" look. Making the edges of highlights and shadows slightly more intense often works well.

25.5 Treatment 5: Map

The map treatment is less an example of a practically useful technique than it is a creative reinterpretation of something plain and straightforward. I was just playing with an eight-color Potrace tracing (Path ▸ Trace Bitmap, 18.5.2) of this area of the original photo when I noticed that it acquired an interesting map-like look when I discarded the fill colors and assigned a dashed stroke to the resulting shapes, as shown in Figure 25-19.

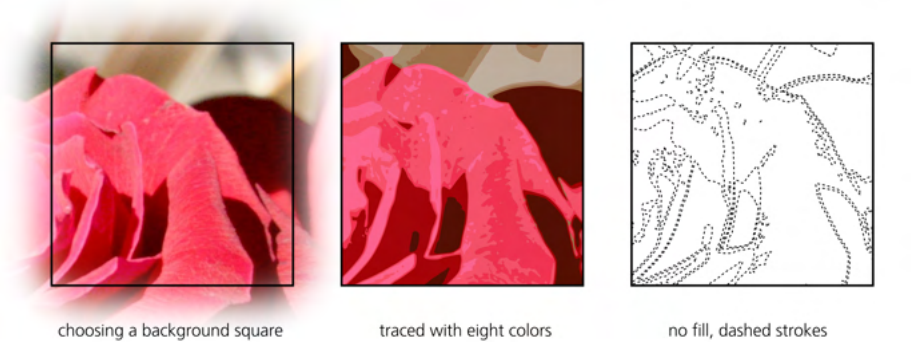


Figure 25-19: Tracing the source photo

To imitate the surface of an old map, I created a yellowish, slightly rough (Filters ▸ Distort ▸ Roughen), semitransparent (so that the background drawing showed through a bit) background rectangle. Then I added some cryptic text labels to complete the picture.

25.6 Treatment 6: Spruced-Up Photo

For the spruced-up photo treatment, I wanted to showcase Inkscape's abilities as a tool for making bitmaps look better, rather than as a vector editor. Of course "better" is a subjective term, but you can make a number of simple edits in Inkscape to improve almost any photo's appearance. Figure 18 in the color insert shows a fragment of the original rose photo with colors brightened, shadows deepened, highlights added, and textures empasized.

As a final touch for the combined rose image, I added dotted frames to all six squares containing various treatments. Figure 17 in the color insert shows the complete image.

Whenever I import a photo into Inkscape, I usually make one or more of the following changes:

- I apply Filters ▸ Color ▸ Lightness-Contrast to expand the photo's dynamic range. In a bitmap editor such as Photoshop or GIMP, I would use the Levels command for this, but in Inkscape, the Lightness-Contrast filter is the closest approximation. The idea is to stretch all the colors of the photo apart so that its darkest area becomes maximally dark (that is, black) and its lightest area becomes maximally light (that is, white). If performed correctly, no image details are lost in this operation, but the photo's visual appearance improves, often considerably. I typically enable the Live preview checkbox in the Lightness-Contrast dialog and tweak the two values, trying to contrast without losing either shades or highlights.
- Lightness-Contrast range expansion works on the entire image indiscriminately. Usually, however, I can see that some areas would benefit from manual lightening or darkening. My favorite way of doing that is to create a white circle (11.4), paint it with a white-to-transparent elliptic gradient (10.1), and assign the Overlay blend mode (17.2) to it. I duplicate it, move it to where I need a highlight, and scale to taste; if a particular highlight seems too strong,

I tone down its opacity. For emphasizing a dark detail, I create a similar shader object with black instead of white. Sometimes, I also use other colors of highlights for color accents. That's what I did on Figure 18 in the color insert (bottom) to make the shading of the petals more expressive; the extents of the overlays are shown with the black and white outlines.

- Almost always, an image that is part of a design composition needs to have its saturation adjusted. Some images—such as our rose—look best when highly saturated. Most of the time, however, a photo may benefit from being fully or partially desaturated. For this, I use the Filters ▸ Color ▸ Color Shift filter that has two sliders for hue and saturation.
- I often improve low-resolution photos by using the Filters ▸ Image Effects ▸ Sharpen or Sharpen More filters.

26

TUTORIAL: ARTWORK FOR A GAME

Games for all kinds of hardware—desktops, gaming consoles, mobile devices—are a huge, always-growing, and intensely graphics-hungry industry. Much of game graphics these days is in 3D, but plenty of work for traditional 2D graphic editors like Inkscape remains—if only as creators of graphic assets for 3D worlds. In this tutorial, I show how Inkscape can be a game developer's Swiss Army knife.

This tutorial's examples are from two games that I, with the help of my family, created for Android mobile devices. They are simple 2D games with arcade-like gameplay based on physical simulation (using the Box2D library). Creating a game is a complex process that includes design, programming, artwork creation, and music, followed by seemingly endless testing, tweaking, and bugfixing—not to mention marketing and promotion. We used many different tools and resources to create our games. Still, I would rank Inkscape at the top of the list, right after the Android Studio IDE by Google. This tutorial is a collection of several miniprojects that demonstrate the use of Inkscape for various game development tasks.

26.1 BotP: Ice Rink

One of the games we developed, called *Battle of the Pucks* (*BotP*, available at *http://kirsanov.com/battle-of-the-pucks/*), features two teams of pucks on an ice rink. Players propel their pucks with a finger swipe, aiming to drive their opponent's pucks into the void beyond the edge of the rink.

Creating the image for the ice rink is a challenge because it must give an impression of a real ice with its dents, scratches, and skate traces. Figure 26-1 shows how we did it using multiple semitransparent paths with Roughen path effect. The final image is exported as PNG and loaded into the game as a graphic asset.

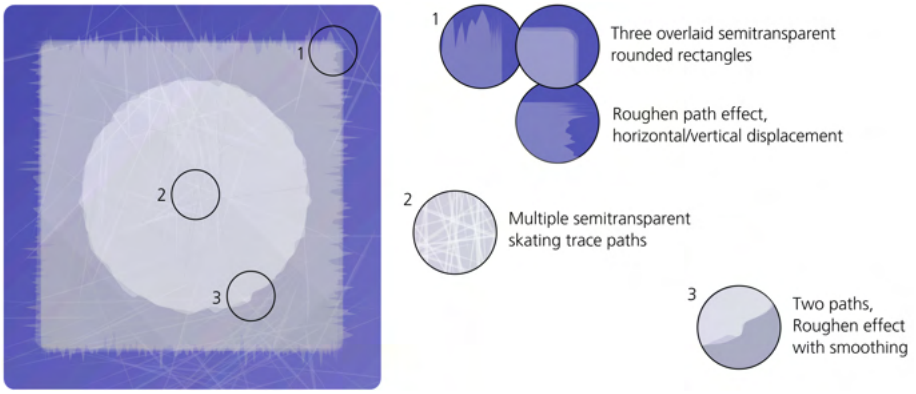


Figure 26-1: Making a realistic ice surface

26.2 BotP: Inkscape as a Level Editor

Battle of the Pucks has multiple levels with a varying initial placement of the pucks (and, on advanced levels, various obstacles) on the game board. I could define those levels in code, but with more than 90 levels, doing so would be tedious work. Generally, when you have visual information, you should use a visual tool to work with it, even if that involves some additional investment up front. That's how we came to the idea of using Inkscape as a level editor for our game.

Any game with levels needs some kind of a level editor, and for complex game projects, it may be a custom piece of software developed in parallel with the game itself. For relatively simple games, such as *BotP*, it makes a perfect sense to use Inkscape for this task. Levels are drawn as images, each on its own layer, and saved as SVG. The game code loads and parses the SVG file, finds the current layer element in it, and reads its child elements to place the pucks and obstacles at the start of each level.

As you might expect, this attractively simple idea was not exactly as straightforward to implement. The Java library we used for the game had an XML parser, but it had nothing to deal with SVG-specific data such as transforms (A.7). As a result, we had to observe certain limitations when creating and editing the *levels.svg* file for the game.

26.2.1 Pucks

Pucks in our game can be of different sizes, but all of them are round, so in *levels.svg*, they are represented by `circle` elements created by the Ellipse tool (11.4). We had to be careful to keep circles circles. If you accidentally squeeze or stretch a circle, Inkscape turns it into an `ellipse` element in SVG; if you touch the round arc/segment handles of the shape (11.4.2), it becomes a path. Both of these changes would break our game's parsing code, which recognizes only a `circle` element and uses its `cx` and `cy` attributes to place the puck and the `r` attribute to determine its size (and mass). The white or black color of the circle (parsed out of its `style` attribute) determines to which of the two commands this puck belongs (Figure 26-2).

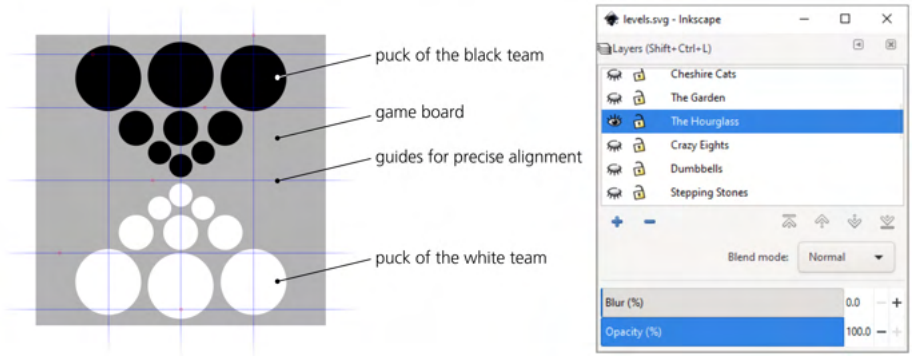


Figure 26-2: A BotP level called "The Hourglass": the two puck armies are ready to battle.

When the puck circle is moved around in Inkscape, it is important to ensure that the circle element doesn't get the `transform` attribute. In Preferences, on the Behavior ▸ Transforms page, the Store Transformations must be set to Optimized. Now, when you move the circle, Inkscape updates its `cx` and `cy` attributes without creating a `transform` attribute. This approach has nonobvious limitations, though: when you rotate or flip a circle, it still gets a `transform` even with optimized transformations. We had learned we needed to arrange our puck armies without flipping but only by moving or scaling.

