Full Circle

INKSCAPE SERIES SPECIAL EDITION

THE INDEPENDENT MAGAZINE FOR THE UBUNTU LINUX COMMUNITY

INKSCAPE SERIES SPECIAL EDITION Vol 9

INKSCAPE

Volume Nine   **Parts 58 - 64**

Written by Mark Crutch

Inkscape - Part 58

I'm going to take a slight departure from the usual format for the first part of this month's article – and talk about politics. Not Trump, Brexit or the rise of populism, but rather the politics of open formats and browsers.

First, a very brief (and simplified) history lesson: SVG, the file format used by Inkscape, was created under the auspices of the World Wide Web Consortium (W3C) – the organisation that was also charged with creating the specifications for HTML and CSS. HTML was already an established language, but loosely defined, and with a host of differences between the various browser implementations. The W3C tidied things up, but ultimately decided that the best way to get everyone writing good, cross-browser HTML was to effectively abandon the undisciplined language it had become, and move to a rigorously defined and structured alternative, XHTML. This was also part of a larger plan to promote XML, which can be thought of as a language for defining languages. XHTML was HTML re-cast as an XML language, bringing with it more scope for interoperability with other XML languages, including SVG.

As academically pure as the aims of XHTML were, they fell down in the real world. HTML had thrived partly because it was so lax. Browsers would do their best to interpret even the most malformed syntax, which greatly lowered the barrier to entry for non-programmers to create their own web pages. Lowering it further still were applications such as Dreamweaver and HoTMetaL, which would allow users to create web pages as easily as they would a Word document. HTML continued to proliferate online, and it would have been commercial suicide for any browser to render only XHTML. For all its purity and technical superiority, XHTML inevitably lost out to the looser standard, and the W3C's work became largely irrelevant. It was clear that the approach of having a standards body to write the specs, and only then for the browsers to implement them, was not one that would work in practice.

What followed was a period of stagnation for the web. No browser wanted to introduce any radically new syntax into HTML or CSS for fear of re-kindling the bad old days of proprietary extensions. But eventually, the browser companies began to talk amongst themselves about ways to push the web forward again. The result was the formation of another standards body, WHATWG, whose remit was to improve the old HTML specs largely by documenting what the browsers already did, making it easier for all the vendors to bring their programs up to the same level of compliance. They also added a few new features to HTML, branding it as “HTML5”, although we're now several years on and many of their more useful ideas still haven't been universally implemented (how are those date and time pickers coming along, Mozilla?).

Eventually the W3C gave up on their philosophical march towards XHTML purity, and embraced the work of WHATWG, such that the HTML standard is now nominally back in their hands. But structurally, things have changed: no longer can the W3C write specs and expect browsers to implement them; now the browser vendors agree what to implement amongst themselves, and then the specification is written to match the implementations. Okay, in practice it's more nuanced than that, but the key point is that, these days, specs are largely driven by what browsers are prepared to implement.

This has an impact on Inkscape because, as an SVG editor, its feature set follows the capabilities written into the SVG specification. But the SVG spec, in practice, can't gain any new capabilities without support from the browser vendors. Yet those vendors are loath to implement many of the new features, given that there are barely any files online that use them. Users, meanwhile, are equally loath to create content using these new features because

contents ^

no browser supports them. The authoring tools (such as Inkscape) would like to implement them, but, without browser support, the spec is unlikely to be finalised and supported – so any work they do could be rendered obsolete if the specification changes.

And so we go round in circles: no files using the new features online means no browser support; no browser support means the specs don't stabilise; unstable specs make authoring tools less likely to support the features; no support in authoring tools makes users less likely to create and post files that use the new features; no files using the new features online means no browser support... and so on.

To be fair, some limited support for new SVG features has made it into browsers – but mostly in areas where the SVG Working Group has relinquished ownership in order to move the feature to CSS. This is both good and bad news: CSS is one of the cornerstones of the web, so adding features there, rather than in SVG, makes them more likely to be adopted by browsers; conversely it further weakens the position of SVG as a stand-alone format, and requires non-browser applications to comply with standards that often don't sit easily outside the web environment, diminishing SVG's position as an independent file format.

With more features moving to CSS, and the vendors showing little interest in implementing those that remain part of SVG, there's even been talk of not renewing the SVG Working Group's charter beyond a short period to stabilise the work that has been done on the SVG 2 specification over the past couple of years. That would mean no SVG 3, and no new features in the future. Given how many great ideas were dropped from SVG 2 with the promise that they could be revisited for later specs, this would be a tragedy. Sure, Inkscape would likely continue, probably adding proprietary extensions to SVG to support new features as time goes on. But the promise of an open vector format that could be used cross-application, and rendered natively on the web, would have died.

Is there anything that we, as users and advocates of open formats, can do to help ensure that SVG has a future? Since it's largely in the hands of the browser vendors, the best we can to is to show them that there is a demand for the format, and for the new additions that are being made to it. We need to create more SVG documents, especially those with features from the SVG 2 specification, and post them online. And we need to encourage others to do the same. But this approach isn't without its problems.

The SVG 2 spec isn't yet finalised. Creating documents using the current version could render them obsolete if there are further changes to the specification before it's finally ratified. So any files you create now might require some (hopefully minor) fixes if they are still to work in a year's time. A bigger problem for most people is how to create them in the first place. Hand-coding SVG is certainly possible, but it's probably not a practical option for most people, which means that the only way to get new features into your files is to wait for them to become available in authoring tools. Thankfully, Inkscape is, to some extent, leading the way for this approach. The recent 0.92 release adds support for rendering several SVG 2 features, although, unfortunately, UI support for creating them in the first place is somewhat more limited. Nevertheless, there are a couple of SVG 2 features that you can start using in your Inkscape drawings today, the first of which I'll cover in this article, and the second next time.

The first step towards using these new features is, of course, to install version 0.92.x of Inkscape. Windows users can just download an installer from https://inkscape.org/en/download/windows/ whereas MacOS users are left behind somewhat, with no official .dmg files available at the time of writing (see https://inkscape.org/en/download/mac-os/ for more details and alternative options).

Linux installation instructions vary between distributions, but there's now a distribution-independent Snap package available. Systems that aren't based on Ubuntu may have to install the “snapd” daemon separately (see https://snapcraft.io/docs/core/install for details), but if you're already on Ubuntu 16.04 or later, you should simply be able to run the following command:

```
sudo snap install inkscape
```

Unfortunately the snap doesn't necessarily have all the prerequisites to get Inkscape up and running correctly. One change in 0.92, for example, is that Inkscape no longer bundles a built-in copy of the Potrace library (for tracing bitmaps or using the bucket fill tool); I had to use:

```
sudo apt-get install
libpotrace0
```

to get it working on my system. There have also been theming issues with early snaps (which I also fixed by apt-get installing some additional libraries), although by the time you read this, there should have been a point release which fixes those issues. I strongly recommend launching Inkscape from the command-line at first (just enter “/snap/bin/inkscape”) as error messages in the console may make it clear if there are any unmet dependencies, whereas launching from an icon might leave you with no Inkscape window, and no indication as to what went wrong.

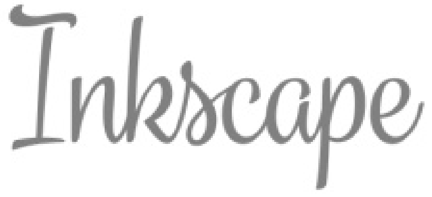
If you already have Inkscape installed via the normal Apt tools, you will find that the old version is still installed, even after you've added the snap – and that it probably gets run in preference to the new release when you just execute “inkscape” from the command-line, or click the launcher in your menu. You'll need to modify your path to give the /snap/bin directory priority over /user/bin or update your launchers and links to point to the snap version instead.

There are still traditionally packaged versions available for several distributions as well, which is especially useful if you have an older system that doesn't support snaps. See https://inkscape.org/en/download/linux/ for details. For example, on Ubuntu 14.04, you might prefer to use the stable PPA that is available, by issuing these commands:

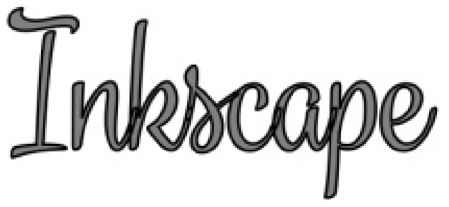
```
sudo add-apt-repository
ppa:inkscape.dev/stable

sudo apt-get update

sudo apt-get install inkscape
```

Whichever approach you take, it's worth visiting Help > About Inkscape to ensure that you are running version 0.92.

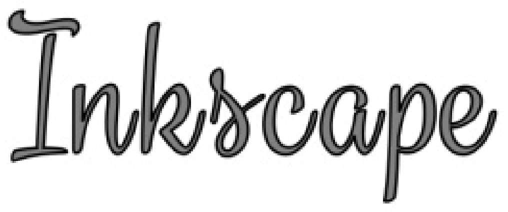
The first SVG 2 feature exposed in the UI is “paint-order”. This is actually a rather uncontroversial feature for the browser vendors, as it has already been implemented in at least Firefox, Chrome, Opera and Safari. It solves a very common problem in SVG, especially when dealing with text: any stroke applied to an object is painted on top of the fill, and extends half in and half out of the object. Consider this simple bit of text, rendered in a cursive font:



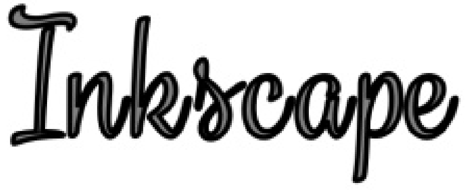
Suppose we want to add an outline to it, to make it stand out a little bit more against its background. That's simple enough, right? Just give it a thin stroke. Unfortunately that's where the problems start.



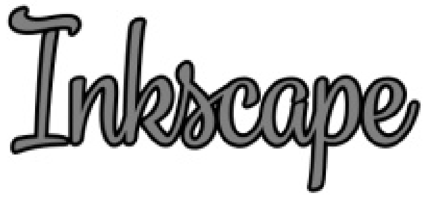
It certainly stands out more (the fact that the fill appears darker is an optical illusion that helps enhance the effect further), but, due to the construction of the font, we've now got bits of the outline appearing “inside” letters, where the tail of one flows into the body of the next. We can adjust the kerning to separate the problem characters, but that pretty much defeats the point of using a cursive font in the first place. Converting the letters to paths, then creating a boolean union, fixes the visual problem, but now our text isn't actually text any more, which in many cases makes this approach a non-starter. Let's suppose we resign ourselves to having to separate the letters. A little manual kerning gives us this:



What if we want it to stand out a bit more? Let's double the thickness of the stroke and see what effect it has.



Urk! That's not good. All the thin parts of the script have become completely filled by the stroke, ruining the light elegance that we wanted from the font in the first place. The problem, of course, is that increasing the thickness of the stroke not only adds more pixels to the outside of it, but also to the inside, obscuring more of the fill. One common solution to this – and to the previous problem – is to copy the text, putting an unstroked version directly on top of a stroked copy.

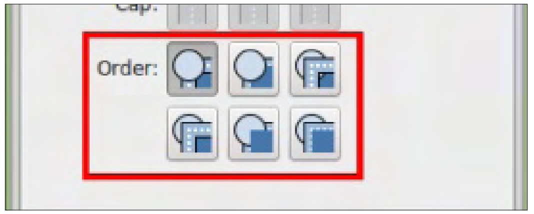


This works, but now you've got two text objects to keep in sync. With a little effort you can do the trick with clones instead, using an unset fill and stroke, but, if you want anything other than a black fill, you'll be trying to keep three objects (a text object and two clones) under control.

The problem would completely go away if only you could tell Inkscape to render the fill on top of the stroke, instead of the other way round. And that's precisely what the SVG 2 “paint-order” property does! Except it goes a step further, and also includes any markers that are on the path. Considering all the possible orderings for the three things to be rendered, this gives six possible combinations:

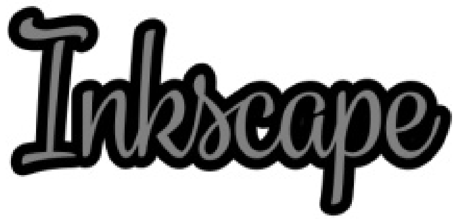
- Fill, Stroke, Markers
- Fill, Markers, Stroke
- Stroke, Fill, Markers
- Stroke, Markers, Fill
- Markers, Fill, Stroke
- Markers, Stroke, Fill

The first of these is the default, and is the way that SVG 1.x operated. But now there's an extra section in the Stroke Style tab of Inkscape's Fill and Stroke dialog that presents six buttons to let you choose your preference for any selected paths.



In each icon, the circle represents a marker, the dark blue rectangle is the fill, and the light blue path represents the stroke, with a dashed white line to indicate its centre. You can produce a similar collection of shapes by drawing a square with a thick border, converting it to a path, then setting a start marker. Clicking each of the buttons whilst your bigger version is selected will immediately reflect the change, and make it much clearer to see what the result of each option is. I recommend creating a shape like this and switching between the different modes to help you to fully understand the effect.

As for our text, because there are no markers involved, any of the three modes that draw the stroke before the fill will give our desired result, with only a single text object and no need for clones, copies, or other workarounds. It even works well with a really thick outline.



Even though paint-order is already well supported in browsers, I urge you to create new designs and works of art that use it, and put them online. The more files we share that use SVG 2 features, the more likely it is that the browser vendors might realise there's a demand for them, so going after “low hanging fruit” such as this is an easy way to express your interest without having to worry about posting files that don't render in the browser.

Next time I'll move onto Mesh Gradients – perhaps one of the most useful, and most desperately needed, new features in SVG 2, but one which is in very real danger due to browser vendor antipathy.

Mark uses Inkscape to create three webcomics, 'The Greys', 'Monsters, Inked' and 'Elvie', which can all be found at http://www.peppertop.com/

Written by Mark Crutch

Inkscape - Part 59

In the previous instalment, I talked about the very real danger facing the SVG 2 standard: although it contains some extremely useful new features, lack of buy-in from the browser vendors threatens the very future of SVG as an independent standard. As individual users, there's little we can do to influence the outcome, but one thing we can do is to start putting documents online that use some of the SVG 2 features, to prove that there's a demand for them. Short of hand-coding documents, however, this limits us to those features that have support in the authoring tools – which, practically, means those supported in Inkscape.

Last time I looked at the uncontroversial addition of Paint Order to the SVG spec. This time I'll look at another big addition – one that has been eagerly anticipated by Inkscape users since it first made an appearance in development releases – Mesh Gradients (also referred to as Gradient Meshes, depending on who you ask). To follow along, you'll need a copy of Inkscape 0.92; see the previous article for some hints about installing it.

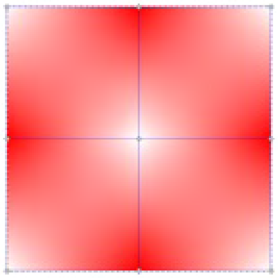
There's no doubt that the simple linear and radial gradients available with SVG 1.x are extremely limiting. Rather than add separate types for conical, spiral, square and other gradients, SVG 2 adds Mesh Gradients, which are flexible enough to cover all these cases and more – albeit at the expense of more manual tweaking to get the exact results you want. As with any type of gradient, you'll first need an object to apply it to – I'll begin with a simple square. With the target object selected, click on the Mesh Gradient icon in the toolbar:

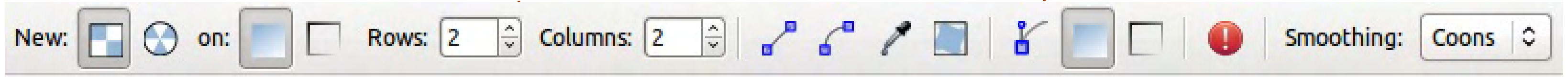
This will display the Mesh Gradients tool control bar (shown below).

To create a new mesh on your object, you first have to set up a few simple parameters. The first two icons (labelled "New:") let you select between a Mesh Gradient and a Conical Gradient. In practice there isn't a genuine conical gradient in SVG – the button just creates the convenient illusion of one using a mesh gradient. For now, ensure that the first button is selected. The next pair of buttons ("on:") define whether to create the mesh on the fill or the stroke of your object – I'll use Fill for these examples. Finally you need to decide how many rows and columns should be present in your mesh. Higher values will slow down rendering, but grant you finer control over the gradient. I'll start with a simple 2×2 mesh, as that's sufficient to explain the basics.

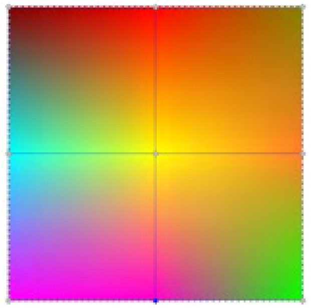
As you move your cursor back over the canvas, the status bar informs you that you can "Drag or double click to create a mesh". With a single object selected, both approaches give the same result, so I usually just double-click on the object. With more than one object selected, however, a double-click will apply the mesh to only one of them, with the others becoming de-selected. Click-dragging the mouse, on the other hand, applies separate meshes to each object. You don't even have to drag over the objects themselves, so, if you've got a particularly busy drawing, you can simply drag over a blank bit of canvas around the periphery to have the same effect without any risk of accidentally affecting other elements.

Whichever approach you take, your object will now be filled with a grid of nodes and connecting lines, dividing the area into the number of rows and columns you selected in the tool control bar. Each node is given a color, alternating between the fill color and white, to give a result something like this:

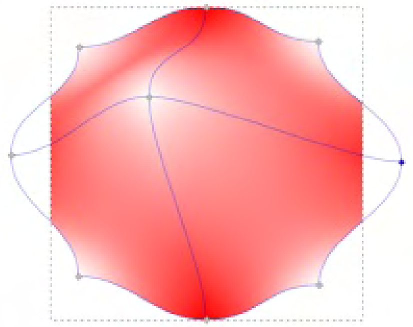




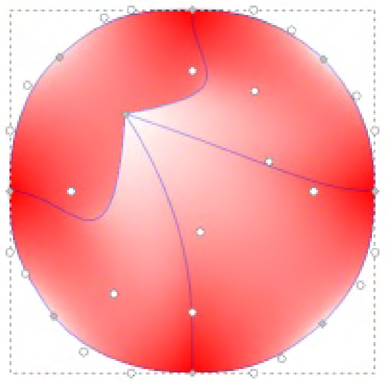
In this case there are nine nodes. With the Mesh Gradient tool active, you can click on an individual node to select it, a connecting line to select the nodes at each end, or you can drag over a number of nodes to perform a rubber-band selection. You can also use the Shift key to add nodes to the selection, or remove them from it. With one or more nodes selected, you can use the swatches at the bottom of the window, or the Fill & Stroke dialog, to assign a color to them. Setting each node to a different color immediately lets you create complex gradients that would have been extremely tedious to create with earlier releases of Inkscape:



As well as giving each node a different color, it's also possible to move them around by dragging with the mouse, allowing you to set the color at any point on your object, not just the regularly spaced ones that you're initially provided with. Dragging the edge nodes inside the object leaves empty gaps around the edge; there's no notion of repeating gradients here. Dragging those nodes outside simply cuts off the fill at the edges.



Halfway along the tool control bar you'll find a toggle button for showing and hiding the nodes' handles. With this active, you can use the handles to control the shape of the paths that join the nodes, and therefore fine-tune the precise shape of the gradients within the mesh. Long-time readers of these articles may remember that early instalments presented a variety of different ways to create a circle in Inkscape. Mesh Gradients add another to the armoury, although you're more likely to end up with something circle-ish than a mathematically correct shape.



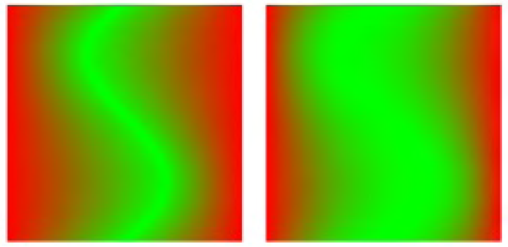
While we're in the vicinity of the Show/Hide Handles button, it's worth mentioning the buttons to the right of it. First are a pair of buttons to show and hide the editing nodes for fill and stroke gradients respectively. Depending on your preferences, gradients can (to some extent) be modified while the Node tool is active, which makes it easy to tweak your gradients by just double-clicking on your object. The nodes and handles of the gradient, however, can interfere with the handles you might usually use to manipulate your object. In the case of a rectangle, for example, the default positions of the mesh nodes put them directly on top of the handles for resizing the rectangle or changing the radius of the corners. One solution is to double-click the Node tool to access its preferences, then uncheck the "Enable gradient editing" option so that there's no longer any conflict between node and gradient editing. If you do want to be able to edit gradients with the Node tool, however, you can work around the co-located handle problem by switching to the Mesh Gradient tool, toggling the Show/Hide button(s) to turn off the visibility of the gradient nodes and handles, and then switching back to the Node tool to make your changes.

Moving on from those buttons there's one with a warning symbol, intended to make it absolutely clear that the Mesh Gradient tool is still slightly experimental. Click on it and you'll be presented with a dialog that tells you the SVG syntax could still change, and that "Web browser implementation is not guaranteed". That's quite the understatement!

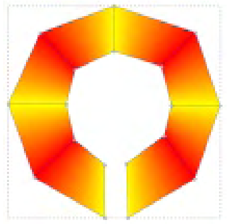
Finally on the toolbar is a drop-down menu to select the algorithm used to interpolate the colors between each node in the mesh. "Coons" mode uses linear interpolation to ramp from one color to the next, but this can result in visible banding at the boundaries between patches (a phenomenon known as "Mach

contents ^

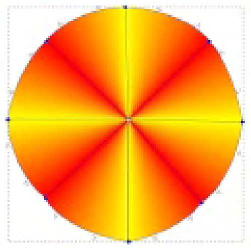
Banding"). "Bicubic" mode uses a non-linear interpolation which results in smoother transitions. In the example below, note the visible banding in the middle of the green section on the left-hand image (Coons), compared with the smooth transition of the right-hand image (Bicubic).



Now imagine, if you will, creating a mesh of one row by eight columns on a square object, then moving the bottom nodes upwards to give you a thin gradient strip running along the width of the shape. With a little manipulation, you can bend the strip into an arc, then even further into a donut. The top nodes become the outside of the mesh, and the bottom row of nodes becomes the inside. It would end up looking something like this:

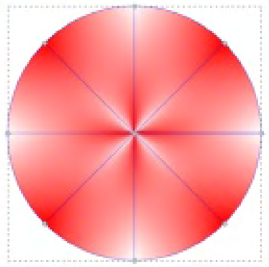


Now use the node handles to curve the outer edges, whilst also moving the inner nodes towards each other until they're all co-located at the center of your shape. You've just turned your mesh into a conical gradient – something hitherto impossible to create with SVG.

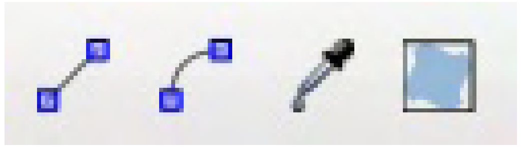


It's not a "true" conical gradient, of course. The XML markup is way more complex than would be necessary if SVG had native support. For a start, the center of the cone is defined not by a single node, but by eight of them stacked upon each other. If you want to move the point – or the bottom middle "node" (which is actually two co-located nodes) – you need to rubber-band select in order to ensure that all the nodes are moved at once. Otherwise your cone will quickly deconstruct and reveal the secret sauce behind it.

The reason for showing you this is that it helps to explain what happens when you use the second button on the toolbar to create a conical gradient, rather than a mesh gradient. The result may look different – for a start it will actually be circular, rather than the rough approximation in my hand-constructed version – but in reality it's still a mesh gradient, and if you want to manipulate it or move the nodes, you'll need to treat it in the same way as the manual version by rubber-band selecting any co-located nodes. Don't worry if you move a single node by mistake though; just CTRL-Z to undo your edit, then rubber-band and try again.

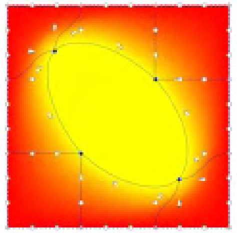


There's a block of four icons on the tool control bar that I haven't covered yet – although don't let that grouping fool you into thinking that they share related functionality. This is really just the "miscellaneous" bin of the toolbar, where odd buttons that don't have any friends are bundled together.

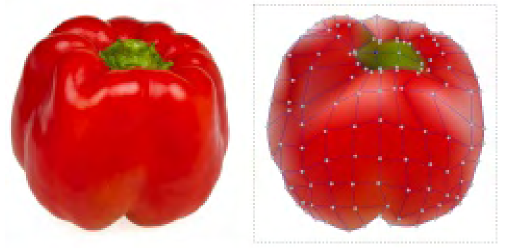


The first of these icons lets you switch selected lines between straight and curved modes. Straight lines have no Bézier handles, but behave a little more predictably when you drag them around, so sometimes it's easier to convert to a straight line, move the nodes, then convert back to Béziers for final tweaking. It's how I produced the manual version of the conical gradient above. Note that you use the same button to switch back to Béziers – the second icon might look like it's the right one for the task, but it's not.

Rather, the second button will change the length (but not the angle) of the Bézier handles of any selected nodes, in order to make the paths form an ellipse, if possible. This may seem like a rather arbitrary thing to do, but it does make it easier to smooth the transitions between different parts of your mesh, simplifying the creation of gradients like this:



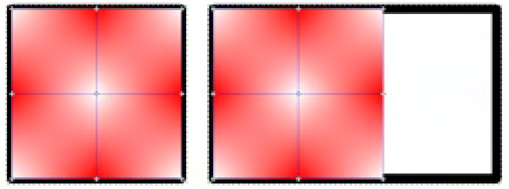
The third button is intended to save you a little work if you're using mesh gradients to reproduce the tones of an existing image. Clicking it causes any selected nodes to take on the color of the object behind them in the z-order. Consider trying to create a vector representation of a bitmap image: you can drag the nodes around to position them over key points in the image, press CTRL-A to select all the nodes, then click the button to set them to the respective colors. In this example I've used this technique to "trace" a bitmap image of a red pepper:



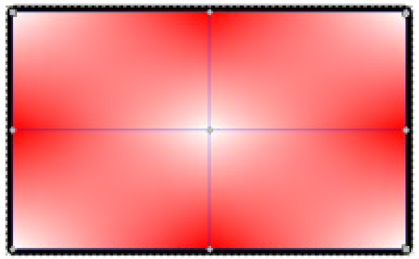
The mesh in this case was 10×10, but even then there aren't enough nodes in some areas to capture the details – the green stalk at the top is perhaps the most obvious example. This is an unfortunate limitation of mesh gradients: there's no way to subdivide individual patches in the mesh, to allow for fine detail where it's needed, while retaining broad swashes of color elsewhere. The workaround is to stack several meshes on top of each other, but this then becomes harder to manage.

If you do need more detail it is possible to split a row or column in two, creating more patches to work with. With the Mesh Gradient tool active, double-click on the line connecting two nodes and an extra set of nodes and lines will be inserted. Remember, though, that the operation affects the entire row or column, not just a single patch, so potentially adds many more nodes than you actually need. Currently there is no way to delete a row or column, so it's usually better to start with a slightly coarser mesh than you need and then split to add more nodes where you need to, rather than be stuck with an unnecessarily fine mesh that you can't simplify.

The last rogue button in the toolbar is used to address a small deficiency in the mesh gradients implementation: they don't track changes to the bounding box of your object. This is easiest to demonstrate by creating a mesh gradient on a square, then switching to the Node tool to convert the square to a rectangle by dragging the handle in the bottom right. You'll see that the mesh retains its previous dimensions.



With the newly rectangularised (that's a word, right?) object selected, switch to the Mesh Gradient tool and click on the final button in the miscellaneous bin. Lo and behold, the mesh is stretched to suit the new bounding box of the object.



That's the new Mesh Gradient tool in a nutshell. Now go ahead and create SVG files filled with meshes, and put them online for Google to index! Of course you'll face the catch-22 issue that browsers won't know what to do with the new content in your files, so you might also want to post an exported bitmap image until such time as the browsers (hopefully) catch up. Alternatively you could try using the JavaScript polyfill written by Tavmjong Bah (the author of the mesh gradient code in Inkscape): http://tavmjong.free.fr/SVG/POLYFILL/MESH/mesh.html

Mesh gradients are just one feature of SVG2 that is at risk if browser vendors can't be persuaded to support it. The only way that SVG will continue to have a strong future is if people use it. Whether your SVG files use mesh gradients or not, post them online and encourage others to do the same.