26.2.2 Obstacles

Advanced game levels contain obstacles—walls that reflect the pucks—making the gameplay more interesting and often quite challenging. In *levels.svg*, obstacles are represented as `path` elements with a red stroke color. The game code can parse the path data in such an element and re-create it as an object in the game, but it has a limitation: it can only create obstacles out of linear fragments. To create curvilinear obstacles, we had to approximate them with short linear segments, using the Extensions ▸ Modify Path ▸ Flatten Beziers extension, as Figure 26-3 demonstrates.

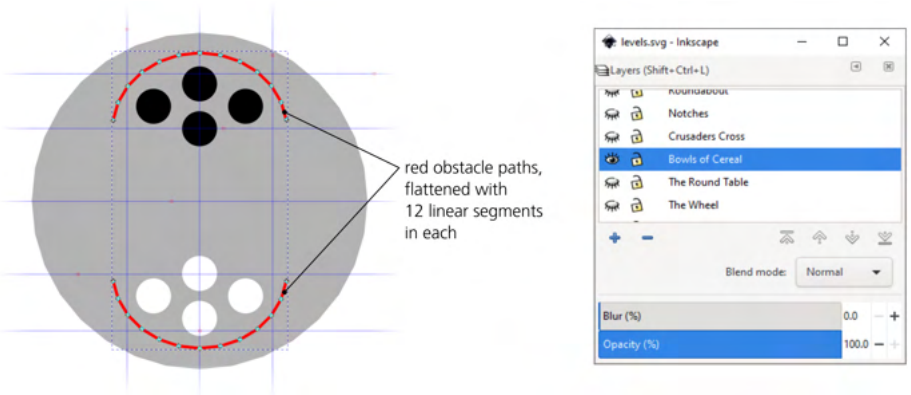


Figure 26-3: A BotP level called "Bowls of Cereal" where four pucks on each side have their backs against the bowl-shaped walls

Figure 26-4 shows how this level is rendered.

Figure 26-4: The "Bowls of Cereal" level in the actual game

When the game code reads the SVG file, it extracts the coordinates of the path's points from the `d` attribute. That data consists of a sequence of one-letter commands (such as `l` for linear segment or `c` for a Bézier curve) followed by their numeric coordinates. Normally, the coordinates in path data are relative; each pair of coordinates provides the displacement relative to the previous point. In our game code, we found it was much simpler to deal with absolute coordinates. For this, in the Input/Output ▸ SVG Output page of the Preferences dialog, we switched path string format to Absolute. The absolute variants of path commands use uppercase letters, so after the change you will see `L`, `C`, and so on in a `d` attribute of a path.

26.2.3 Shrinking the File

Mobile apps are very sensitive to file size, so before including *levels.svg* into the game's assets, we tried to reduce its size as much as possible. Inkscape offers a couple out-of-the-box ways to do this.

- Input/Output ▸ SVG Output page of the Preferences dialog has a number of options that make the saved SVG files smaller: inline attributes (that is, all attributes of each element are placed on the same line), number of indentation spaces, and the precision (number of digits) of numeric values. This affects all SVG files saved from Inkscape.
- The Optimized SVG output file type that you can select in Save, Save As, or Save a Copy dialogs can perform some additional optimizations. It can shrink the numeric data by rounding values to a given number of digits, shorten color values, convert CSS properties to XML attributes, remove unneeded grouping, remove comments and metadata, remove unused IDs, embed raster images, and pretty-print SVG.

For our purposes, however, this was not quite enough. We wrote our own Python script that removes comments, unneeded namespaces, attributes, and style properties from the SVG file. This script also alerts us if, despite all precautions, some circles have a `transform` attribute or some path data (`d` attribute) uses lowercase commands (relative) instead of uppercase (absolute).

26.3 BotP: The Pucks

The pucks in *BotP* can have different colors, depending on the team to which they belong and the chosen color theme. Therefore, each puck object in the game is composed of three layered images: a ring-shaped bottom shadow, then a colored circle, and then a semitransparent overlay that adds a circular ridge and some light color accents and scratches for realism (Figure 26-5).

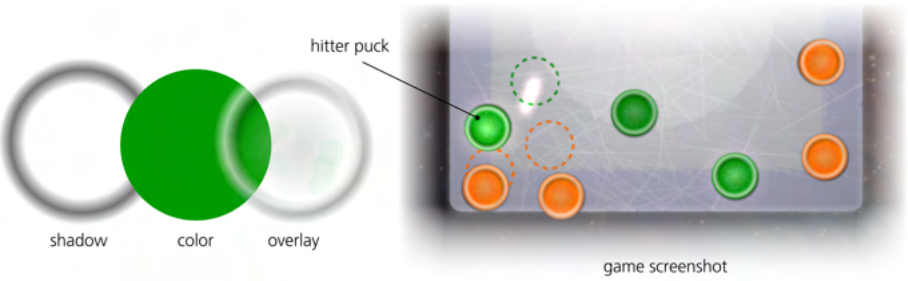


Figure 26-5: Anatomy of a puck and how it looks in the game

Interestingly, although the top ridge has a presumed direction of lighting (from the top left), in the game, the pucks—including the overlays—rotate freely, but this rotation never produces an impression of miscast shadows.

26.4 BotP: Splash Animation

As an intentional contrast to the realistic style of the ice rink and pucks, the splash animation (which plays while the game is loading) and the main menu feature freehand, untidy, jittery animation. The secret to producing such animation is simple: draw every frame anew from scratch, including the parts that do not really move from frame to frame. We used the Calligraphic pen (14.2) with Tremor and/or Wiggle and tried not to be too tidy or stick too closely to the guidelines.

Figure 26-6 shows the 10 frames of *BotP*'s splash animation, featuring an anthropomorphized puck streaking across the ice in a frenzy. While each of these images on its own seems just badly drawn and random, when shown in quick succession, they merge into a lifelike and even relatable character.

Figure 26-6: The frames of the animated splash image

26.5 Batonic: An Iconic Character

Another game we created is called *Batonic* (*http://kirsanov.com/batonic/*). What's interesting about this game is the title character's design, which is used as the game's icon as well as on the cover of this book.

Figure 26-7 shows how it looks when sliced into its component layered objects. We applied the Envelope Deformation path effect to a path of rectangular stripes to create the striped overalls and then clipped the result by a copy of the blue garment path.

Figure 26-7: Layers of Mr Bubbel

In the game itself, this amusing gentleman, named Mr Bubbel, spawns a whole army of Bubbels that are falling out of the sky upon a circus baton that you balance on the tip of your finger. Here, each character's image is separated into body, eyes, arms, and legs, which are animated to make the Bubbels come alive, as shown in Figure 26-8.

Figure 26-8: Batonic*: Bubbels attack your baton.*

A

AN SVG PRIMER

This appendix is a very high-level glance at the workings of SVG and its implementation in Inkscape. If you want the final, complete, and authoritative reference on SVG, refer to the W3C's SVG specification at *https://w3.org/Graphics/SVG/*. As of this writing, SVG 2.0 is still a draft; Inkscape supports the latest released version of the standard, 1.1, that you can find at *https://w3.org/TR/SVG11/*.

Knowledge of SVG and, more generally, XML (SVG is based on XML) is not strictly necessary for mastering Inkscape—but it helps. It will allow you to peek under the hood of Inkscape drawings, and it will often reveal the true reasons behind some of Inkscape's features (or lack thereof). Let's start with a quick intro to XML that you may find useful in many situations, even those that don't involve Inkscape or SVG.

A.1 A Quick Introduction to XML

If you've ever looked at a web page's source code, you already know what XML looks like, because XHTML, the markup language used on the web today, is one of the subspecies of XML—more precisely, an *XML vocabulary*. SVG is another such vocabulary.

XML is a standardized way to record structured information in plaintext. It is easy for computers to parse and yet quite understandable for humans. Unlike most computer-related concepts that tend to be as complex as you think they are (or more), XML is almost unbelievably simple.

The basic building block of an XML document is an *element*. Here is an example of an element containing some text:

```
<example>Here goes some text.</example>
```

The text inside the element is called its *content*, and that content is delimited by *tags*. Everything between the less-than sign (`<`) and the greater-than sign (`>`) is a tag. Here, the opening tag and the closing tag are almost identical, except the closing one has a forward slash (`/`) before the element name, which in this case is `example`.

An element may have no content at all:

```
<example></example>
```

or

```
<example/>
```

These two *empty elements* are equivalent; the second one, consisting of a single tag, is just a spelling variant of the first. Note the different position of the forward slash (`/`) in the single-tag empty element.

In addition to text, elements can also contain other elements, but those elements must lie *entirely* within the containing element. For example, this is wrong:

```
<a><b></a></b>
```

because the element `b` starts inside `a` but ends outside it. If an element starts inside some other element, it must also end within that element. Here is an example of correct XML:

```
<a><b/><c/><d><e/></d></a>
```

We say that `a` is the *parent element* of `b`, `c`, and `d`, while `d` is the parent of `e`. An element may have multiple children, but it has only one parent (except for the root element of the document, which has no parent at all). Figure A-1 is a graphic representation of this XML fragment.

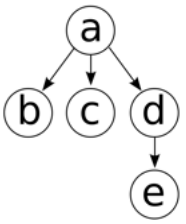


Figure A-1: A tree representing the XML code

NOTE *This explains why groups of objects in Inkscape cannot contain objects from different layers (4.8). A group is a parent element for its children—but so is a layer. A group may be contained inside a single layer, but it can't be scattered across several layers, because that would mean it has several parents.*

Apart from child elements, XML elements may have *attributes*. Each attribute is a name with some associated value. Attributes are specified in the opening tag of an element:

```
<text type="important" font-size="10">Here is some text.</text>
```

Note the equal sign (=) and the double-quotation marks around attribute values. Both are mandatory.

NOTE *An element cannot have two attributes with the same name. The order of the attributes in the opening tag doesn't affect anything: attributes in XML are unordered.*

An entire XML document is just a single element (called the *root*), possibly with some text content and child elements—which, in turn, can have more content and more children, and so on. An XML document can thus be thought of as a tree growing from a single root. For example, the following is a complete and valid SVG document whose root element, `svg`, contains two elements representing a rectangle and a text string:

```
<svg>
  <rect x="100" y="100" width="300" height="50" fill="blue"/>
  <text x="100" y="150">This is a text string.</text>
</svg>
```

A.2 Vocabularies and Namespaces

XML itself is just a foundation—and you can erect a lot of different buildings on that foundation. You won't believe how many work hours were wasted, before the advent of XML, on inventing new data formats for each application, formulating their rules, writing and testing software to support them, and working around the inevitable bugs, exceptions, and version incompatibilities.