IMAGE CREDITS

https://commons.wikimedia.org/wiki/File:Red-Pepper.jpg

Mark uses Inkscape to create three webcomics, 'The Greys', 'Monsters, Inked' and 'Elvie', which can all be found at http://www.peppertop.com/

Written by Mark Crutch

Inkscape - Part 60

As you may have read elsewhere (you do read the rest of the magazine, right?), this issue marks the 10th birthday of Full Circle Magazine. As you may also have read, that means that we (the authors) have been given the freedom to do something a little different with our articles. So I've decided to make a cake.

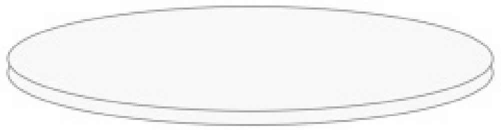
Ingredients:
• One computer, suitable for running Inkscape
• One recent version of Inkscape
• One pointing device, suitable for controlling Inkscape
• A monitor, to provide feedback as you make the cake
• (Optional) Artistic talent

Luckily, I had all of the ingredients already – with the exception of the last one. That means that my cake will be acceptable, if a little ordinary: if you have some artistic flair, I've no doubt you can make a much tastier creation.

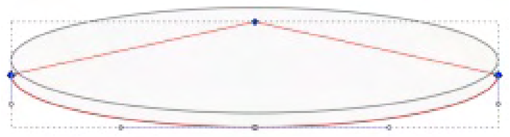
Method:
• Go to the shops and buy a real cake. Use it for reference, or simply eat it with a nice cup of tea or coffee to keep your spirits up when Inkscape crashes for the third time.
• Using the pointing device to control Inkscape, and the monitor to see the result, draw a cake.

What? You want something more detailed. Okay then...

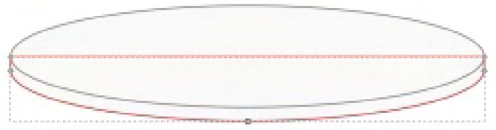
We'll construct our cake layer by layer, starting from the bottom and working our way up. But before we can even begin to think about whether we want a vanilla or chocolate sponge, we'll need a platter for the cake to sit on. Start by drawing an ellipse, then duplicate it (CTRL-D) and move the duplicate up a little using the arrow keys.



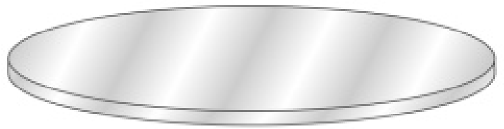
To make it look less like one disk stacked on another, and more like a solid platter, we need to straighten the sides of the bottom shape. Select it and convert it to a path (Path > Object to Path, or press CTRL-SHIFT-C). Double-click on it to edit the nodes and you should see that there are four nodes. Drag a box to select the top three, then convert them all to corner nodes using the button in the toolbar. You should now have three selected diamond-shaped nodes, and one square node. Finally click the "Make selected segments lines" button on the toolbar to give you an elliptical pie-slice.



Now double-click on one of the straight lines to create an extra node, then drag the two top nodes to the left and right, so that they touch the edges of the upper ellipse to give the appearance of straight sides. You may need to zoom in to position the nodes accurately, or enable snapping to smooth nodes (which will also snap to the quadrant points of an ellipse). If everything has gone well, you should be faced with something like this:



Switch back to the selection tool, click on the background to de-select everything, and you should find your shapes give the appearance of a thin disc. You may want to add a gradient fill to give it more depth, but I'm going for a fairly flat effect on my image (that's non-artist speak for "getting highlights and shadows right is a bit tricky!"). Instead, I used a repeating linear gradient running from white to light gray, to give the impression of a simple silver finish to the platter:

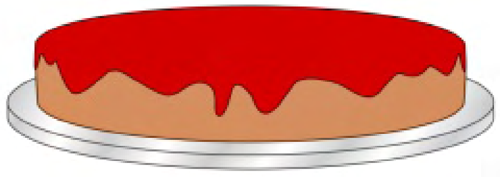


Building the first layer of the cake is very similar. Start by duplicating the top of your platter then scaling it down (hold CTRL-SHIFT as you drag the resize handles to get it to scale proportionally from the center). Give it a suitable fill and stroke (I've decided on vanilla sponge), then duplicate it, move it up with the arrow keys (a bit further this time), and repeat the steps above to form a layer of cake. You might want to bow the sides out a little to give it a better shape, but otherwise it's just a variation on making the platter.

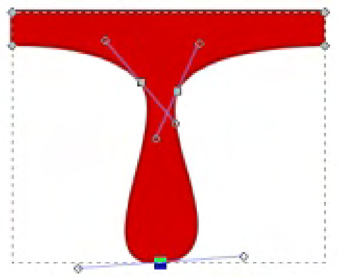
As any competent pâtissier knows, a cake is only as good as its filling. We'll go with the classic combination of jam ("jelly" to US readers) and cream. Obviously, once we spread jam all over the top of the cake, we won't be able to see that top ellipse any more, so we may as well change its fill and stroke to shades of red, and use it as the basis of the jam layer instead.

This is where a little psychology kicks in: when presented with the dessert trolley at a fancy restaurant, we tend to be attracted to the perfectly formed, beautifully mirror-glazed, geometrically precise offerings, rather than those which look like the chef may have accidentally sat on them. When drawing an image, however, a flat disc of jam lacks the appeal of an overloaded cake, gloopily dripping great gobs of strawberry goodness from its sides. So gloopy gobs it is for our image. Which means more node editing.

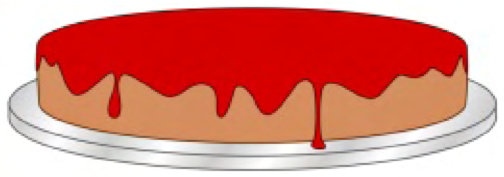
Convert your jam ellipse to a path, but this time we'll be working with the bottom half of the shape, not the top. Select the bottom three nodes, leaving the top one alone, but don't convert them to corner nodes this time: rather, press the INSERT key a few times to create more smooth nodes between the existing ones. Click the background canvas to de-select them all, then start dragging individual nodes downwards, working from the center out, to give the appearance of the jam flowing down the side of the cake. Move some of the nodes to the left or right, and add or delete some to give it a more random appearance – dragging the paths rather than the nodes can also help. Try to stick to smooth nodes – there are no sharp corners in our jam! Watch out when removing nodes, as the adjacent ones may be converted to corners automatically. With a bit of manipulation you should get something similar to this:



To really get a gloopy effect, we'll want to add some drips running down the side. The basic form of a drip is created by adding a couple of nodes to let you pinch in the "neck", with one in-between that you can drag down to form the base of the droplet itself. Make that center one into a symmetric node to save yourself a little editing work, and position all three to give you the desired appearance.



A drip running all the way down to the platter will need a couple more nodes to handle the transition from vertical to horizontal, but otherwise it's the same idea. With those additions, here's the cake so far – it's starting to look good enough to eat!



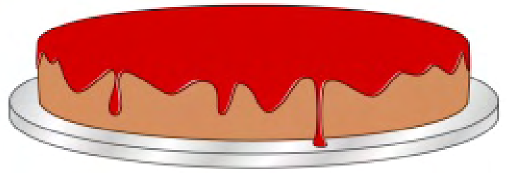
Despite my earlier avoidance of highlights and shadows, there's no denying that the best jam has a glistening surface that makes ours look somewhat lifeless. An obvious candidate for adding specular lighting would be to use the filter primitive of the same name – but that's probably overkill for the result we're looking for, and would likely take a lot of trial-and-error to even come close. For the comic-style image we're creating, hard-edged highlights are frequently a better choice, and can often be made using copies of the very shape you're trying to highlight.

Start by duplicating your jam path and moving the duplicate up, away from the cake, by using SHIFT-UP ARROW. The advantage of moving it in this way is that the final result can be moved back

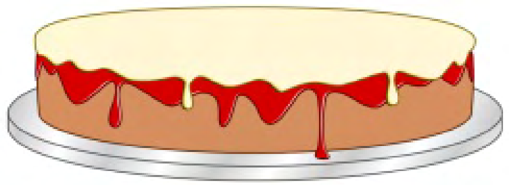
contents ^

precisely by pressing SHIFT-DOWN ARROW, even if you've changed the zoom level whilst working. Remove the stroke from this copy. Now duplicate the new copy (so that you've got two duplicates of the original, stacked on top of each other), and give the top one a contrasting fill. Holding the ALT key, use the cursor keys to shift the top copy slightly to the left and down, so that you have a thin slice of the underlying red path peeking out at the right of each descending loop.

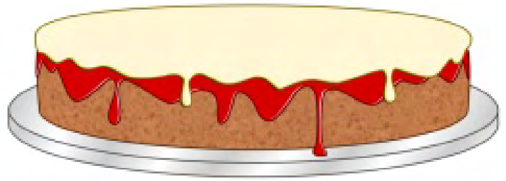
Drag a selection box around both duplicates (that's why we moved them away from the main part of the cake), and ensure that the status bar shows two paths are selected. Now use the Path > Difference menu entry to remove the top shape, leaving only those thin edges. Use SHIFT-DOWN ARROW to move the resultant path back into place, then set its fill to white. Holding the ALT key, you can now use the arrow keys to precisely position the highlights until most are in the right place. Now use Path > Break Apart to separate each highlight into its own object, making it easier to fine tune them, by moving, editing or even deleting those that don't look quite right.



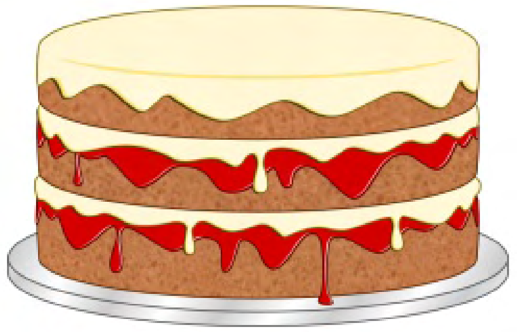
Repeat the jam layer, but with a pale yellow fill color to represent cream. Highlights don't work so well on something so pale, so, instead, create some lowlights – shifting your duplicate in the opposite direction, and using a slightly darker fill color to add a subtle shadow to each glob of cream. You can also give it a little more depth by simply putting a dark duplicate below the cream path and shifting it so that it peeks out ever so slightly.



That's the first layer of our cake pretty much finished, but there is one final trick that will make the sponge look more… well, spongey. Duplicate the beige sponge object and move it up away from the cake. Tweak the color to make it a darker shade, then duplicate it. Open the Fill & Stroke dialog, and select a pattern fill, specifically the "Sand (bitmap)" pattern. Depending on the size you're drawing at, the grain of the image may be too coarse or too fine, so, if necessary, double-click on the object to gain access to the handles for scaling the pattern. These appear as a cross, circle and square, anchored to the top-left of the page. Use the cross to reposition them, if you wish. Dragging the square will change the scale of the pattern, whereas the circle will let you rotate it (which you'll need later on). By the time you're finished, you should have a dark beige cake base completely covered by a course, sandy duplicate. Select both, then use Object > Mask > Set to reduce the pair to a single, blobby, brown shape. Move it back down over the cake, and use the Page Down key (or the button on the toolbar) to drop it down in the z-stack until it's above the cake base, but below the jam.

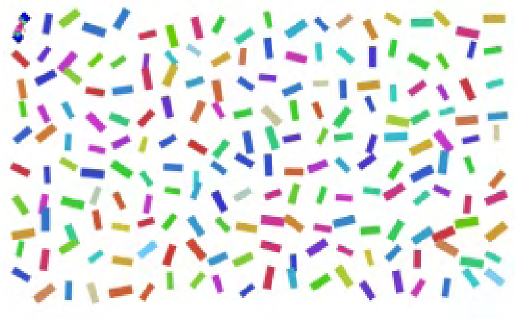


Repeat the steps above to create more layers of cake, jam and cream. On your topmost layer, add a little shadow to the front edge by using Path > Difference between two ellipses to create a thin crescent of color which helps to define the shape.



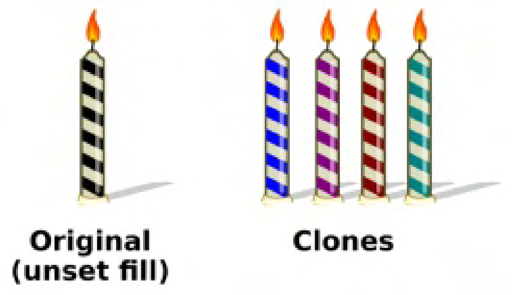
As this is a celebratory cake, it should probably have a little more decoration. How about some sprinkles? There are a few ways to approach this, but I opted to use tiled clones (see parts 33 to 36 of this series). My parent object was just a short straight line which I cloned into 10 rows by 20 columns. Each row and column position was randomised, as was the rotation of each clone. Once I achieved the sort of distribution I was looking for, I unset the stroke on the original object so that I could use the Color tab of the Tiled Clones dialog to set a random hue to each sprinkle, giving me this result:

contents ^



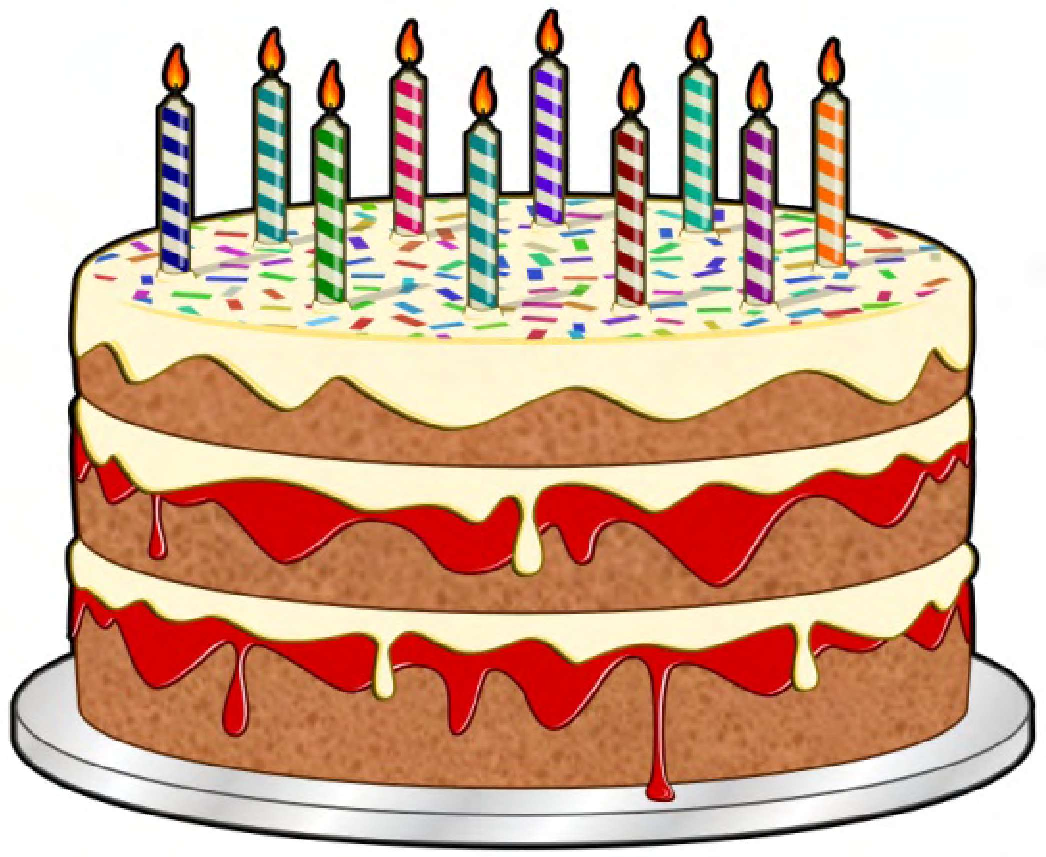
Grouping the clones, then scaling the group vertically gives the right sort of perspective appearance. Clipping with an ellipse results in a neat top for the cake.