These days, if you need to record some data, you no longer need to go into the low-level details of which characters have special meanings and what the sizes of various fields are. By choosing XML (and you need serious reasons *not* to choose it), you can concentrate on what really matters: the structure of your data and how to name elements and attributes to best express that structure.

Moreover, you may not even need to do that, because a common standardized XML *vocabulary* likely already exists for your kind of data. Each vocabulary defines a set of elements and attributes, detailing what they are supposed to mean, in what contexts they may be used, what is obligatory or optional, and so on. SVG is one such vocabulary; XHTML is another. Standardized XML vocabularies exist for many exotic things, from dog genealogies to star catalogs. Many of them reuse parts of other vocabularies.

To be able to mix different vocabularies without confusion, XML uses *namespaces*. A namespace is a way to indicate where a specific element or attribute comes from. For example, here's an `image` element from the SVG vocabulary, representing a linked bitmap image inside an SVG document:

```
<svg:image xlink:href="/dir/file.png" width="200" height="100"/>
```

Note that the element name and one of the attributes use a colon (`:`). The part of the name before the colon is called the *namespace prefix*. To find out which namespace corresponds to that prefix, look for a namespace declaration in the form of a special `xmlns`-prefixed attribute (`xmlns` is short for *XML NameSpace*), either on the same element or on any of its ancestor elements up the tree.

It is customary to place all namespace declaration attributes on the root element, so you will likely find the declarations for `svg` and `xlink` prefixes in the root `svg` element:

```
<svg:svg ...
         xmlns:svg="http://www.w3.org/2000/svg"
         xmlns:xlink="http://www.w3.org/1999/xlink"
         ... >
```

This means that the `svg` prefix is bound to the namespace of `http://www.w3.org/2000/svg`, and the `xlink` prefix corresponds to `http://www.w3.org/1999/xlink`. Here, the namespaces are URLs, but they don't need to be; the only requirement is that they are globally unique, and URLs are the easiest way to ensure that.

The namespace URL for SVG, `http://www.w3.org/2000/svg`, must look exactly as shown. If the namespace prefix of the elements in your SVG document does not resolve to `http://www.w3.org/2000/svg`, Inkscape (or any other SVG software) will not recognize the document as SVG. The namespace *prefix*, however, may be arbitrary, so long as it's bound to the correct namespace. The prefix can even be empty if you declare it thus:

```
<svg ...
     xmlns="http://www.w3.org/2000/svg"
     ... >
```

Inside the element with that declaration, any element name *without* a prefix is assumed to be in the SVG namespace. For example, the following will be recognized as a valid SVG `image` element that links an external image file in PNG format:

```
<image xlink:href="/dir/portrait.png" width="200" height="100"/>
```

Namespace errors, such as no namespace declared for a prefix or a wrong namespace URL, are the most common when you edit SVG by hand—watch out for them!

A.3 Root

Here's a typical root svg element of an Inkscape SVG document with a bunch of namespace declarations and other attributes:

```
<svg
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:cc="http://creativecommons.org/ns#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:svg="http://www.w3.org/2000/svg"
   xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:sodipodi="http://sodipodi.sourceforge.net/DTD/sodipodi-0.dtd"
   xmlns:inkscape="http://www.inkscape.org/namespaces/inkscape"
   width="385.6"
   height="210"
   id="svg2"
   sodipodi:version="0.32"
   inkscape:version="1.0 (4035a4fb49, 2020-05-01)"
   version="1.1"
   sodipodi:docname="file.svg"
>
```

SVG uses the SVG namespace (`http://www.w3.org/2000/svg`) for its own elements and the XLink namespace (`http://www.w3.org/1999/xlink`) for the linking attributes (A.9). On top of that, Inkscape adds elements and attributes in the namespaces belonging to Inkscape (`http://www.inkscape.org/namespaces/inkscape`) and its predecessor Sodipodi (`http://sodipodi.sourceforge.net/DTD/sodipodi-0.dtd`; see 1.6 for a brief history of Inkscape). Other namespaces declared here are used for metadata elements (general information about the document) and license identifiers (A.4).

In this example root element, the only attributes defined in the SVG standard are `width` and `height` (defining the size of the canvas), `id`, and `version`. The rest are either namespace declarations or attributes in the Inkscape/Sodipodi namespaces.

The `inkscape:version` specifies the version of Inkscape in which this document was created, in this case 1.0 (with a build number and date in parentheses). It also claims that the version of Sodipodi that this document will work with is 0.32 (`sodipodi:version="0.32"`), because after that version Inkscape forked off Sodipodi. The non-namespaced `version` attribute refers to the version of the SVG specification this document implements, 1.1.

When you save a document, you have a choice between Inkscape SVG and Plain SVG formats; the only difference is that Plain SVG strips away all the elements and attributes in the Inkscape and Sodipodi namespaces, leaving only the standard namespaces. Generally, no Inkscape-specific elements or attributes are allowed to alter how the document *looks*; they can affect only the way it *behaves* when edited in Inkscape (for example, an attribute on a `g` element may tell Inkscape to treat that element as a 3D box instead of a simple group; see 11.3). Thus, saving as a Plain SVG will make a file a little smaller but at the cost of losing some of its Inkscape editability. If you find a file that renders differently after being saved as Plain SVG, you've found a bug—please report it to the developers!

A.4 Defs, View, and Metadata

Let's descend from the root element of a typical Inkscape SVG document to see what else is in there.

First comes the `defs` element. A part of the SVG standard, `defs` is a storage bin for things that, by themselves, are not displayed on the canvas but may be referred to by other elements. This includes gradients, patterns, markers (arrowheads, 9.5), clipping paths, masks, filters, and so on, which are all stored as child elements of the `defs` element. The SVG specification allows you to place multiple `defs` elements almost anywhere in the document, but Inkscape always uses a single `defs` under the root `svg`. Here's an example `defs` element containing a gradient:

```
<defs id="defs4">
  <linearGradient inkscape:collect="always" id="linearGradient2798">
    <stop style="stop-color:#000000;stop-opacity:1;" offset="0" id="stop2800"/>
    <stop style="stop-color:#000000;stop-opacity:0;" offset="1" id="stop2802"/>
  </linearGradient>
</defs>
```

Adding the `linearGradient` element to `defs`, by itself, does not change anything on the canvas. Now, however, any object can reference (10.2.1) this gradient for painting its fill or stroke.

Most of the stuff in `defs` is kept even if no visible element uses it. Exceptions are the elements with the `inkscape:collect="always"` attribute; such elements are automatically deleted when no longer used. Inkscape often uses this flag to mark various supplemental elements it creates on its own and can safely dispose of; it will never delete anything that you have created or explicitly added without your permission. This way, you can reuse things you once defined but then stopped using.

Because of this, `defs` may grow larger and larger as you work on your document. To remove any unused definitions from `defs`, use the **File ▸ Clean Up Document** command. Sometimes, this command may fail to delete *all* unused stuff in `defs` on the first try; you may need to quit, reload the document, and clean it up again.

The next element under the root of an Inkscape SVG document is `sodipodi:namedview`:

```
<sodipodi:namedview inkscape:window-height="950"
                    inkscape:window-width="1280"
                    inkscape:window-x="0"
                    inkscape:window-y="24"
                    inkscape:pageshadow="2"
                    inkscape:pageopacity="0.0"
                    bordercolor="#666666"
                    pagecolor="#ffffff"
                    inkscape:zoom="0.64306408"
                    inkscape:cx="836.6226"
                    inkscape:cy="619.40089"
                    inkscape:current-layer="layer2"
                    showguides="true" />
```

This Inkscape-specific element holds, in its attributes, all kinds of options specific to this document: the zoom level and scroll position that Inkscape will use upon loading the document, the rotation angle of the canvas, the colors and spacing of the grids, various snapping modes, the current layer, document units, and so on. From within Inkscape, you can set most of these options via the File ▸ Document Properties dialog (3.1.2).

Finally, the `metadata` element appears right before the document's content actually starts. It stores information about the document's author, its purpose, date, license, and so on, and it corresponds to the File ▸ Document Metadata dialog in Inkscape. The elements inside `metadata` use the RDF standard (see *https://w3.org/RDF/*), which may employ additional XML namespaces to describe different sorts of information.

A.5 Layers and Groups

After the `metadata` element, the SVG document's actual visible content begins. Naturally, each document *object* corresponds to an SVG *element*. Moreover, the order of the elements in the file corresponds to the order in which the objects are visually stacked on the canvas (4.4): elements closer to the end of the file are painted on top of those at the beginning.

When you group objects together (4.8), they are placed inside a `g` element. For example, here's a group that contains a rectangle and a text object:

```
<g id="g2893">
  <rect id="rect2310"
        y="350"
        x="100"
        width="300"
        height="150"
        style="opacity:1;fill:#ff0000;stroke:#000000;stroke-width:1px;" />
  <text id="text2889"
        y="400"
        x="200"
        style="font-size:40px;font-style:normal;font-weight:normal;fill:#000000;
               stroke:none;font-family:Bitstream Vera Sans"
        xml:space="preserve">Text in rectangle</text>
</g>
```

Besides groups, another way to organize your objects is via layers (4.9). SVG does not have a special element type for layers; instead, Inkscape uses an SVG `g` element, adding a custom attribute so it knows to treat the group as a layer when the file is edited.

```
<g inkscape:groupmode="layer"
   inkscape:label="Layer 1"
   id="layer1">
  ...contents of the layer...
</g>
```

Since layers are thus a specific type of groups, it is easy to see how Inkscape can "enter a group," treating it temporarily as a layer (4.9.1). This is also why,

upon importing Inkscape's SVG file into another vector editor (such as Adobe Illustrator), you usually lose layers but objects get additional layers of grouping: the other editor knows nothing about Inkscape's convention of using some `g` elements as layers and treats all of them as regular groups.

Inside layers and groups, other SVG elements represent the actual objects of your drawing. See *https://w3.org/TR/SVG11/eltindex.html* for a complete list of elements in SVG 1.1, but note that Inkscape doesn't support all of them yet.

A.6 Coordinates and Units

You can use a number of units to express distances and coordinates in SVG drawings: centimeters (`cm`), millimeters (`mm`), inches (`in`), points (`pt`), and a few others. One unit, however, is special: the *SVG pixel* (abbreviated `px`) is sometimes called the *anonymous unit* because you can write it without any unit designation at all. For example, in this `rect` element, `x`, `y`, `width`, and `height` are expressed in SVG pixels:

```
<rect id="rect2310"
      y="350"
      x="100"
      width="300"
      height="150" />
```

You could also specify these same lengths in millimeters:

```
<rect id="rect2310"
      y="92.604mm"
      x="26.458mm"
      width="79.375mm"
      height="39.688mm" />
```

and Inkscape would understand it correctly. However, if you change and resave this SVG file, Inkscape will convert all sizes back to SVG pixels.

An SVG pixel is not the same as a *screen pixel*. When you zoom in, a distance of one SVG pixel becomes larger than one pixel on your screen; when you zoom out, it becomes smaller than one screen pixel. However, when your zoom level (displayed at the right end of the status bar of your Inkscape window; see Figure 2-2 on page 20) is 100 percent, one SVG pixel exactly corresponds to one screen pixel, which is convenient when creating graphics for screen viewing.

In Inkscape, 96 SVG pixels are equal to one inch, so when you export your image at the resolution of 96 dpi (pixels per inch), it will show up exactly as displayed in Inkscape at 100 percent zoom. In most places where you can specify sizes or distances in Inkscape's UI, there's a way to specify the units, and the `px` unit (SVG pixel) is the default. You can choose a different default as the Display unit for your document in the Document Properties, Page tab.

SVG uses rectangular Cartesian coordinates for specifying objects' positions on canvas. The coordinate origin in SVG is in the top-left corner of the page, and the `y` coordinate grows downward. By default, Inkscape's UI uses the same convention, although you can switch the origin to the bottom-left corner with `y`

growing upward (**Preferences ▸ Interface**, uncheck **Origin at upper left** that is checked by default). In older Inkscape versions, origin at bottom left was the only available option, even though it was always against the SVG convention.

A.7 Transformations

Every object in SVG has its own natural place on the canvas. For example, for a rectangle, the `rect` element's `x` and `y` coordinates define this place:

```
<rect id="rect2310"
      y="350"
      x="100"
      width="300"
      height="150" />
```

However, one of SVG's most interesting features is that this position can be affected by the `transform` attribute. Often, this attribute contains a sequence of six numbers inside a `matrix(...)`:

```
<rect transform="matrix(0.96333,0.26831,-0.26831,0.96333,203.200,-160.066)"
      id="rect2310"
      y="350"
      x="100"
      width="300"
      height="150" />
```

In this form, the attribute represents an *affine transformation matrix*. A treatment of matrix algebra is outside the scope of this book, but here's a list of transformations that are called affine:

- Any *moves*, also called *translations* (Figure 6-1 on page 92).
- Any *scalings*, including both uniform and nonuniform; for example, scaling only width or only height (Figure 6-2 on page 93).
- Any *rotations* around any center (Figure 6-6 on page 95).
- Any *skews*, sometimes called *shears* (Figure 6-7 on page 96).

Not coincidentally, these transformations are exactly those that the Selector tool can perform (Chapter 6). For example, perspective transformations are *not* affine: they cannot be expressed by a `transform` attribute and cannot be performed by the Selector tool.

An element with a `transform` attribute tells Inkscape to draw this element at its natural position and size, and then move, scale, rotate, or skew it as specified in the `transform`.

A `transform` value on an element's parent (for example, on the `g` element that contains this object) affects the object, too. All the transforms on the object and all its ancestors are *combined*. This is why, for example, when you move or scale a group, all objects belonging to the group are moved and scaled by the same amount, although it is only the parent `g` element whose `transform` attribute is modified.

On the Behavior ▸ Transforms page of the Preferences dialog (3.1), there's a Store transformations choice with the values of Optimized and Preserved. This determines the strategy Inkscape uses when transforming objects. With Preserved, it will always record all transformations of all objects as `transform` attributes, leaving all other attributes intact. With Optimized (default), Inkscape will try, whenever possible, to record the transformation into the object's other attributes and not `transform`. For example, when you move a rectangle, in the optimized mode, it will change the rectangle's `x` and `y` attributes instead of adding or changing its `transform`. Not all kinds of transformations and not all types of objects allow for such optimization, however, so even in the optimized mode, `transform` attributes will still be created. The only object type that can optimize any kinds of transformations and make do without the `transform` at all times is `path`.

A.8 Style

Naturally, the style properties of an object can be represented as attributes of the corresponding element. Such attributes are called *presentation attributes*. For example, this rectangle has blue fill and black stroke 1 px wide:

```
<rect id="rect2310"
      y="350"
      x="100"
      width="300"
      height="150"
      fill="blue"
      stroke="black"
      stroke-width="1" />
```

This, however, is only one of the ways to record style properties in XML. Another way is to pack all the properties into a single attribute, called `style`, using semicolons (`;`) to separate properties and colons (`:`) to separate the name from the value in each property:

```
<rect id="rect2310"
      y="350"
      x="100"
      width="300"
      height="150"
      style="fill:blue;stroke:black;stroke-width:1" />
```

Inkscape understands both methods—but when writing properties to SVG, for historical reasons, it uses only the second one, with a single `style` attribute. When both are present, the properties in the `style` attribute take precedence over the same properties in presentation attributes.

This appendix does not list all the style properties SVG uses (for a complete list, see *https://w3.org/TR/SVG11/propidx.html*), but you should be able to recognize the most common ones.

SVG prescribes that most (but not all) style properties can be inherited by children from their parents, provided a child does not specify its own value for that property. For example, if a rectangle has no `fill` property specified but its

parent g has fill="blue", the rectangle will be painted blue. In Inkscape, such inheritance rarely plays a role, because normally, objects have most of their properties explicitly set whether you changed them or not. However, for fill and stroke properties, there's a way to remove, or *unset*, these properties via the UI, making it possible to inherit these properties from the parent element (8.2).

A.9 Linking

Often, elements in SVG need to refer, or *link*, to one another. This is most common when visible elements on the canvas use some of the definitions in defs (A.4). For example, if you have a rectangle filled with a linear gradient, the rect element describes only the rectangle itself. A different element called linearGradient in the document's defs describes its gradient, and the rectangle links to that gradient definition.

In order to be linkable, an element must have an id attribute whose value is unique inside this document. Inkscape provides unique ids for all elements automatically. The URL for linking to an element is simply its id preceded by a hash mark (#)—for example, #linearGradient2128. To use this URL from a style property, you need to enclose it in parentheses and prefix it with the string url. For example, here's a linearGradient element and a rectangle linking to it from its fill property:

```
<defs>
  <linearGradient id="linearGradient2128">
    <stop id="stop2286" offset="0" style="stop-color:#0000ff;stop-opacity:0;" />
    <stop id="stop2284" offset="1" style="stop-color:#0000ff;stop-opacity:1;" />
  </linearGradient>
</defs>

...

<rect id="rect2310"
      y="350"
      x="100"
      width="300"
      height="150"
      style="fill:url(#linearGradient2128)" />
```

Linking is not always done by style properties; for example, the SVG standard says that gradients can link to one another in order to share color stops and other attributes. In such cases, SVG uses the xlink:href attribute (in other words, the href attribute in the XLink namespace; XLink is another W3C standard, separate from SVG, used in many XML vocabularies for linking). The xlink:href attribute uses the plain URL without the url() wrapper—for example, xlink:href="#linearGradient2128".

SVG allows you to link not only to other elements in the same document but also to other documents, accessible locally or on the internet, as well as to elements inside them. Inkscape does not yet support such cross-document linking to SVG documents—although it can link to external bitmap files inserted as bitmap objects in a document (18.2.1).

A.10 Object Types

Inkscape can create various object types, each object remembering its type and providing controls and behaviors specific to that type (Chapter 11). SVG directly supports some of these object types; for example, `rect` elements from SVG represent Inkscape's rectangles. But others are unique to Inkscape—for example, SVG has no special elements for spirals, stars, or 3D boxes. How can Inkscape use these object types while staying compatible with SVG?

The solution is the `sodipodi:type` attribute. Inkscape saves a star as a universal `path` element that can represent any shape, but adds to it the `sodipodi:type` indicating it's actually a star, as well as some other extension attributes storing various star-specific parameters. The `path`'s standard `d` attribute, meanwhile, provides an exact representation of the star's shape:

```
<path sodipodi:type="star"
      style="fill:#ff0000;"
      sodipodi:sides="5"
      sodipodi:cx="342.85715"
      sodipodi:cy="703.79077"
      sodipodi:r1="105.75289"
      sodipodi:r2="40.397606"
      sodipodi:arg1="0.90027477"
      sodipodi:arg2="1.5285933"
      inkscape:flatsided="false"
      inkscape:rounded="0"
      inkscape:randomized="0"
      d="M 408.571,786.647 L 344.561,744.152 L 284.362,791.893
         L 304.997,717.884 L 240.990,675.383 L 317.754,672.139
         L 338.395,598.132 L 365.202,670.135 L 441.965,666.897
         L 381.770,714.642 L 408.571,786.647 z" />
```

When loading a document with such an element, Inkscape recognizes the `sodipodi:type` and, for editing, treats the object as a star rather than a path. When any other SVG software loads the same file, it ignores any Inkscape-specific attributes and interprets this element as a simple path—which, however, *looks* exactly the same as the star displayed in Inkscape. In other words, while only Inkscape can *edit* the star as a star, using the standard `path` as the base for this object type ensures that it remains compatible with any SVG software.

A.11 Inkscape's SVG Extensions

To conclude this necessary but very superficial appendix, I give a list of the most important extension elements and attributes that you may see in Inkscape SVG files.

One point bears repeating: none of these additional elements and attributes make Inkscape documents invalid SVG. The X in "XML" stands for "eXtensible" for a reason: the ability to mix different vocabularies freely was one of XML's goals from the very beginning. This means that any Inkscape SVG file must—and does—render absolutely the same in all compliant SVG renderers.

- The `inkscape:version` attribute on the root element identifies the version of Inkscape in which the file was last edited and saved.
- The `inkscape:collect` attribute is added to those elements in `defs` that can be deleted automatically if no longer used (A.4).