As this is specifically a birthday celebration, candles are also in order. I began by creating a single candle using the same basic techniques of drawing primitives, converting them to paths, and tweaking the shape using the node handles. To add stripes to the candle, I used the same approach (of a patterned fill masking a flat color) that I used for the texture on the cakes. This time the fill was “Stripes (1:1) White”, and I used the previously mentioned pattern adjustment handles to not only scale, but also rotate the pattern. The object behind was given an “unset” fill, then the whole candle was grouped. Because of the unset fill, it was possible to clone the group and give each clone its own individual color:



After placing the candles onto the cake, there was one final step to really make the image stand out: in classic comic-strip style, I wanted to give it a thick black outline. To do this, I selected the whole image and duplicated it, then moved the duplicate well away from the original. I deleted the sprinkles, highlights and lowlights, since they wouldn’t have an effect on the silhouette. Then I repeatedly used Edit > Clone > Unlink Clone and Object > Ungroup – until the status bar showed that there were no more clones or groups in the selection. Finally, Path > Union joined all those separate objects into a single shape, which I could fill with black as a silhouette. I moved it back over the original image, gave it a thick black stroke, then sent it to the back of the z-stack.

So here’s the result of all that work: **Happy 10th birthday Full Circle Magazine!**



Mark uses Inkscape to create three webcomics, 'The Greys', 'Monsters, Inked' and 'Elvie', which can all be found at http://www.peppertop.com/

Written by Mark Crutch

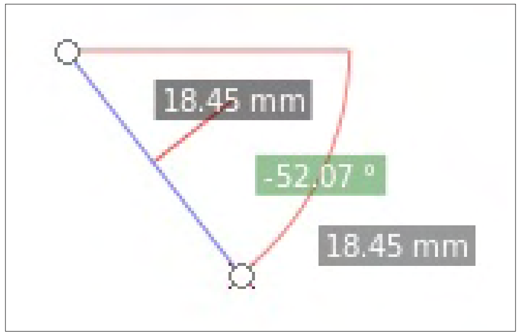
Inkscape - Part 61

Prior to last month's celebratory distraction, I was advocating that readers should install Inkscape 0.92 and begin posting files that use its SVG 2 features online, as a way to help show the browser vendors that there is a demand not only for SVG on the web in general, but for a format that lives and grows to encompass more capabilities than those that were baked into the specification over 15 years ago. The release of 0.92 also offers an option to turn off "spacebar panning", which was my biggest complaint with 0.91. So, I have finally retired my old 0.48 system and move entirely to 0.92. Which gives me a prime opportunity to spend a few months introducing some of the new features of 0.91 and 0.92 that I haven't covered in any real detail to date.

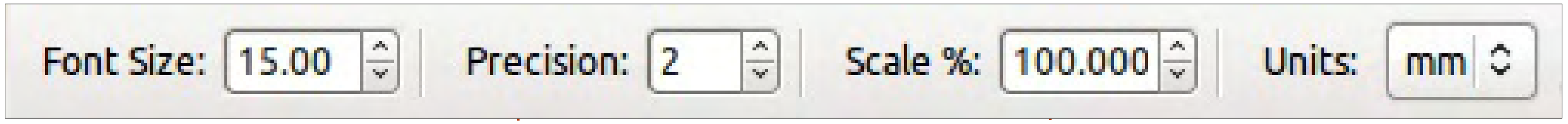
Many of the changes from 0.48 aren't things that are directly reflected in the user interface: the usual round of performance improvements, bug fixes and compatibility tweaks don't always make for headline news, but are vital to the stability and capabilities of the program, nevertheless. But I'll be concentrating more on the new UI features that, as a user, will most affect your day-to-day use of Inkscape. I'll begin with a completely new tool that can be found on the main toolbar, just after the Zoom tool: the Measure tool (also referred to as the Measurement tool).

You can, of course, activate this tool by clicking on the button – or you can use the keyboard shortcut, which is "M" by default. This tool had limited utility in 0.91, but gained a lot of extra features in 0.92, so I'll be describing the latter. In either version, at its simplest, the Measure tool does exactly what you would expect: it lets you measure the distances, and angles, between parts of your drawing. In its basic mode, the measurements are ephemeral – you can make only one measurement at a time, and as soon as you switch to a different tool, the measurement disappears. The new features in 0.92 include some slightly clunky workarounds for both of these issues, but it's still not the same as the sort of dimensioning tool you would find in a fully fledged CAD application, as will become clear.

With the tool active, start by clicking and dragging a line on your canvas. You'll see that there's a blue line that follows the mouse, and red lines marking out its angle from the horizontal. There doesn't seem to be a way to measure the angle from the vertical, let alone any other arbitrary datum line. Your line will also be annotated with its overall length, which could appear on it twice, depending on your settings. The midway annotation shows the length of the line segment, whereas the annotation at the end shows the total length of the line. On a blank canvas, these will show the same value, as there's only a single line segment to consider.



If you now click the mouse somewhere else, the starting point of your line will move to the location you click on. This can be useful for taking several measurements from the same datum point, but it does seem counter-intuitive to me that it's the start that moves, not the end. If I wanted to take several measurements from the same point, I would naturally expect to click first on the common datum, and then click on each point I wish to measure. Instead, Inkscape requires you to drag from the first point back to the datum, and then click on each subsequent point. You can also drag the handle at either end of your line to move it to a different location, so if you're happy to drag rather than click you can use that approach to work with the datum as the first point.

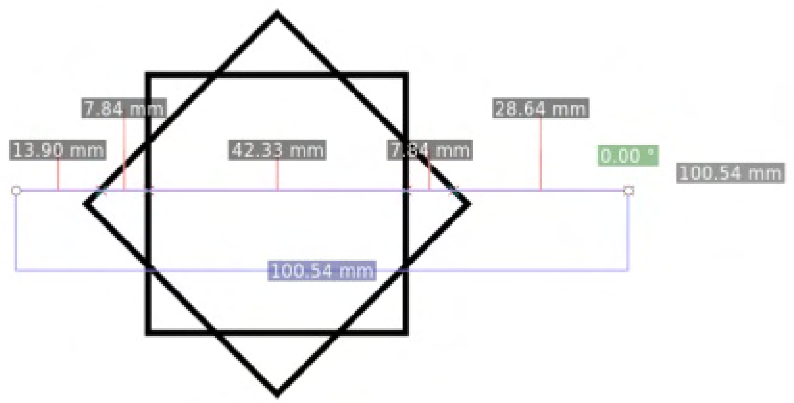

Font Size: 15.00 Precision: 2 Scale %: 100.000 Units: mm

As you might imagine, holding CTRL whilst dragging the initial measurement will constrain it to particular angles based on the rotation steps setting in the Inkscape preferences. This is particularly useful to limit your measurements to the horizontal or vertical direction. It doesn't work so well when dragging the endpoints around, though, as it tries to constrain the angle of the measurement, not the direction of your mouse movement, which isn't always what you need.

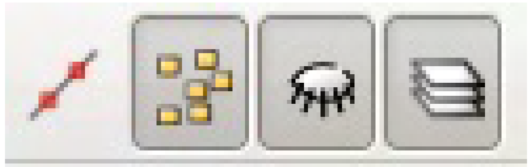
The labels themselves can be modified a little using the first few widgets on the tool control bar:

Font Size and, at the other end, Units, are pretty self-explanatory. Precision dictates the number of decimal places that are shown. The Scale control adds a multiplication factor to all your lengths; setting this to 50% will halve the values shown, whereas 400% would quadruple them. It's intended for use where your drawing isn't a 1:1 representation of the original, although it would be useful to have a mode that lets you set this value by measuring a line and entering its real length. That would make it much easier to trace a bitmap diagram, where the actual scale is not known but one of the original lengths is. Instead you'll have to place your measurement, then adjust this control iteratively to suit. For most uses, however, just leaving the scale at 100% will be fine.

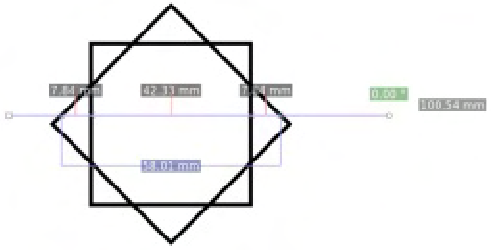
I mentioned earlier that drawing a measurement on a blank canvas only shows a single length. What happens if we draw on a non-blank canvas instead? Here's the result of drawing a horizontal line with arbitrary start and end points, over a couple of shapes:



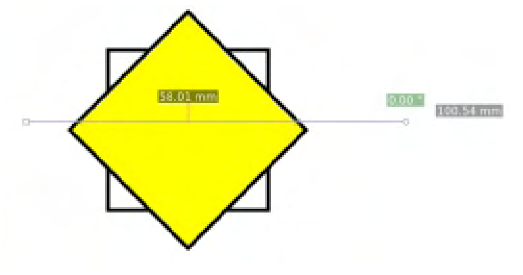
Notice how Inkscape, rather cleverly, marks the length of each segment of the measuring line between the points at which it crosses other shapes. It also shows the overall length of the line. But, in this case, my endpoints were arbitrarily chosen, so I'm probably not terribly interested in the fact that the line extends 13.90mm to the left and 28.64mm to the right. Back to the tool control bar once more…



These four buttons determine which points are considered when working out the measurements. The first has a tooltip of "Ignore first and last", and toggling that on does exactly what it suggests: the first and last points on the measurement line are dropped from the main part of the display, making it clearer to see the overall length of the section I'm concerned with (though the overall length of the measurement line does still appear at the far right):

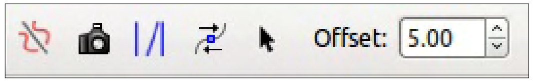


The second button on the toolbar turns off the intermediate measurements, making for a much clearer display when you just need the overall length between a couple of points. The third button has a similar effect for any intersections that are hidden behind other objects. By making my topmost shape opaque you can see the difference when this option is toggled off:



In this particular case, the result is the same as if the second button was toggled off, but that won't always be true. If you use the measurement tool on a complex drawing, with many overlapping objects, the output can become very hard to read if you don't use this button, or the next one, to reduce the number of items that are included in the measurement process. As for that next button, it simply determines whether the measurement will be limited to objects in the currently selected layer, or in all layers. NOTE: If you turn this button off, Inkscape will, indeed, only show you measurements based on objects in the current layer, but the measurements don't update live as you switch between layers: you need to either make a change to one of the tool controls (such as toggling a button on and off again), or adjust the position of one of the endpoints of the measurement.

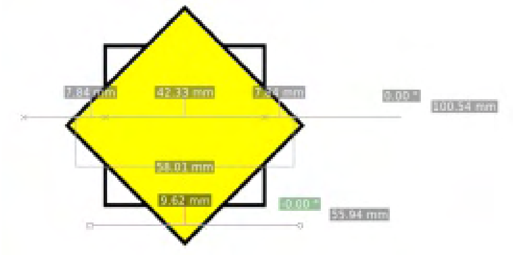
The last few widgets on the tool control bar offer some options for extra things you can do with the measurements:



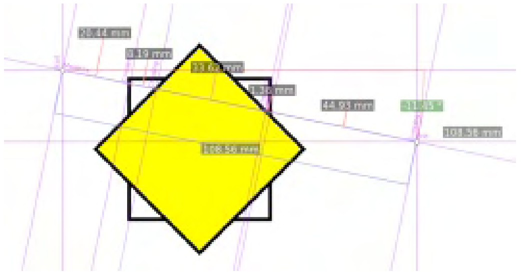
Offset: 5.00

The first button swaps the endpoints of the measurement line; in doing so it also switches the angle being measured from the inside to the outside angle (or vice versa). No, I don't know why it has an icon that better represents reflection than reversal.

The second button's icon makes a bit more sense: the camera takes a "snapshot" of the current measurement, allowing it to hang around while you make a second measurement somewhere else. Despite the icon, however, Inkscape doesn't use the word "snapshot", preferring to refer to it as a "phantom measurement". Whatever the terminology, it renders your first measurement in gray – both lines and labels – whilst your second measurement still uses the normal colors. This feature does make it easier to compare measurements, but you can still only have one phantom snapshot, and one live measurement: if you click the button again, the existing phantom vanishes, and the current live version is converted into a snapshot instead. In this example, you can see that my previous measurements have been converted, and I also have a live measurement going on at the bottom:



The next button on the bar will create guides that correspond to the key points of your measuring line. Be warned, this can easily create lots of guides – you'll get one that follows the direction of the measurement path, and a guide for each labelled path intersection that is drawn perpendicular to the measurement line. In addition, there will be a horizontal and vertical path for the endpoints of your line, though not for the intersection points. It's important to use the earlier tool control bar buttons to reduce the number of intersections being measured, especially if you're working with a complex drawing, otherwise you can easily end up with way more guides than you wanted. You can, at least, immediately use Edit > Undo if you do make a mistake with the settings. If you want to remove just a few of the guides, remember that you only have to hover the mouse over a guide until it changes color, then press the Delete key to remove it – a quicker option than deleting via the guide's dialog.



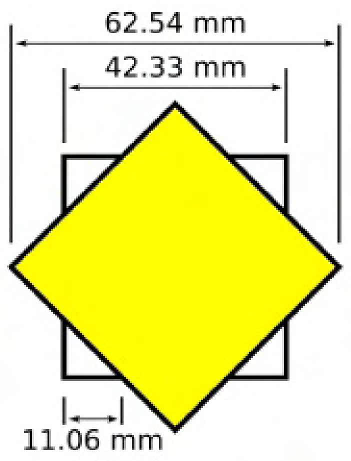
Should you need to see more than two measurements at once, the next button provides something of a solution. Clicking it will convert the current measurement layout to a group of real objects. You can then enter the group and manipulate them as you would any other shapes – including deleting unnecessary measurements or changing the text of the labels. Because they're real objects, they'll still be visible when you draw another measurement. But equally, as real

contents ^

objects, they can be included as crossing points in any measurements themselves, should your new line cross them, which can lead to a confusing collection of lines and labels on the screen. One possible solution to this is to put your converted measurements onto a separate layer and turn off the “Measure all layers” toggle.

You might think that this button makes for a viable way to mark up the dimensions on a technical drawing, but there is a (slightly) better method. The penultimate control on the toolbar, “Mark Dimension”, will render a line with arrowheads at either end which runs the whole length of your measurement path, but with a slight offset, the amount of which is set by the last control. The length of the dimension will also be created as a text object alongside the new line; it’s larger than the normal measurement labels, but can still be adjusted by a relative amount using the first control on the toolbar.

At first this might seem like a more limited choice than the previous “Convert to item” button, but, because it doesn’t create a whole load of superfluous elements, it can be used more rapidly without requiring a lot of cleaning up afterwards. The secret is to use snapping when placing the start and end points of your measurement line, so that it stretches exactly over the dimension you wish to measure. The orientation of the text will depend on the direction of the measurement path, so, if it’s upside down, simply undo the operation, click the tool control button to swap the ends of the path, then “Mark Dimension” once again. You can quickly mark out a drawing in this way, but do note that, in the following example, I had to enlarge the arrowheads and draw the vertical projection lines myself (the latter was made easier with the Measurement Tool’s “Convert to guides” option):



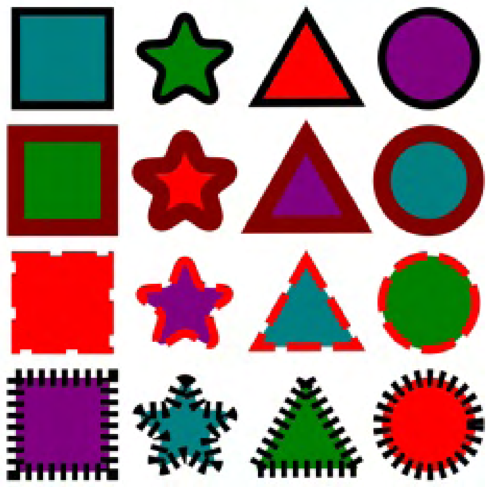
Perhaps the biggest limitation of using this method to dimension a technical drawing is that the dimensions are just lines and text objects, with no relationship to the objects they measure. If you change the size of an object, the dimensions won’t update on their own: you’ll need to either modify or re-create them. This, combined with the additional steps needed to produce the dimensions in the first place, is the main reason why, for anything more than the most basic of diagrams, you’re much better off using a real CAD program for technical drawings. Handy, then, that FCM is running a tutorial series on FreeCAD – the program that I turn to myself when I need to draw something more technical than artistic.

Mark uses Inkscape to create three webcomics, 'The Greys', 'Monsters, Inked' and 'Elvie', which can all be found at http://www.peppertop.com/

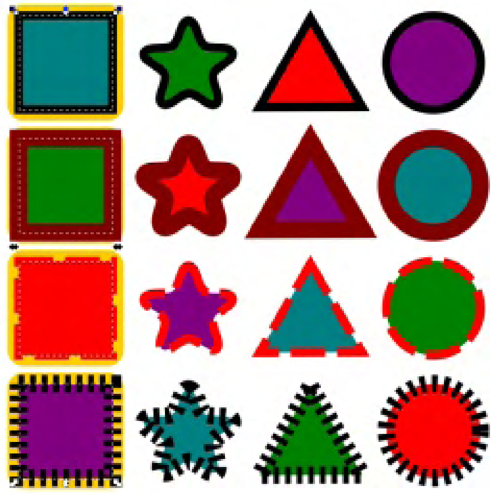
Written by Mark Crutch

Inkscape - Part 62

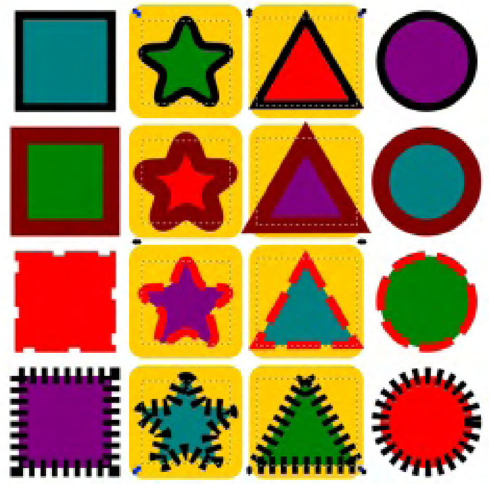
One area of functionality that has received a huge boost in 0.91 and 0.92 is finding and selecting objects. There are many situations in which you may need to find similar, or identical objects in a busy drawing, perhaps to delete or change them all at once. In previous versions, this was a chore, especially if you had to make the same selection multiple times. So let's look at the various ways in which recent releases make this easier. I'm going to use this contrived grid of objects to give you a feel for how the new tools operate:



The first few items all live under the Edit > Select Same submenu, and offer several different ways to select all items which share a particular visual style. To use any of them, you must first select one or more objects whose close relatives you wish to also select. As a first example, if I select the teal square at the top left of my grid, then use Edit > Select Same > Object Type, all the other squares are selected. In my image I've given the selected items a gold colored background to make them stand out a little better, but you won't see that in normal use.

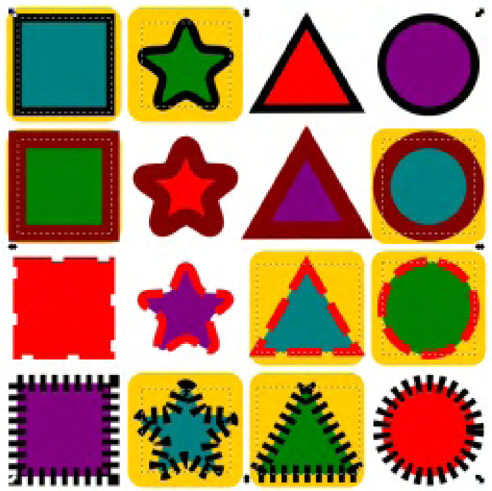


One important thing to note is that selecting the same "object type" purely refers to the type of underlying Inkscape object – so squares and rectangles, or circles and ellipses, are considered equivalent. You can see this effect if I select the green star on the top line, then use Select Same > Object Type. Not only are the stars selected, but so are the triangles, as they were all created using the 'star' tool.



Frustratingly, the select by type option doesn't work terribly well with multiple items selected. If every object in the selection is already the same type, it will work to select the other objects of that type; but if any one of them is of a different type then, rather than selecting all the objects of either type, everything is de-selected instead.

Select Same > Fill Color has no such problem: if I have both the teal square and the green star selected, then all the objects with a teal or green fill will be selected by the command.



Note, however, that selecting by color is extremely specific: it will only select items whose fill

contents ^

color is absolutely identical to your starting object. There's no way to set a threshold, in order to select all the items with approximately the same shade of green, for example.

There's also a Select Same > Stroke Color option, which does the same thing, with the same exacting requirements for a match, but based on the stroke color rather than the fill color. For example, selecting the teal square once again (before executing the command) will select all the shapes on the top and bottom rows; they all have a black stroke, even though the stroke thickness and dashes are different between them. Once again, starting with a multiple selection will result in a cumulative set of items whose stroke colors match any of the objects in your initial line-up.

The counterpoint to selecting by stroke color is to select by Stroke Style. This matches objects only if they have the same stroke thickness and dashes – including the same dash offset, which seems a little strict, in my opinion. Stroke color, line caps, join type and markers don't seem to have an effect, whereas the mitre limit does. The choice of what properties do and don't contribute to a "matching" line style seem rather arbitrary and counter-intuitive, which may limit the usefulness of this option in many situations.

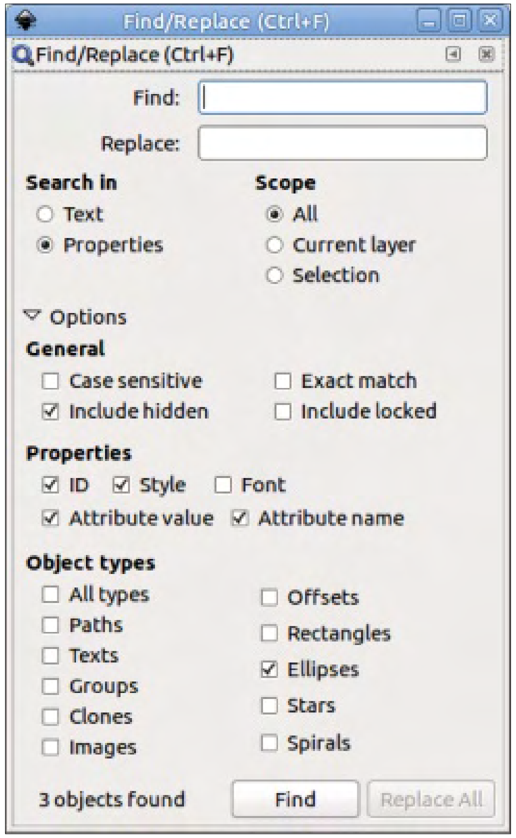
Even more limiting, however, is the Select Same > Fill and Stroke menu entry. This will match items only with an identical stroke, but where both the fill and stroke colors also match. It's the digital equivalent of a club doorman not only refusing entry for wearing sneakers, but actually only allowing entry to people with exactly the same shoes as him, in the same color and style – and only then if they're also the same size.

The next tool we'll look at this month is the recently souped-up Find/Replace dialog, which can be opened from the Edit menu or by pressing Ctrl-F. The key to working with this dialog effectively is to understand that all it's really doing is a search (and possibly a replace) of text within the XML code that makes up an Inkscape document. So, first, a quick recap on XML using a heavily abridged snippet of code representing a text object in Inkscape:

```
<text id="text1178">

  <tspan

    id="tspan1180"

    style="font-style:normal;

          font-weight:bold;

          font-family:Arial;

          -inkscape-font-
specification:'Arial Bold';

          fill:#000000;"

    x="320.68646"
y="86.745667">

      This is some text

  </tspan>

</text>
```

XML represents a tree structure in text. Here you can see that we have an opening <text …> element, which gets closed by the </text> line at the end. Inside of that is a single <tspan …> element, again being closed near the end. Inside that is the text itself, "This is some text". This is between the opening and closing <tspan> tags, but is not a tag itself. This piece of literal text is what XML refers to as a 'text node', whereas <tspan> and <text> are 'element nodes'. The element nodes also have name=value pairs of extra data, referred to as 'attributes'. In the <text> node, for example, the id="text1178" is an 'id' attribute, with a value of "text1178". The 'style' attribute is a little special – it holds CSS style data, which itself takes the form of a list of 'property:value;' pairs.

With that information in mind, let's take a look at the Find/Replace dialog, with its Options section expanded:



Find/Replace (Ctrl+F)

Find/Replace (Ctrl+F)

Find:

Replace:

Search in
Text
[✓ Properties]

Scope
[✓ All]
Current layer
Selection

Options

General
Case sensitive
Exact match
[✓ Include hidden]
Include locked

Properties
[✓ ID] [✓ Style] Font
[✓ Attribute value] [✓ Attribute name]

Object types
All types
Offsets
Paths
Rectangles
Texts
[✓ Ellipses]
Groups
Stars
Clones
Spirals
Images

3 objects found   Find   Replace All

As you might expect from a Find/Replace dialog, the first two fields allow you to type in text to be searched for and, optionally, text to replace it with. The search will match against substrings, unless the 'Exact Match' checkbox is ticked, so searching for "Fred" will match against Fred, Freda, Winifred and Alfred – although the latter two will match only if the 'Case Sensitive' checkbox is cleared. Interestingly, you can search with this field left blank. As you might guess, an empty string matches against everything, but the match is further restricted by the choices in the 'Object types' section at the bottom. Usually you would probably leave this set to 'All types', but, by setting specific checkboxes and leaving the search field blank, it provides a fast way to select all the objects of specific types – a workaround for the Select Same > Object Type's restriction on finding only one type at a time.

Assuming the more common case, where a search string is entered, the 'Search in' radio buttons determine whether the XML text nodes will be searched, or the attributes on the element nodes (though the latter is labelled as 'Properties' in the Inkscape UI). Use the former to search and replace text content – handy if you're using Inkscape as a poor-man's desktop publishing program (though I do recommend that you learn how to use Scribus if you need to do any significant text layout work). Use the 'Scope' buttons, and the checkboxes in the 'General' section to limit your search a little, if necessary.

Switching the 'Search in' control to 'Properties' provides a great deal more power, if you're comfortable enough with the insides of an SVG file to know what to look for. By 'Properties' it means attributes and attribute values, including CSS properties. Making a distinction between these terms might seem a little petty, but understanding them is key to making effective use of this section of the dialog. There you'll find checkboxes to tell Inkscape which parts of the XML it should search:
• ID – Only search the value of the 'id' attributes.
• Style – Search for CSS properties and CSS values within the value part of 'style' attributes.
• Font – Presumably this searches for only CSS font names, but I was unable to get it to work at all on my test file. Use a 'Style' search instead.
• Attribute Value – Search within the values of attributes. This includes searching the values of id and style attributes, even if the earlier checkboxes are blank.