[1.1]

- The `inkscape:isstock` attribute is set on those elements in `defs` that are taken from Inkscape's stock markers, patterns, gradients, and so on, as opposed to those you created yourself. The `inkscape:stockid` attribute provides the `id` that this element had in its stock source.
- The `sodipodi:namedview` element and its attributes are where Inkscape stores per-document preferences (A.4).
- The `sodipodi:type` attribute is what Inkscape uses to mark objects of non-SVG types (A.10). An element with this attribute usually has a bunch of other Inkscape-specific attributes that store various parameters unique to this object type.
- The `inkscape:transform-center-x` and `inkscape:transform-center-y` attributes are set on an object if you have moved its fixed point (6.4). Inkscape remembers the position of this point for each object for which it was changed.
- The `sodipodi:nodetypes` attribute is where Inkscape stores the types of all nodes of a path that you edited with the Node tool. Classifying nodes as smooth, symmetric, and cusp (12.5.5) is something Inkscape allows you to do, but SVG has no provisions for—hence the need for an extension attribute.
- The `sodipodi:role` attribute with the value of `line` is set on those `tspan` elements inside a `text` element that represents a line (15.2.1).
- The `inkscape:groupmode` attribute on a `g` is what differentiates a group from a layer in Inkscape (A.5). For a layer, this attribute stores the value `layer`.
- The `inkscape:label` attribute can be set on any element to provide a human-readable label for the corresponding object (the `id` attribute is bad for this purpose because it cannot contain spaces and punctuation; `inkscape:label` is free from those limitations, 4.1). In this attribute, Inkscape stores the human-readable name of a layer.
- The `inkscape:menu` and `inkscape:menu-tooltip` attributes appear on a `filter` element in `defs` when you apply one of the preset filters (17.4) to an object in your document. They store the submenu from which this filter came and its brief description (which Inkscape uses for a status bar tip when you browse the menus).
- The `inkscape:perspective` element is stored in `defs` when you use the 3D Box tool (11.3); it stores a shared perspective to which multiple 3D box objects may refer.
- A collection of `inkscape:tile-*` attributes appear on an object after you use the clone tiler on it (16.6); they store the tile's saved size and position.
- The `inkscape:export-filename` attribute, which you can set on any element (including root), stores the filename, including the path, into which this object (or, in case of root element, the entire document) was last exported

as a bitmap. This value is automatically placed into the Filename field of the Export PNG Image dialog (18.6.1.4), so you can quickly redo exporting to the same file. (This is why Inkscape claims that the document has changed and needs resaving after you do a bitmap export from it.) Similarly, the `inkscape:export-xdpi` and `inkscape:export-ydpi` attributes contain the horizontal and vertical resolution you last used for exporting this document.

B

IMPORT AND EXPORT

This appendix is a collection of notes on the capabilities, limitations, and prerequisites of the major import and export formats that Inkscape supports.

You need to download and install external software if you want Inkscape to import certain formats. As a rule, if you don't have the necessary external software installed (or if it is not in PATH), Inkscape simply omits this format from the list of supported formats in the Open or Import dialogs.

If Inkscape doesn't support your favorite vector format, check whether an open source utility exists that can bridge the gap between that format and one of the known formats (ideally SVG, but PDF would work too). If such a converter exists, adding support for a new format to Inkscape is a matter of writing an input or output extension (Chapter 19).

B.1 Save vs. Export

A note on terminology: Inkscape uses the term *export* only for the Export PNG Image dialog (18.6.1). The vector formats it supports, including the default Inkscape SVG format, are listed in the Save as type list in the Save, Save As, and Save a Copy dialogs. In other words, you don't *export* to these formats, you *save*

to them. Since all vector formats except Inkscape SVG lose some editability features or even drop some classes of objects, saving in such a format and then trying to close the document displays a warning, as shown in Figure B-1.

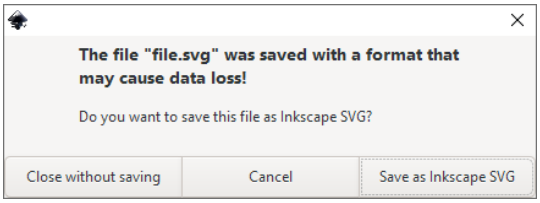


Figure B-1: Careful! You are closing a document last saved as something other than Inkscape SVG!

If you make any changes after saving a non-SVG file, you may have to Save As the original SVG file again. To avoid this annoyance, use the **Save a Copy** command (Shift-Ctrl-Alt-S) for saving to a non-SVG format. After saving a copy, the opened document remains associated with the original SVG file, and any further changes are saved to it.

B.2 SVG Variants

Inkscape uses the *Inkscape SVG* vector format. As explained in 1.4, this format is standard-compliant SVG with some Inkscape-specific extensions that affect only whether you can edit various object types in Inkscape. These extensions never change those object types' appearance. It does not make much sense to save as *Plain SVG* except for a (modest) file size gain or when you encounter some buggy software having problems with Inkscape SVG. Both SVG flavors have *compressed* varieties (using the *.svgz* filename extension), which produce much smaller non-human-readable files but are otherwise the same and should be understood by most SVG software.

You can open SVG files exported from Adobe Illustrator as usual. These files usually contain a lot of AI-specific elements, which are useless for Inkscape but blow up the size of the SVG file. Inkscape has an import filter that removes the AI-specific binary chunks and converts AI layers into Inkscape layers; to trigger it, the SVG file needs to have a filename extension of *.ai.svg* instead of simply *.svg*.

B.3 PDF

After SVG, PDF is the most powerful and widely recognized vector format that Inkscape supports (1.5.1.1). While Inkscape is not the best tool for *roundtrip* PDF editing (that is, opening a PDF, editing its content, and saving back as PDF), PDF is by far the best option for connecting Inkscape with software that does not support SVG. Inkscape supports most of the latest versions of PDF's static features (shapes, text, images, gradients, opacity, meshes) but not PDF's interactive features (such as forms).

B.3.1 Import

The biggest stumbling block is that PDF supports multiple pages in a document, while SVG does not. The first thing you see in the PDF import dialog (Figure B-2) is therefore a page selector. The dialog shows you a preview of the selected page before you click OK to import it.

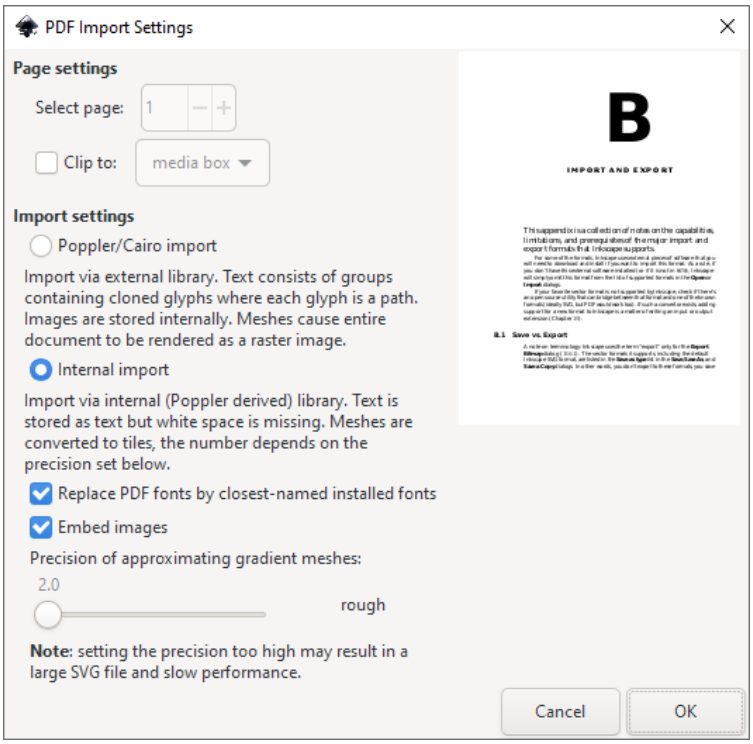


Figure B-2: PDF import dialog when opening or importing a PDF file

You can also clip (18.3) the imported artwork to the various boxes that may be defined in the PDF document. For example, clipping to the *media box* (that is, the page size) hides any objects that the PDF might contain outside the page area; most PDF viewers won't show them anyway, but Inkscape allows you to discover them.

Inkscape's PDF import comes in two types: *Poppler/Cairo* and *Internal* import. The import dialog claims that the Poppler/Cairo import is done "via an external library," but that is misleading; Inkscape users don't need to install anything external for either of these import types. Which type to choose depends on its features and the different ways it treats text in PDF.

B.3.1.1 Text and Fonts

A PDF file can—and usually does—have its own fonts embedded into it; SVG does not (see 15.8 for the backstory). Inkscape can deal with this limitation in two different ways.

If you mostly care about how your document looks, choose the **Poppler/Cairo import** option (Figure B-3, left). With it, each text character becomes a symbol (16.7) that you can unlink (16.5) to convert to an editable path. This option fully preserves the appearance but, obviously, the text is no longer editable as text.

In contrast, Internal import preserves text as text—but it will look correct only if you have the corresponding font installed in the system (Figure B-3, right). For example, if a PDF file uses the font FancyShmancy for text, you don't need to care (or even know) when you just view that PDF file in Adobe Acrobat Reader or another PDF viewer, because the font is embedded into the file and the PDF viewer uses it. However, if you want to import that file into Inkscape using the Internal import option, you need to get the FancyShmancy font somewhere and install it on your computer before attempting the import. Without that, the FancyShmancy text will look all wrong.