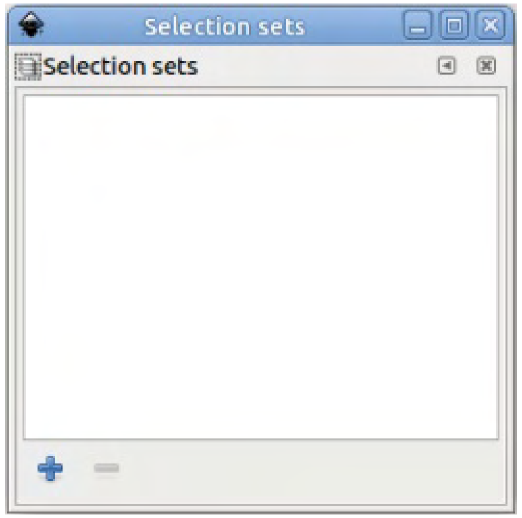
• Attribute Name – Search for elements with a particular attribute name. Not as useful as searching for values.

Usually, searching for an attribute value is sufficient. It will find matching IDs, styles and font names, without you having to understand how they're stored in the XML. Only if it finds too much, is it worth switching to search only IDs or styles. Searching for attribute names is never really necessary for normal users, but might have its place if you're using Inkscape to work on graphics for a web application that carry additional custom metadata.

There are a couple of things to beware of when using this dialog in 'Properties' mode, however. Firstly, it can end up selecting objects you didn't expect. Consider an element with a red stroke; later you change the stroke width to zero, so it isn't visible any longer. If you search for "#ff0000" hoping to select all the elements with a red fill, you'll also select the object with no stroke. Although it might not be visible, there will still be a color stored for the stroke in the style attribute, which is enough for it to become part of your selection.

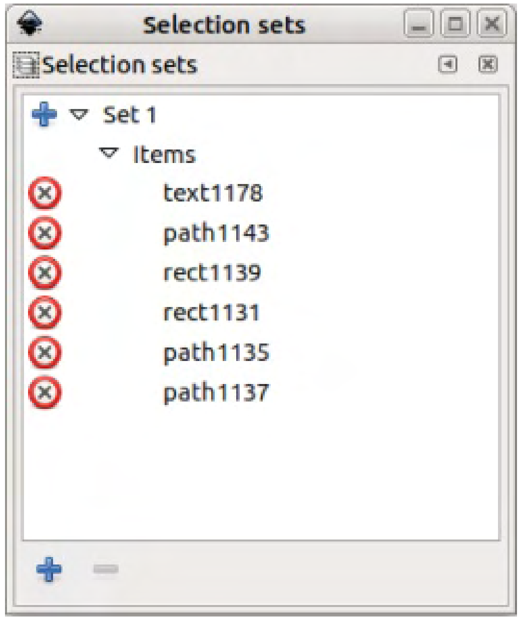
The second warning is with regard to the 'replace' field: performing a find and replace in text mode is generally safe; doing so in properties mode could have unforeseen results. Thankfully, Inkscape is sensible enough to stop you performing a replace operation if the 'Attribute name' checkbox is selected, but arbitrarily replacing strings within attribute values can be almost as destructive. It might seem safe enough to replace your black fills with white, by searching for "000000" and replacing it with "ffffff", but a series of six zeroes could easily appear in the coordinates of your objects, where hexadecimal values are an error that could break your entire drawing. If in doubt, save a backup of your file first.

Having used the previous tools to make a selection, Inkscape now also provides a way to save that selection for later use. The Object > Selection Sets… menu item opens a rather empty dialog for managing stored collections of selections:



The most important thing to note about this dialog is that the "+" button at the bottom does not create a set containing your current selection. Instead it creates a new, empty selection set which has another "+" button next to it. Only when you click this other button will your selection be stored in the set. The same button can be used later to add another selection to any existing items in the set.

If you click on the triangle next to a set, hoping to view the elements within it, you'll quickly find that there's a rather useless intermediate level, labelled "Items", that you also have to expand. If you click the triangle next to the "Items" entry to expand it, you'll finally get to a list of the objects in your selection.



Clicking on the cross next to an item in the list will remove it from the set. Clicking on a name in the list will select it, whereas clicking on the Set itself will select all the items in that set. Unfortunately, clicking on a Set doesn't highlight all the items within it, so, if you want to re-select all the items except one, for example, you have to highlight each entry separately whilst holding the CTRL-key. You can even use the CTRL-key to select multiple Sets, or combinations of Sets and individual items. What you can't do, however, is select something from this dialog, then add to (or remove from) the selection by CTRL-clicking on objects on the canvas, which is a pity as it effectively turns Selection Sets into a separate selection mechanism rather than something which integrates seamlessly into your normal Inkscape workflow.

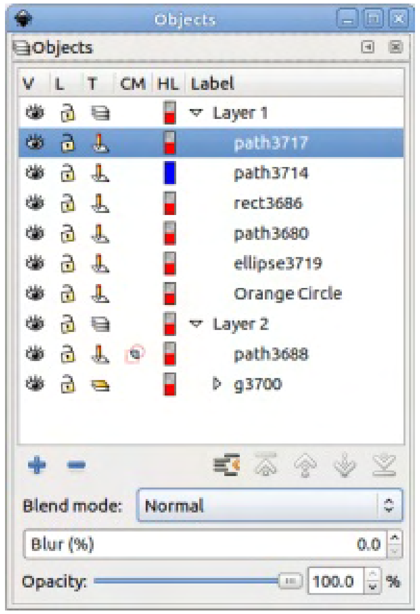
Selection sets are preserved when you save a document, however they do get cleared when the Edit > Clean Up Document menu is used. Nevertheless, they have their uses – for example, as another means to work around the restriction on Select By Type only working with a single type at a time. If you want to select, say, all the circles and rectangles, you could use the Select By Type option twice, adding each resultant selection to the same Set.

The new and extended selection tools in 0.92 are a welcome addition, especially when working with complex documents. They do have their limitations and idiosyncrasies, but hopefully they'll continue to mature in future releases to provide even more capabilities.

Mark uses Inkscape to create three webcomics, 'The Greys', 'Monsters, Inked' and 'Elvie', which can all be found at http://www.peppertop.com/

Inkscape - Part 63

A frequently requested feature, particularly from users coming to Inkscape from other vector tools, is a dialog that shows a hierarchical tree of the objects in a drawing. This is now present, as of version 0.92, via the Object > Objects… menu entry. This feature was actually back-ported from Ponyscape, a fork of Inkscape that is no longer being developed, so bravo to the Inkscape developers for merging it back into the program, and thanks to the original Ponyscape developer for creating it in the first place. Here's what the dialog looks like, with a few objects in a drawing:


Objects

Objects

V L T CM HL Label

Layer 1
path3717
path3714
rect3686
path3680
ellipse3719
Orange Circle
Layer 2
path3688
g3700

Blend mode: Normal

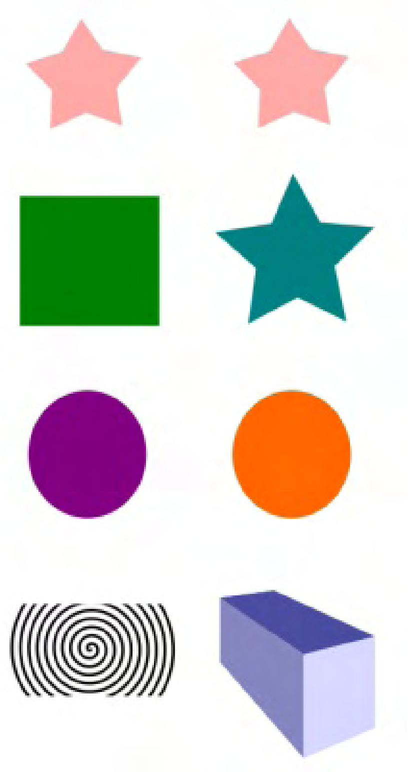
Blur (%) 0.0

Opacity: 100.0 %

It shares some similarities with the Layers dialog (see part 9 of this series), and can largely be used as a replacement for it if you wish – although I prefer the simplicity of the old dialog. At the top is the hierarchy of objects in your drawing, with groups and layers displayed as collapsible entries that can be opened to reveal the objects within them, or closed to hide the clutter. The lower section provides buttons for adding layers, deleting selected objects, "Collapse All" (more on that in a moment), and buttons for moving individual objects up or down in the list (which, in turn, moves them up and down in the z-order). There's a pop-up for selecting a blend mode – which can now be easily applied to any object. Previously, only layers could easily have blend modes applied; other objects required a trip to the filter editor, so this is a welcome addition. It's also possible to set the blur and opacity of the selected object, group or layer.

As for that "Collapse All" button, what it actually does is collapse every top level layer or group except the one that the currently selected object is in. This is a common theme for this dialog – right-click on an object in the tree and you'll be presented with a wide range of options in a context menu, in which several choices (those from the "Solo" and "Lock All" sections) actually work on top-level layers and groups, regardless of how deeply nested the object is that you clicked on.

Each object is listed by its "Label", which was previously only settable in the Object Properties dialog or the XML Editor. Now you can simply double-click on the entry in this new dialog to change the label – a significant usability improvement. But unless you're fastidious about changing the default labels assigned by Inkscape (as I have done in the case of "Orange Circle") you could well be in for some surprises. Take another look at the screenshot of the Objects dialog, considering the labels of the objects there. Now take a look at the canvas that produced that list:

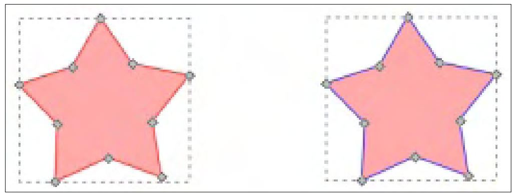


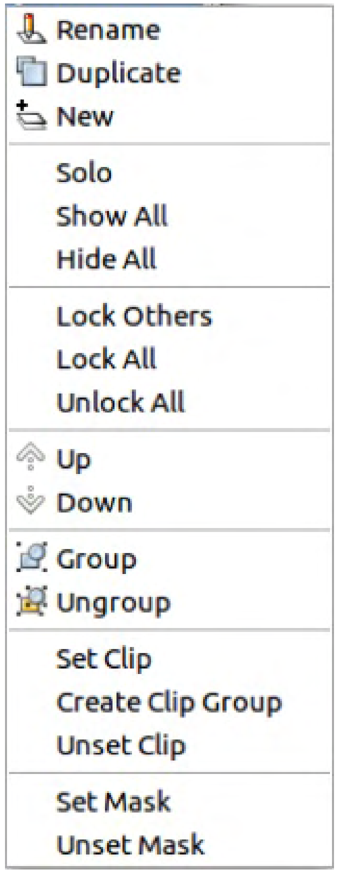
Notice that the stars are labelled as "paths", as is the (clipped) spiral. The 3D box has a label of "g3700", because in SVG terms it's actually a group, so Inkscape labels it as such. Which path corresponds to which star? The only way to tell is to select each one and see which object becomes selected on the canvas. I hope that a future release of Inkscape will add an extra column to show the "Inkscape type" of the object (e.g. star, spiral or 3D box), and preferably a small preview image of the object as well.

To the left of the object tree, you'll have noticed that there are five other columns, with terse titles that only make sense once you've hovered over each of them to read the tooltip. The first and second should be familiar from the Layers dialog – they toggle the visibility and locked state of each object. At last there's an easy way to unlock individual objects: many new users have found themselves tempted by the Lock option in the Object Properties dialog, only to then find that they could no longer select the object to unlock it! The third column, nominally "Type", holds an icon to indicate whether the item is an object, layer or group. As you may recall, SVG doesn't actually have a concept of layers, so Inkscape implements them as groups with some extra metadata. An interesting extra in this dialog is the ability to click on the Type icon of a layer to turn it into a group, or vice versa.

The next column is supposed to show an icon to indicate whether an object is clipped and/or masked. In my screenshot, you can see the effect on the clipped spiral, however my copy of Inkscape does not show an icon there for masked objects, despite the claim of the tooltip. In this case the icon is purely informative; the dialog doesn't provide any additional capabilities for working with clipping paths or masks, so there's still no way to edit masks or some types of clip paths without releasing them first.

The last column is rather specialised, and will probably be rarely used. Within the Inkscape Preferences (Edit > Preferences), in the Tools > Node section, you can set the default color used to draw the path outline when the Node tool is active. Typically this is set to 100% red on a standard installation. The swatches in this column can be used to set the path color on a per-object basis. Clicking on a swatch, then setting a color with a non-zero opacity, will change the color; set the opacity to zero, regardless of the color, to revert to the default set in the preferences. This may occasionally be useful if you're editing an object whose color is similar to the default, and you want to set it to something contrasting, but that's really the only sensible use for this option. You can see the effect on the top two stars which I have converted to paths, and changed the outline color on the second to blue:





Let's take a look at the context menu that appears when you right-click on any entry in the object tree. The first two entries let you rename the selected item (more easily done by double clicking on the label) and duplicate it (the same as pressing CTRL-D). The third item should probably be called "New Layer" for clarity, as it opens the new layer dialog regardless of what type of object was clicked on in the tree. This seems like a pointless addition to the context menu given that there's a dedicated button for this on the dialog.

The "Solo" option takes its name from music software, in which it is used to mute all the other tracks so that you can easily work on just one. In Inkscape, it hides all the top level layers and groups, except for the one that the selected item is in. It doesn't hide other elements that are in the same group, so it acts as a literal "solo" option only if each of your objects is in its own group or layer. It should probably also be renamed to "Hide Others" for consistency with the "Lock" options that follow in the next group. "Up", "Down", "Group" and "Ungroup" are fairly self-evident, but it is worth noting that "Ungroup" is available even if the selected object is not a group or layer. The same lack of context awareness exists for the "Set Clip" and "Set Mask" options, which are available even if you have only a

contents ^

single object selected – they require at least two objects in order to work. Thankfully in all these cases, the result of selecting an option that is not valid is for no change to take place, but it would still be better if invalid options were hidden or disabled in this menu.