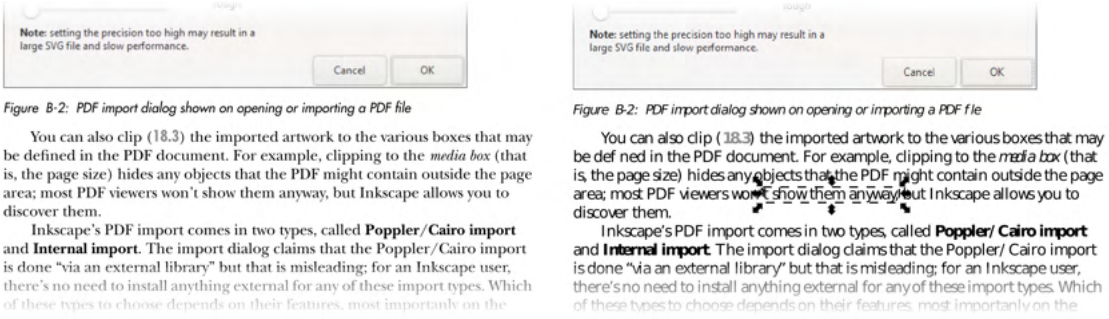


Figure B-3: A PDF page of this book imported with Poppler/Cairo (left) and with the internal importer but without the main text font installed on the computer (right)

Even if you do have the exact font that the PDF document uses, editing texts in an imported file is not very natural. A paragraph of PDF text is never converted into a single text object with automatic line wrap. Instead, each line becomes a text object of its own. Often, it is even worse than that: you will see lines broken into text-object fragments at seemingly random points (as the right side of shows). This happens because PDF is fundamentally a visual format that doesn't provide any higher-level logical structures—even as simple as paragraphs.

Inkscape offers the Replace PDF fonts by closest-named installed fonts option that may help with this issue. With it, if you don't have the FancyShmancy font installed but have FancyPancy, Inkscape will use that font, which may even work if that font is a close enough replica.

An additional complication is that most PDFs refer to the fonts they use by their *PostScript names*, which may differ from the names of fonts your operating system shows. For example, a PDF may refer to a font called `AlbertusMT-Light`, whereas the same font in your Inkscape font list is called `Albertus MT Lt`. Inkscape does its best to convert PostScript names to regular names, but it can do that only for the fonts you have installed—and even then it sometimes fails and chooses the wrong installed font.

You can always look up the original PostScript name of any text object's font in an imported PDF by looking at its style in the Selectors and CSS dialog (8.1). The property you need is `-inkscape-font-specification`. For example, that property may contain `AlbertusMT-Light`, whereas the standard `font-family` property is `Albertus MT Lt`.

B.3.1.2 Meshes

Like SVG, PDF supports mesh gradients (10.7). However, Inkscape does not attempt to translate PDF meshes into SVG meshes; instead, it approximates them with grids of flat-color tiles. At the bottom of the dialog (Figure B-2 on page 489), you can select how precise you want this approximation to be, from rough to very fine. Typically, you'd need to experiment to figure out the best value for your mesh-using PDF file, but the dialog will warn you that requesting too high a precision will make your file too big and slow to work with.

NOTE *Imported gradient meshes, with their lattices of small colored paths, are convenient objects for moving, painting, and reshaping with the Tweak tool (6.10, 8.9, 12.6).*

B.3.2 Export

As mentioned, Inkscape does not have a separate "Export" command for vector formats; instead, go to **File ▸ Save As** or **File ▸ Save a Copy** and choose the PDF format in the **Save as type** list. After you type the filename and click **OK**, you will be presented with a dialog for setting PDF export options (Figure B-4).

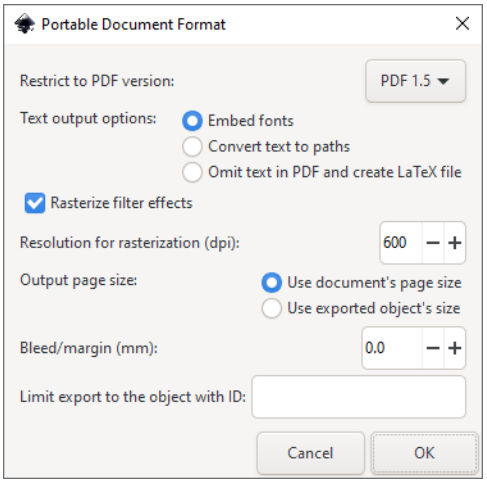


Figure B-4: PDF export options

You can export PDF version 1.4 or 1.5. For the fonts your document uses, you can either embed them into the PDF or convert all text objects to paths on export.

Filters (Chapter 17) are SVG features that have no counterpart in PDF. You have the option of converting any filtered objects to bitmaps on export (Rasterize filter effects). If you uncheck this, filters will be simply ignored (for example, any blurred object will lose blur). Rasterization inflates the file size and loses the vector editability of the objects affected, but it preserves the objects' filtered appearance. The Resolution for rasterization parameter can be the default 96 dpi for PDFs intended for viewing onscreen, but it needs to be at least 300 dpi for PDFs intended for print.

By default, the entire document is exported to PDF, but you can limit your export to a single element by entering its ID (4.1). For example, in a file with several logos, you can export each logo to its own PDF file using this option. The PDF's page size is, by default, the same as that of your SVG, so any objects outside the page are hidden (but still present in the PDF code); with Output page size set to Use exported object's size, the PDF page will be as big as the bounding box of the object you're exporting.

B.4 PostScript and EPS

A couple decades ago, PostScript was *the* exchange format for vector data. Now it has largely ceded its position to PDF, which is much richer and better supported by all kinds of software. If you have the choice, you should use PDF instead of PostScript. However, a lot of old projects and clipart exist as PostScript files, and you still may need to deal with them.

EPS is PostScript with some additional limitations that allow you to import and insert it into other documents. An EPS file is always a single page, always has all fonts and bitmap images embedded (regular PS files are not obliged to embed anything), and its page size is always clipped to its contents.

B.4.1 Import

To import PS and EPS, you need to install the Ghostscript interpreter that Inkscape runs automatically. Ghostscript converts PS or EPS files to PDF format, which are then fed into Inkscape. Versions for all major operating systems are available at *https://www.ghostscript.com/*. Make sure the *ps2pdf* (on Linux) or *ps2pdf.bat* (on Windows) file of Ghostscript is in your `PATH`.

Because PS and EPS files end up as PDF imports from Inkscape's viewpoint, you will see the same PDF import dialog (Figure B-2 on page 489). In particular, if a PS file contains more than one page, this dialog will allow you to choose which page to import.

B.4.2 Export

Inkscape supports PS and EPS export natively. The dialog has many of the same options as the PDF export dialog, as shown in Figure B-5.

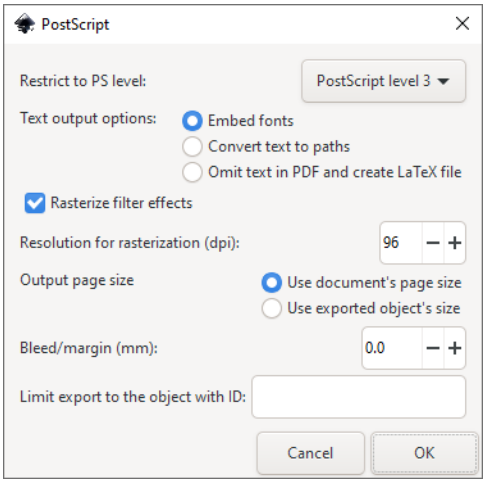


Figure B-5: PS or EPS export options

You can choose the level (that is, version) of PostScript; most modern software and printers support Level 3. Just as in PDF export, fonts are either embedded or converted to paths, and filters can be rasterized. Objects with nonzero opacity are also rasterized (you cannot suppress it), because unlike PDF, PostScript does not support vector transparency.

B.5 AI

Up to version 8, Adobe Illustrator's (AI's) native format was based on PostScript (1.5.1.1). It was not fully standard PostScript, however, and importing it into other software has always been a pain. Various scripts on the web claim to convert this old AI format into something more tractable, but they are all rather limited and unreliable. While you can try using them, Inkscape doesn't officially support import or export of the old AI format.

Starting with version 9, Adobe switched to PDF for the base of its new Illustrator format. While the AI files saved by Illustrator contain a lot of AI-specific extensions, they are standard PDF that any PDF-capable software can open and view. This is what Inkscape does: it treats any file with the *.ai* extension as a PDF file and presents its standard PDF Import Settings dialog to you (B.3). On import, Inkscape loses the AI-specific metadata (such as layers), but at least you get your vector objects as vectors.

There's no support for AI export in Inkscape because recent versions of Adobe Illustrator can import SVG and PDF files without problems.

B.6 CorelDRAW

Inkscape can import various kinds of files created by the CorelDRAW vector editor (with the extensions *.cdr*, *.cdt*, *.ccx*, and *.cmx*) natively. Export to CorelDRAW is not supported.

B.7 WMF, EMF, and EMF+

WMF (Windows MetaFile), EMF (Enhanced MetaFile), and EMF+ (Enhanced MetaFile Plus) are Windows-specific vector formats used by some Windows-only software (such as Microsoft Office) for data exchange and vector clipart. Of these, EMF is the most capable and widely supported, so it's generally preferable (if you have a choice). Inkscape supports these formats for import and export natively.

B.8 XAML

Inkscape can both import and export the XAML (Extensible Application Markup Language) format used by Microsoft in its .NET and Silverlight technologies. No additional software is needed.

B.9 WPG

WPG (WordPerfect Graphics) is an old vector format that the WordPerfect text processor used; collections of clipart in this format still exist. Inkscape imports WPG natively (no external software is needed).

B.10 VSD

VSD is the native format of Microsoft Visio. Inkscape has limited native support for VSD file import.

B.11 DXF and HPGL (Export)

DXF (Drawing Exchange Format) is a common CAD (Computer-Aided Design) format used for plans and technical drawings in software such as AutoCAD. HPGL is a vector format used by some Hewlett-Packard plotters. Inkscape has limited support for importing from and exporting to these formats.

B.12 ODG

OpenDocument Graphics (ODG) is the format used by, among others, the OpenOffice and LibreOffice suites. Inkscape has limited support for ODG export; with recent versions of the office suites, it's better to use SVG for graphics exchange.

B.13 POV

POVRay (*http://povray.org/*) is a popular open source 3D raytracer, not a vector application. However, Inkscape can export the paths and shapes as 3D scenes that POVRay will then render; you can manually edit the text-based *.pov* file to adjust angles, cameras, lighting, and so on.

B.14 LaTeX (Export)

LaTeX is an old and powerful open source document formatting system. Inkscape can output its drawings directly into a LaTeX document. You will need the PSTricks package (*http://tug.org/PSTricks/*) installed in your LaTeX system to render that file.

B.15 Bitmap Formats (Import/Export)

As we saw in 18.6, Inkscape can export to PNG, JPG, TIFF, and WebP formats natively. It has built-in import support for a lot more bitmap formats, including all the major ones (PNG, JPG, TIFF, GIF, and others).

The GIMP bitmap editor uses the XCF format. Inkscape can export to this format via an extension, preserving the layers (that is, layers in the SVG file become layers in the XCF file). You need to have GIMP installed and in PATH for this extension to work.

B.16 Printing

You can think of printing of as exporting—if only because, to print a document, Inkscape exports it to the format that the printer understands. Depending on the printer driver, this is likely either PostScript or PDF, not SVG. If you select the Print to file option from the Print dialog, you will get a PS or PDF file that looks exactly as if you saved your document with this file type.

I'm not going to describe the Print dialog that you get when you select File ▸ Print (Ctrl-P) because it is created by your operating system, not by Inkscape, and its options—paper size, margins, print quality, and so on—reflect your printer's capabilities. The only thing Inkscape adds to this dialog is the Rendering tab.

By default, Inkscape prints by exporting the document to a vector format (PostScript or PDF). As explained earlier, Inkscape is smart enough to rasterize those objects using the features these formats don't support: filters and meshes (both PS and PDF) and transparency (PS only). Alternatively, you can render the entire document into bitmap by switching from Vector to Bitmap on the Rendering tab and setting the bitmap resolution. This option is generally safer, as it avoids any kinds of surprises that format conversion may hold, but it may result in large print files and slow printing.

C

THE COMMAND LINE

Unlike most vector editors—but like the majority of open source software—Inkscape has a powerful command line interface. With it, you can perform tasks such as exporting, document modifications, queries, and so on from scripts or from a command line prompt without using Inkscape's graphical user interface (GUI). In most cases, after Inkscape completes a task as specified in the command line parameters, it simply quits without showing the GUI at all. This makes it run faster and consume fewer resources, as no time or memory is wasted on creating and destroying the GUI.

In this appendix, I explain and provide examples of Inkscape's most common command line parameters. I also introduce the language of actions, available via the `--actions` parameter, that allows you to write complex document-processing scenarios. Since the program's command line interface has been thoroughly revised in recent versions, I also point out the most important changes to help you migrate old scripts that are likely to break with Inkscape 1.0.

C.1 Command Line Executable

On Windows, command line utilities and GUI applications are two different types of programs. A single executable cannot run both as a GUI application and as a command line application. That's why Inkscape on Windows contains, in its *bin* folder, two executable files: *inkscape.exe* and *inkscape.com*.

- *inkscape.exe* is the GUI application to run when you need to work in Inkscape interactively. If you try to use it for command line tasks, you will see no output in the console (command prompt window), which makes the query parameters (C.5) unusable. Also, *inkscape.exe* will return asynchronously (immediately), possibly before a complex action, such as an export, has finished.
- *inkscape.com* is best used for scripting and command line tasks. It displays proper output in the console and returns synchronously. However, if you run it for the GUI, you will always get a parasite command line window in the background that cannot be closed without closing the GUI window.

On Linux or macOS, there's a single executable called simply *inkscape* that is used both for the GUI and the command line. In the examples in this appendix, I use `inkscape` as the name of the executable. Also, I use forward slashes (`/`) in filename paths, which work on all platforms including Windows.

C.2 Getting Help

To get a complete list of the command line parameters known to your version of Inkscape, run it with the `--help` parameter:

```
$ inkscape --help
Usage:
  org.inkscape.Inkscape [OPTION…] file1 [file2 [fileN]]

Process (or open) one or more files.

Help Options:
  -?, --help                                Show help options
...
```

The parameters, or *options*, have names starting with `--`, such as `--help`. Some of them can have values separated by `=`, such as `--export-filename=file.pdf`. Use spaces to separate command line parameters. A more detailed reference is available from Inkscape's UI in Help ▸ Command Line Options (this opens a web browser and fetches the page from the internet) or by typing `man inkscape` on the command line (Linux only).

A very useful parameter (especially if you want to report a bug or request a feature from Inkscape developers) is `--version`, which prints the version number as well as revision and build date of your copy of Inkscape. You can also query Inkscape with `--system-data-directory` or `--user-data-directory`; these parameters print the values labeled, respectively, Inkscape data and User config on the System page of the Preferences dialog.

C.3 Opening Documents

The simplest use of the command line is to list, without any parameter names, the filepaths of the documents you want to open in the GUI. For example, entering the following will start Inkscape's GUI and load two SVG documents and one PDF document (automatically imported) into three Inkscape windows:

```
$ inkscape file.svg folder/subfolder/document.svg pdfs/file.pdf
```

You can specify some import options for opening non-SVG files. The most common are `--pdf-page`, which selects the page to import from a PDF document, and `--pdf-poppler`, which switches to the Poppler import method (B.3.1). For example, here's how to open page 5 from a book chapter in PDF:

```
$ inkscape --pdf-page=5 chapter.pdf
```

If you have a script or program that generates an SVG file and prints it to the console, use the | character to *pipe* that document to Inkscape with the `--pipe` parameter, for example:

```
$ svg-generating-script | inkscape --pipe
```

C.4 Export

One of the most common Inkscape tasks is exporting SVG documents into other formats. Via the command line, Inkscape can export bitmaps to PNG (18.6) and vectors to PS (PostScript), EPS, PDF, EMF, WMF, and XAML (Appendix B). Inkscape determines the format to use based on the filename extension you provide on the command line; if you want to specify the format explicitly, add `--export-type` with the required type (valid values are `svg`, `png`, `ps`, `eps`, `pdf`, `emf`, `wmf`, `xaml`). For example, exporting to PDF is as simple as entering the following:

```
$ inkscape --export-filename=file.pdf file.svg
```

This is equivalent to

```
$ inkscape --export-filename=file --export-type=pdf file.svg
```

This will load *file.svg* and create the PDF file; no GUI is loaded, and after completing the export, Inkscape quits. If you want to force overwriting the export file, add `--export-overwrite`.

You can specify multiple input files and multiple export types (separated by commas) in `--export-type`. For example, the following

```
$ inkscape --export-type=png,pdf file.svg file2.svg
```

creates four files: *file.png*, *file.pdf*, *file2.png*, *file2.pdf*.

You can also export a document to SVG—for example, you can use Inkscape to convert a PDF file to SVG or to extract an element from one SVG file and save it as an SVG of its own. If you add `--export-plain-svg`, the document will be saved as plain SVG instead of Inkscape SVG (1.4). If you add `--vacuum-defs`, the resulting SVG will not have any unused definitions in `defs` (A.4). For example, this will convert one page from a PDF file into plain SVG:

```
$ inkscape --export-filename=file.svg --export-plain-svg --pdf-page=5 file.pdf
```

[1.1] In older versions, Inkscape had separate parameters for each exported format: `--export-png`, `--export-pdf`, `--export-ps`, `--export-eps`, and so on. They all took a value of the filename. Now, `--export-filename` has replaced these. The `--export-plain-svg` remains but is now a Boolean switch instead of a filename parameter.

In addition to the export formats it supports natively, Inkscape includes a number of output extensions (19.4.2) for less common formats. You can use these extensions via the command line, too. The `--export-extension` parameter takes as its value the unique ID of the extension, which you can look up in its *.inx* file (19.4.1). For example, Inkscape 1.1 includes an extension for exporting to the HPGL format used by plotters. In the *share\inkscape\extensions\hpgl_output.inx* file, you can find its ID, which is `org.ekips.output.hpgl_output`. Now, running

```
$ inkscape --export-extension=org.ekips.output.hpgl_output \
    --export-filename=file.hpgl file.svg
```

converts *file.svg* into HPGL format, creating a file named *file.hpgl*.

C.4.1 Export Area

By default, Inkscape exports the document's page (2.3), and objects falling outside the page are invisible in export. Add `--export-area-drawing` to make the export cover all visible objects of the document, regardless of its page size. For example:

```
$ inkscape --export-filename=file.png --export-area-drawing file.svg
```

The only exception is EPS, where the drawing is exported by default; with this format, use `--export-area-page` to export the page. Even then, due to the limitations of EPS, the export area will be clipped to the actual objects in the page if they do not reach to the page edges.

When exporting from SVG, you can export a single object out of a document, so that the exported file covers that object's bounding box and nothing else. The object is specified by its `id` attribute (A.9):

```
$ inkscape --export-filename=text.pdf --export-id=text2054 file.svg
```

For PNG export, if other objects overlap with the exported object's bounding box and are visible, they will also show in the exported file. For SVG export, all other objects are still present in the file even when they are outside the exported

page area. To suppress the other objects and make a rendering of only the chosen object, add `--export-id-only`. For PDF, PS, and EPS, this is the only possible mode—other objects are always dropped when you specify `--export-id`.

For PNG export, you can also provide the export area explicitly by specifying its two corner points. For example, the following exports the area that spans from point 0, 0 to point 200, 100 (in px units):

```
$ inkscape --export-filename=area.png --export-area=0:0:200:100 file.svg
```

Also for PNG export, no matter which method you use for specifying the area, you can *snap* that area to the pixel grid—that is, round it outward to the nearest whole coordinates in `px` units—by adding `--export-area-snap`. This is useful when you export at the default 96 dpi and want your objects drawn to the pixel grid (7.2) always to be crisp in the exported bitmap regardless of the area you are exporting.

C.4.2 Export Size and Resolution

For PNG export, you can specify the size of the exported bitmap or its resolution (by default 96 dpi). For example:

```
$ inkscape --export-filename=file.png --export-dpi=600 file.svg
$ inkscape --export-filename=file.png --export-width=1000 file.svg
$ inkscape --export-filename=file.png --export-height=400 file.svg
```

The first line exports *file.svg* at the resolution of 600 dpi, so that a document page 3 inches wide exports to a bitmap 1800 pixels wide. The other two examples explicitly set the pixel size of the export, and the resolution is calculated to match this requirement (overriding `--export-dpi` if it is also present).

If only `--export-width` or `--export-height` is present, Inkscape calculates the other dimension at the same resolution. However, with command line export, you can also get a PNG file where the vertical resolution is not equal to the horizontal resolution (something you can't do from the GUI, 18.6.1.2). For example, you can export a 1000 by 1000 pixel area into a distorted PNG file of 2000 by 500 pixels with `--export-width=2000 --export-height=500`.

The `--export-dpi` parameter also affects export to vector formats (PS, EPS, PDF) whenever Inkscape needs to rasterize some of the file's features (such as filters and mesh gradients for PS and PDF, or transparency for PS).