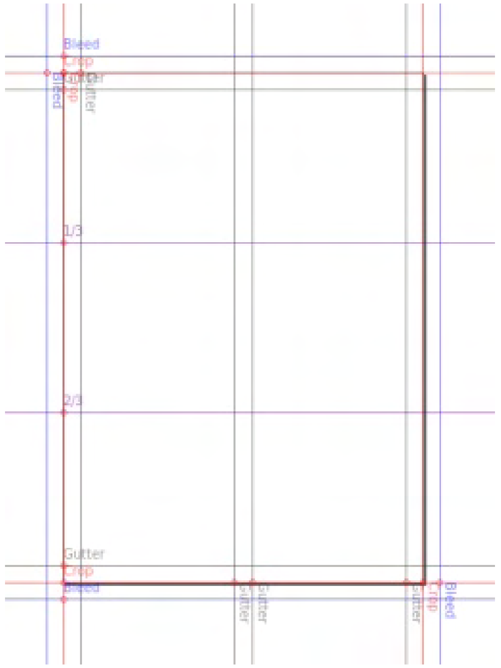
The new “Create Clip Group” option is also present on the context menu for an object but, despite experimenting with it myself and searching online, it appears that nobody really seems to know what it’s for, artistically speaking. Choosing this option will group any selected objects, then clip them with a clone of the group. Quite why you would want to do this – and especially why anyone wants to do it so frequently that it needs an entry on the context menu – is beyond me. I can think of a couple of esoteric situations in which this kind of structure is useful, but nothing that isn’t made clearer by creating it step-by-step. If anyone has any good ideas about this one, please let me know!

Moving on from the Objects dialog, 0.91 also introduced some small UI improvements that can have a big effect on how you use Inkscape. The first is the ability to enter simple calculations into spinboxes (the numeric fields with up/down arrows next to them). You can use the usual symbols for the main arithmetic operations of addition (+), subtraction (-), multiplication (*) and division (/), as well as brackets to group parts of the calculation. Any simple numbers will be used in the current units that are set for the field, but you can also append a unit name (e.g. “pt”, “mm” or “px”) to individual numbers to have them converted for you. For example the following, when applied to a field set as “mm”, will result in a value of 45.4mm: 10 + 1in + 1cm

Because the spinbox typically shows the current value of the parameter, you can often just tack bits of a calculation onto the end, then hit the RETURN key. Do you need a field to be 50% wider? Just add “*1.5” to the end of its value. Want to reduce it by 1cm, regardless of the current value? Just append “-1cm”.

One place where I find this feature invaluable is when setting guides. The Guideline dialog has had a “Relative change” checkbox for some time (see part 16 of this series), but now there’s really no need to ever use it, as you can just append a relative offset to the end of the existing value inside the X, Y or Angle spinbox. I’d really like to have a way to duplicate guides in a future release, as this would make it much easier to create a series of them by just setting the first in place then repeatedly duplicating and adding an offset.

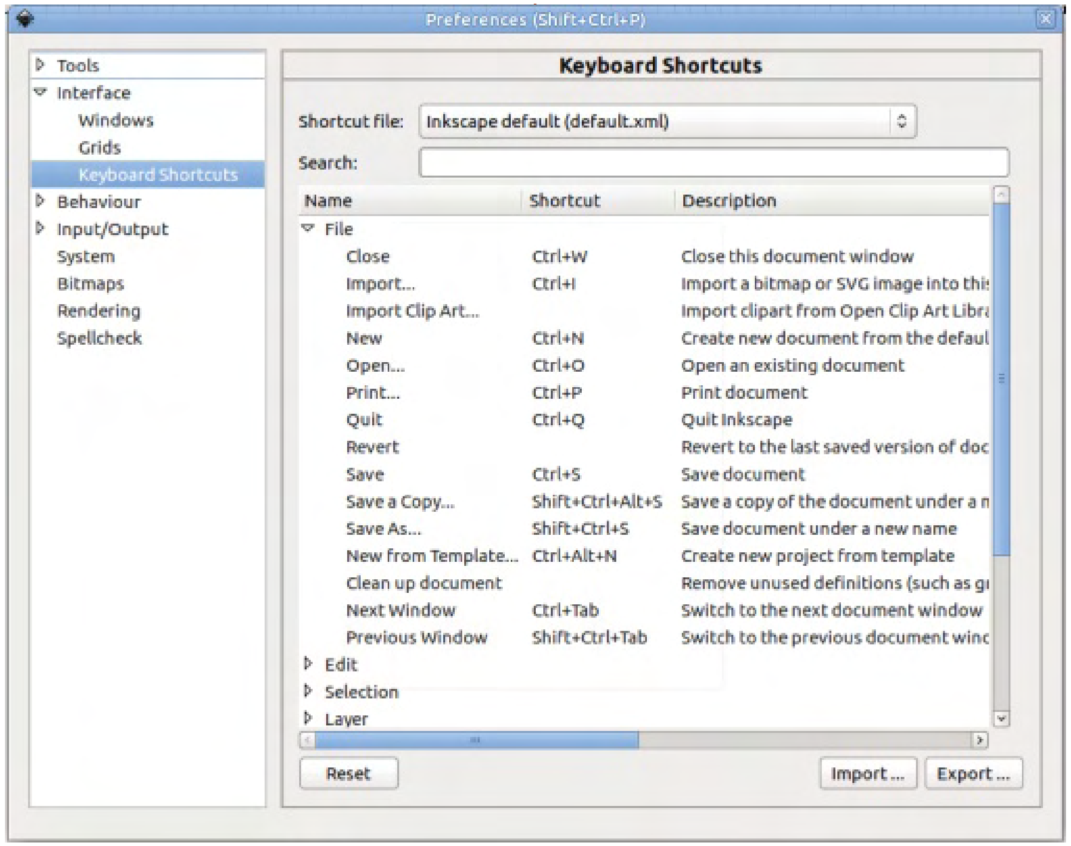
Although there’s no Duplicate button, the Guideline dialog has gained a few other features. It’s now possible to name your guides (the label appears in small text by the anchor point), give them



individual colors, and lock them against accidental movement. This can make guides much more useful when setting up a common template page that might be used as the basis of multiple similar documents in future.

Another UI addition that is particularly welcome is a new section in the Inkscape preferences for setting keyboard shortcuts. You can get to it by opening the preferences from Edit > Preferences (or SHIFT-CTRL-P by default), then drilling down to Interface > Keyboard Shortcuts. There’s a drop-down list at the top to let you select from a range of preset options, which could be especially useful if you’re coming to Inkscape from another product. As has become the norm with these kinds of dialogs in GTK programs, setting a new shortcut is done by highlighting the row you wish to modify, then clicking in the “Shortcut” column – either on the existing shortcut, or the blank space if there isn’t a shortcut defined for the operation you’ve selected. The field will populate with some text that says “New accelerator…” or something similar, which is your cue to press the keyboard shortcut you wish to assign. Note that there's no warning if you use a shortcut that's currently assigned to another operation – it will just be removed from the old command and assigned to the new one – so you may wish to choose your shortcuts carefully. If you want to remove a shortcut entirely, press the BACKSPACE key when prompted for the new accelerator.

If things go completely awry, you can change the shortcuts back to the defaults using the Reset button at the bottom of the dialog – where you'll also find options for importing and exporting shortcut configurations, so that you can back-up your highly customised settings somewhere safe.



Shortcut file: Inkscape default (default.xml)
Search:

Name	Shortcut	Description
File		
Close	Ctrl+W	Close this document window
Import...	Ctrl+I	Import a bitmap or SVG image into thi
Import Clip Art...		Import clipart from Open Clip Art Libr
New	Ctrl+N	Create new document from the defaul
Open...	Ctrl+O	Open an existing document
Print...	Ctrl+P	Print document
Quit	Ctrl+Q	Quit Inkscape
Revert		Revert to the last saved version of doc
Save	Ctrl+S	Save document
Save a Copy...	Shift+Ctrl+Alt+S	Save a copy of the document under a n
Save As...	Shift+Ctrl+S	Save document under a new name
New from Template...	Ctrl+Alt+N	Create new project from template
Clean up document		Remove unused definitions (such as gı
Next Window	Ctrl+Tab	Switch to the next document window
Previous Window	Shift+Ctrl+Tab	Switch to the previous document winc
Edit		
Selection		
Layer		

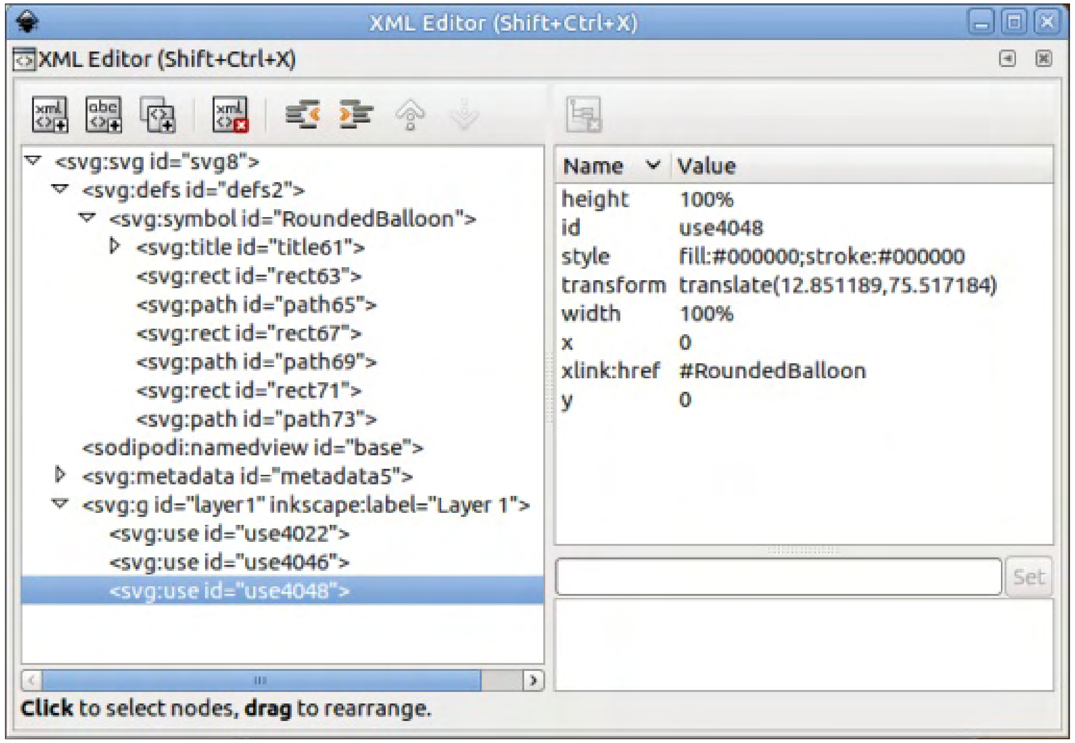
Mark uses Inkscape to create three webcomics, 'The Greys', 'Monsters, Inked' and 'Elvie', which can all be found at http://www.peppertop.com/

Written by Mark Crutch

Inkscape - Part 64

This month, we'll be continuing our look at some of the new features of Inkscape 0.9x by looking at the Symbols dialog. This is a frequently requested addition that allows you to create and maintain collections of related images that can then be inserted into the current document as necessary. Think about the boxes and shapes in a flowchart, the bubbles and word balloons in a cartoon, or the component symbols in an electronic circuit diagram, and you'll be on the right lines.

The SVG format has always allowed for symbols. They are stored in the <defs> section of the XML, where various definitions are held that could be re-used throughout the document. This is where you'll also find gradients and filters, for example. Just having a <symbol> element in this section won't actually render it on the page, however. For that, you have to include one or more <use> elements which refer back to the symbol. This screenshot of the XML editor shows the basic idea: the <defs> section near the top of the tree holds a single <symbol> element, which itself acts like a group, holding all the various parts of the symbol; meanwhile, at the bottom of the tree you can see three <use> elements, all of which refer to the same symbol via the "xlink:href" attribute you can see in the right-hand panel.



XML Editor (Shift+Ctrl+X)

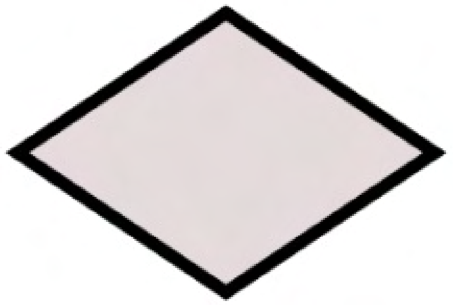
```
<svg:svg id="svg8">
  <svg:defs id="defs2">
    <svg:symbol id="RoundedBalloon">
      <svg:title id="title61">
      <svg:rect id="rect63">
      <svg:path id="path65">
      <svg:rect id="rect67">
      <svg:path id="path69">
      <svg:rect id="rect71">
      <svg:path id="path73">
  <sodipodi:namedview id="base">
  <svg:metadata id="metadata5">
  <svg:g id="layer1" inkscape:label="Layer 1">
    <svg:use id="use4022">
    <svg:use id="use4046">
    <svg:use id="use4048">
```

Name	Value
height	100%
id	use4048
style	fill:#000000;stroke:#000000
transform	translate(12.851189,75.517184)
width	100%
x	0
xlink:href	#RoundedBalloon
y	0

Click to select nodes, **drag** to rearrange.

There are two advantages to this approach. The first is file size. You can see that repeating all the content of the <symbol> element three times would take up more space than simply referring to it with <use> tags, and the more complex the symbol the greater the savings. The second is that changes to the original symbol will automatically propagate to any instances of it in the document. If that sounds a lot like clones then you'd be right – clones are also implemented as <use> elements, except that the original target is an object within the document body, not in the defs section. So, in some regards, symbols can be thought of as being clones whose source object isn't visible. If you've ever found yourself hiding the source of a clone behind another object, or outside the bounds of the drawing, you can probably see the advantages. If, however, you're scratching your head as to why you would ever need to hide the source of a clone, read on.