C.4.3 Export Background (PNG Only)

Areas that have no objects in them appear to be transparent in all export formats. However, in PNG export (but not PDF, PS, or EPS), you can specify any background color or opacity to be applied during export. For example, if you want a solid opaque black background, enter:

```
$ inkscape --export-filename=file.png --export-background=#000000 \
    --export-background-opacity=1.0 file.svg
```

NOTE *The \ character in this example shows that the long command line was wrapped for display; you should type it as a single line without the \.*

The syntax for the background color is the same as in SVG; in particular, you can use the `#RRGGBB` format. The background opacity (defaulted to fully opaque) is given either as a floating-point number from 0.0 to 1.0 or as an integer from 1 to 255; for example, `--export-background-opacity=0.5` is equivalent to `--export-background-opacity=128`.

[1.1]

C.4.4 Color Mode (PNG Only)

For PNG export, you can set the exported file's color mode with the `--export-png-color-mode` option that takes one of the following values: `Gray_1`, `Gray_2`, `Gray_4`, `Gray_8`, `Gray_16`, `RGB_8`, `RGB_16`, `GrayAlpha_8`, `GrayAlpha_16`, `RGBA_8`, or `RGBA_16` (18.6.1.5).

C.4.5 Export Hints (PNG Only)

Every time you export a single selected object to PNG via the GUI (18.6), the export filename and resolution are recorded into the export hint attributes added to the corresponding element. If you then save the document with those hints, you can use them for command line export to PNG as well. For example, if you write:

```
$ inkscape --export-id=text2035 --export-use-hints file.svg
```

Inkscape exports only the object with the `text2035` ID to the same file with the same resolution it had when most recently exported from the GUI. Note that there's no `--export-filename` because the filename is taken from the export hint.

C.4.6 Vector Export Options

For exporting to PDF, PS, and EPS, you can set these options:

- `--export-dpi` (C.4.2) sets the resolution for rasterized objects, such as filters and mesh gradients.
- `--export-ignore-filters` suppresses the filtered objects' rasterization, which are exported without filters applied.
- `--export-ps-level` gives the level (version) of the PostScript format to use; it can be 2 or 3 (default).
- `--export-pdf-version` gives the version of PDF to use; it can be 1.4 or 1.5 (default).
- `--export-text-to-path` converts all text objects to paths (this is also available for exporting to SVG).

For example, the following converts all text objects to paths on export:

```
$ inkscape --export-filename=file.pdf --export-text-to-path file.svg
```

The resulting vector file neither needs nor embeds any fonts.

C.5 Querying

Because SVG is a text-based format, it's tempting to generate or edit SVG documents with simple scripts. In such a script, you may need to specify the bounding boxes of some SVG objects—for example, to check whether the text inserted into SVG from a database fits into the provided space, or to create a background rectangle or frame for a specific object. In general, however, calculating the bounding box of an object in SVG is extremely complex—you would need to reimplement a lot of Inkscape's code to take into account everything that may affect an object's bounding box. Even when you're writing an Inkscape extension (Chapter 19), the `inkex` Python library that comes with Inkscape can calculate bounding boxes for paths but not for texts.

Fortunately, Inkscape itself comes to the rescue. For example:

```
$ inkscape --query-width --query-id=text1256 file.svg
45.2916
```

This asks Inkscape to provide the width (in px units) of the object with the `id="text1256"`. Inkscape loads the document, finds that object, prints its width back to the console, and quits.

Similarly, you can use `--query-height`, `--query-x`, and `--query-y` parameters to find the dimensions and coordinates of an object's bounding box. The `--query-id` parameter can contain a comma-separated list of `id`s.

Such Inkscape calls are reasonably fast because they don't load the GUI and don't render the document. However, if you need multiple bounding box numbers for multiple objects, using these parameters may cause a delay. In that case, it is better use the `--query-all` parameter, which returns all bounding box numbers for all objects in a document:

```
$ inkscape --query-all file.svg
svg2,-55.11053,-29.90404,328.3131,608.6359
layer1,-55.11053,-29.90404,328.3131,608.6359
image2372,-8.917463,349.8089,282.12,212.6382
text2317,-39.85518,454.3014,20.40604,13.32647
tspan2319,-32.58402,454.3014,12.79618,4.989286
tspan2408,-39.85518,462.4921,20.40604,5.135838
path2406,-16.43702,386.617,6.34172,154.7896
text2410,-46.11609,376.8753,34.34841,5.135838
tspan2414,-46.11609,376.8753,34.34841,5.135838
text2418,-55.11053,365.9197,43.02429,5.135838
```

Each line is a comma-separated list of the object ID, X, Y, width, and height. Parsing such a line in your script should be easy.

C.6 Actions

Inkscape's command line is not limited to GUI-less export, conversion, and querying tasks. You can script a number of regular editing tasks as well using the `--actions` parameter, which lets you list any number of actions for Inkscape to perform sequentially.

NOTE *In older versions, Inkscape had verbs instead of actions, accessed through the `--verb` parameter. Now, verbs are deprecated in favor of actions. The `--verb` parameter still works, and you can use any verb as an action; Inkscape still supports more verbs than actions. In the future, however, verbs will be removed.*

Each action more or less corresponds to a command that you choose from a menu when editing a document in the GUI. An action has a *name* and, optionally, one or more *arguments*. On the command line, use a colon (`:`) to separate the name and the arguments, a comma (`,`) to separate arguments of a single action, and a semicolon (`;`) to separate actions in the list. You can also include spaces in a string, but then the string must be enclosed in double quotation marks (`"..."`) and can't contain additional quotation marks.

For a complete list of actions your version of Inkscape supports, run Inkscape with `--action-list`. Here's the top of that list, showing some actions that correspond to some command line parameters for export:

```
$ inkscape --action-list
action-list         :  Print a list of actions and exit.
convert-dpi-method  :  Import DPI convert method.
export-area         :  Export area.
export-area-drawing :  Export drawing area.
export-area-page    :  Export page area.
export-area-snap    :  Export snap area to integer values.
export-background   :  Export background color.
export-background-opacity:  Export background opacity.
export-do           :  Do export.
...
```

C.6.1 Example: Changing CSS Property

Let's look at a practical example. Imagine you want to open *file.svg*, select all text objects, paint them red (`#FF0000`), and save the result. Here's how you do this via actions:

```
$ inkscape --actions="file-open:file.svg; select-by-element:text; \
    object-set-property:fill,#FF0000; export-filename:fileout.svg; export-do"
```

Let's break this string into parts at each semicolon and look at the individual actions:

- `file-open` opens the file given in its argument—in this case, `file.svg`. You could also drop this action and simply provide `file.svg` on the command line after the `--actions` parameter.
- `select-by-element` selects all objects with the given element name—in this case, `text`.

- `object-set-property` sets a CSS property (8.1) on selected objects. Two arguments provide the name of the property (`fill`) and the value (`#FF0000`). This has the effect of painting all texts red (unless they have `tspans` with their own `fill` inside, which would override that).
- `export-filename` prepares the export of the result by specifying the filename (and, from its extension, the SVG export format). There's currently no `file-save` action, which may be for the better; it is safer to save (that is, export) the result of your changes into a different file instead of overwriting the original.
- `export-do` (without arguments) performs the actual export.

After this command finishes, you will have, in the current folder, the *fileout.svg* file that is identical to *file.svg* except all of its text objects now have a red fill color.

C.6.2 Shell Mode

Inkscape's shell mode is an easy way to explore the actions. If you run Inkscape with `--shell`, it enters an interactive mode where you can type actions with their arguments (as well as verbs) at Inkscape's prompt and have these actions performed at once. For example:

```
$ inkscape --shell
Inkscape interactive shell mode. Type 'action-list' to list all actions. Type 'quit' to quit.
 Input of the form:
 action1:arg1; action2;arg2; verb1; verb2; ...
Only verbs that don't require a desktop may be used.
> file-open:page.svg
> select-by-element:text
> delete
Unable to find: delete
verbs_action: Invalid verb: delete
> select-by-element:text
> object-set-property:fill,#00ff00
> export-filename:green.svg
> export-do
> quit
```

Sequences of commands that you tested and found working in the interactive shell mode can then be reused in a script.

When typing action names or filenames, press Tab to autocomplete. Inkscape's shell mode remembers the history of your commands across sessions.

INDEX

Symbols and Numbers

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

"THIS EXCELLENT TEXT CAN MAKE YOU A PRO AT USING INKSCAPE."
—IT WORLD

EXPLORE THE WORLD OF VECTOR ART

Welcome to your master class on Inkscape, the world's most popular open source vector editor. This second edition of *The Book of Inkscape* covers the software's latest major release (1.1), including all of its improved features, customizable keyboard shortcuts, advanced path effects and filters, extensions, and new UI conveniences.

Former core Inkscape developer Dmitry Kirsanov shares his deep knowledge of the program's inner workings and the joy of thinking in vector. Starting with a basic primer on creating and manipulating graphics in Inkscape, he presents every aspect of the newest release's enhanced functionality with detailed explanations, illustrations (including eight pages of full-color art), and practical tips. The book's step-by-step tutorials demonstrate the power of Inkscape in projects ranging from animations and business cards to technical drawing, design composition, and 3D sketches.

You'll learn how to:

- Create, transform, style, clone, and combine objects
- Export Inkscape artwork into a variety of formats
- Navigate Inkscape's rotating canvas and customize your workspace
- Use gradients, patterns, filters, and chainable path effects to liven up your designs
- Manage your graphics with work layers, groups, object order, and locks
- View and manipulate image structure with the XML Editor and the new Objects dialog
- Use Inkscape's powerful command line options in automated scripts

As the Inkscape universe continues to expand, its vector-editing tools are more powerful than ever before. This is the only book you need to master them all.

ABOUT THE AUTHOR

Dmitry Kirsanov is an author, developer, and graphic designer. As one of Inkscape's core developers, he added many features to the program and made its UI more fluent and flexible. He currently works as an XML/SVG and web developer. In his spare time, he writes and edits books.

COVERS INKSCAPE 1.1

no starch press

THE FINEST IN GEEK ENTERTAINMENT™

www.nostarch.com

2ND EDITION

$49.99 ($65.99 CDN)

ISBN 978-1-7185-0175-1

9 781718 501751 54999