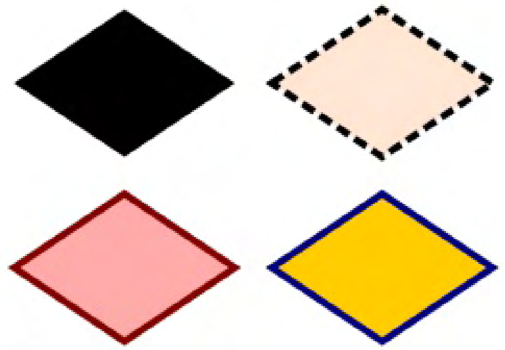
Consider drawing a flowchart, in which you want to use the "decision" block multiple times. The standard shape for a decision block is simply a diamond, so you might draw something like this:



You could use that one for the first block in your chart, then clone it for subsequent blocks, and

contents ^

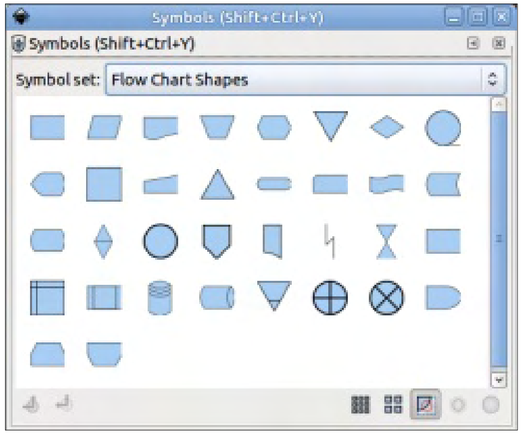
everything is fine… right up to the point where you decide to color-code areas of the chart. Now some of your decision blocks need a different colored background, or a different stroke. A simple clone will no longer do the job. The solution was described in detail in part 30 of this series: you have to unset the fill and stroke on the original, then you can set them individually on the clones.



Great! Now you can style each block exactly the way you want to. But you also have the unset original to contend with. The unset fill appears black, and the unset stroke isn't visible at all, so it's unlikely that you can use it as a decision block in your diagram as it stands. If you try to make it invisible, say by reducing its opacity to 0, you'll find all the clones disappear as well. So you have little choice but to hide it in some way – sneaking it behind another element in the drawing, or placing it outside the bounds of the drawing so that it doesn't appear in a web browser, or if you export the page as a PNG image. Arguably the most sensible approach is to put it onto its own layer, which you can then hide, but it makes it more awkward to use if you have to clone from it again.

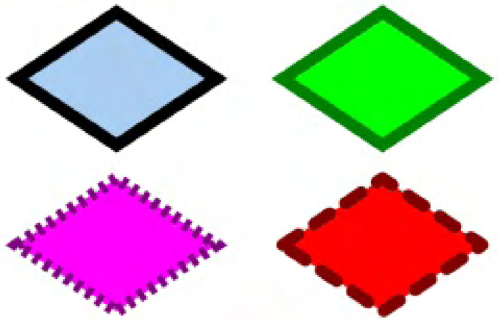
In theory, Symbols should avoid all these problems. But, in practice, the Inkscape implementation brings along enough quirks and issues that it's nowhere near as straightforward to use as it should be. Let's begin by opening the Symbols dialog via the Object > Symbols menu (CTRL-SHIFT-Y by default).

You'll probably see a rather empty dialog with a pop-up menu at the top and some buttons at the bottom. Using the pop-up, switch to the pre-defined Flow Chart symbols. You can use the buttons at the bottom-right of the dialog to change the size of the thumbnails, and whether they are individually scaled to fit, or all shown at the same relative size. Note that the latter can result in symbols that are cut off in the dialog, though they'll be fine when inserted into a document.



Find the Decision symbol – in my screenshot it's the penultimate shape on the top row. Note that each symbol also has a tooltip to help identify it. Select the Decision symbol and then either copy it to the clipboard (CTRL-C) and paste it into your document (CTRL-V), or simply drag it from the dialog onto your page.

With the symbol selected in the document you should see that the status bar specifically identifies the object as a "Symbol called Decision". Duplicating, or copy-pasting it gets you another symbol, as does dragging in an extra one from the dialog. With one of your symbols selected you can change the fill and stroke color, and even the stroke style.



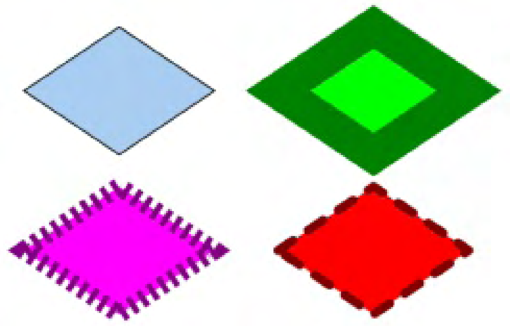
In other words, it behaves pretty much like a clone with unset fill and stroke, but without the original object being present on the page. But all is not perfect: try changing the stroke width and you'll find it stubbornly stuck at one size, regardless of how thick or thin the UI claims it to be.

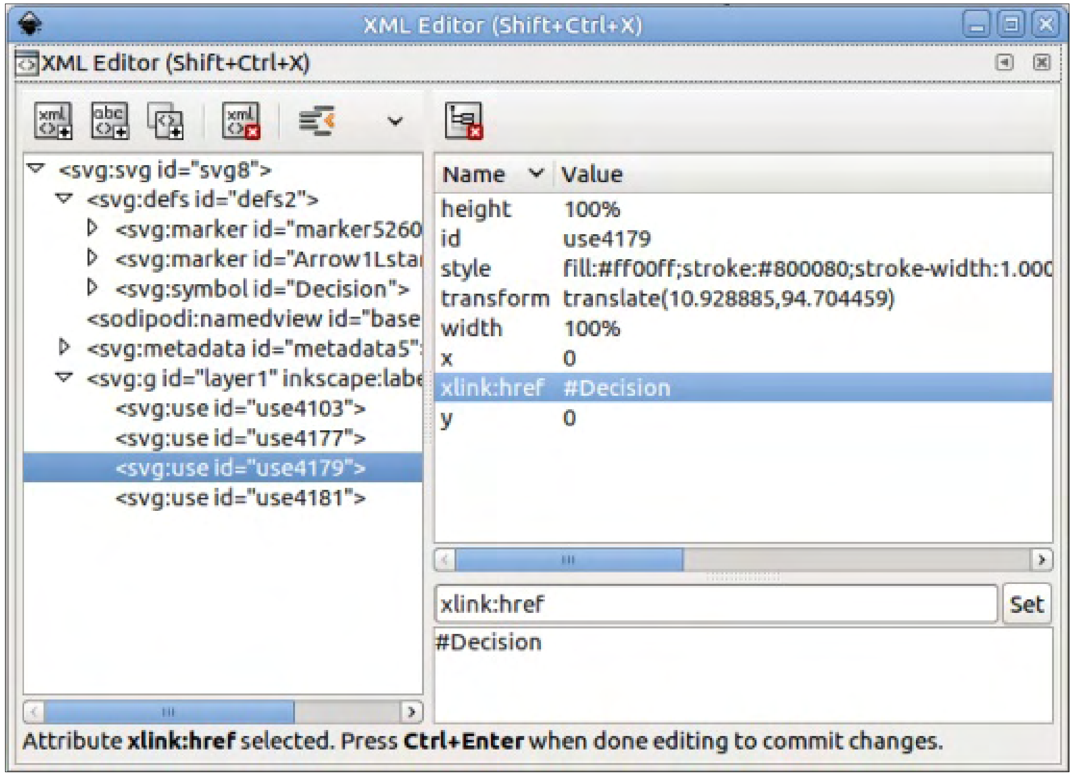
In fairness, this is an affliction that can also affect clones. Although most of the stroke properties are unset on the original object, it does have a stroke width defined, and no amount of fiddling with the cloned version will override it. The only solution is to remove the stroke width from the original object, which in this case means modifying the source symbol that's in the <defs> section of your document. There are two ways to do this: the easy way involving the XML editor, or the tricky way via the GUI. Yes, those sentences are the right way round – somehow the Inkscape developers have made GUI editing of symbols so unintuitive that it's often easier to wade into the XML!

After opening the XML editor (Edit > XML Editor, or CTRL-SHIFT-X) and selecting one of the symbols in my document you can see that the relevant line is highlighted in the left-hand panel of the dialog.

The attributes on the right include "xlink:href", which tells you the ID of the original symbol in the <defs> section, prefixed by a "hash" or "pound" character (#). Looking at the left-hand panel you should be able to see the symbol in question (in this document it's actually the only symbol, which makes finding it even easier). Expanding the symbol using the triangle on the left reveals the elements inside it. Because this is a really simple symbol, it's easy to guess that the element that needs to be edited is the path. The screenshot below shows how the XML editor looks with the symbol expanded and the path selected:

You can see that I've also selected the "style" attribute in the right-hand panel. Because this only contains the stroke-width property, I could remove the entire attribute by clicking on the "Delete Attribute" button in the toolbar at the top. In other cases, however, there may be additional style properties that you wish to keep. In that case you need to select the attribute, then edit the value in the bottom-right panel before clicking the "Set" button to apply your change. In this specific case the end result is exactly the same, as removing the entire text has the effect of also removing the attribute, rather than leaving it in place with an empty value. With





XML Editor (Shift+Ctrl+X)

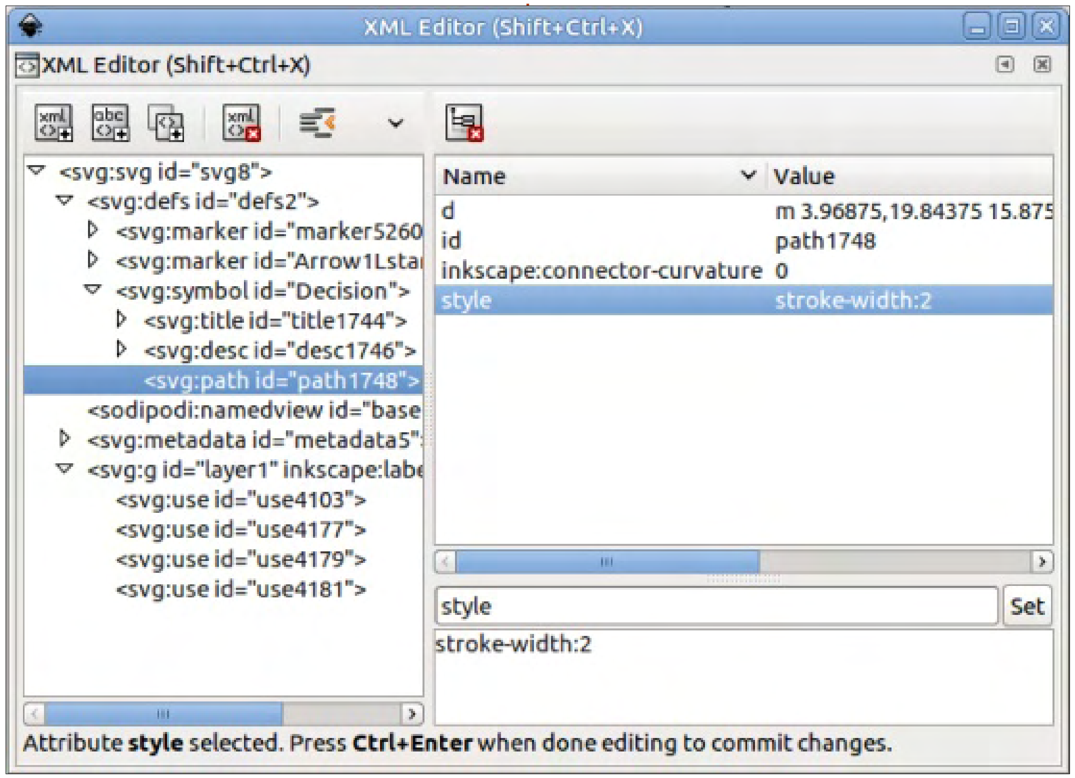
```
<svg:svg id="svg8">
  <svg:defs id="defs2">
    <svg:marker id="marker5260
    <svg:marker id="Arrow1Lstai
    <svg:symbol id="Decision">
  <sodipodi:namedview id="base
  <svg:metadata id="metadata5"
  <svg:g id="layer1" inkscape:labe
    <svg:use id="use4103">
    <svg:use id="use4177">
    <svg:use id="use4179">
    <svg:use id="use4181">
```

Name	Value
height	100%
id	use4179
style	fill:#ff00ff;stroke:#800080;stroke-width:1.000
transform	translate(10.928885,94.704459)
width	100%
x	0
xlink:href	#Decision
y	0

xlink:href Set

#Decision

Attribute **xlink:href** selected. Press **Ctrl+Enter** when done editing to commit changes.



XML Editor (Shift+Ctrl+X)

```
<svg:svg id="svg8">
  <svg:defs id="defs2">
    <svg:marker id="marker5260
    <svg:marker id="Arrow1Lstai
    <svg:symbol id="Decision">
      <svg:title id="title1744">
      <svg:desc id="desc1746">
      <svg:path id="path1748">
  <sodipodi:namedview id="base
  <svg:metadata id="metadata5"
  <svg:g id="layer1" inkscape:labe
    <svg:use id="use4103">
    <svg:use id="use4177">
    <svg:use id="use4179">
    <svg:use id="use4181">
```

Name	Value
d	m 3.96875,19.84375 15.875
id	path1748
inkscape:connector-curvature	0
style	stroke-width:2

style Set

stroke-width:2

Attribute **style** selected. Press **Ctrl+Enter** when done editing to commit changes.

that change made, you can now close the dialog, return to your symbols, and change their stroke widths to whatever value you choose.

Of course you'll sometimes want to make changes to a symbol that go beyond just removing or altering an attribute or two, and in that case you have little choice but to use the GUI method for editing it. In order to do that, however, you'll first have to convert it from a hidden object in the <defs> section to a real object in the page. During this feature's early development, this was done through a "Convert to Group" menu option, which seemed to me to be a simple and sensible approach. So, of course, it was removed from the release version.

Instead you have to start with a trip back to the Symbols dialog. Switch the pop-up menu to "Current Document" – the same view that was empty when you first opened the dialog earlier. Assuming you've added some symbols to your document, this view will now contain an image of each of them. You can think of this as being a sneaky view into the normally hidden world of the <defs>. To edit a symbol, you have to remove it from the <defs> section and insert it into the main document by selecting it in the dialog, then clicking the second button at the bottom left. Confusingly this has a tooltip that says it's going to remove the symbol from the current document – whereas the actual behaviour is to insert the symbol, as a group, into the document, removing it from the <defs> section and therefore from this view in the dialog.

The editable version that is put into your page has far more in common with a clone parent than you might expect. Because it has an unset fill and stroke, it will appear black, and if you select any of the symbols on the page that were created from it, you'll find that the status bar now declares them to be clones, rather than symbols. If you were to check the XML editor, you'd find that they haven't changed – they are still <use> elements with an xlink:href attribute that links them via an ID – it's just that the object with that ID is now in the page itself, not in the <defs> section. See, a symbol is quite literally just a clone parent that's stored in the <defs> rather than on the page. This duality also means that, if you do need to convert an individual symbol into a real object, Edit > Clone > Unlink Clone will work.

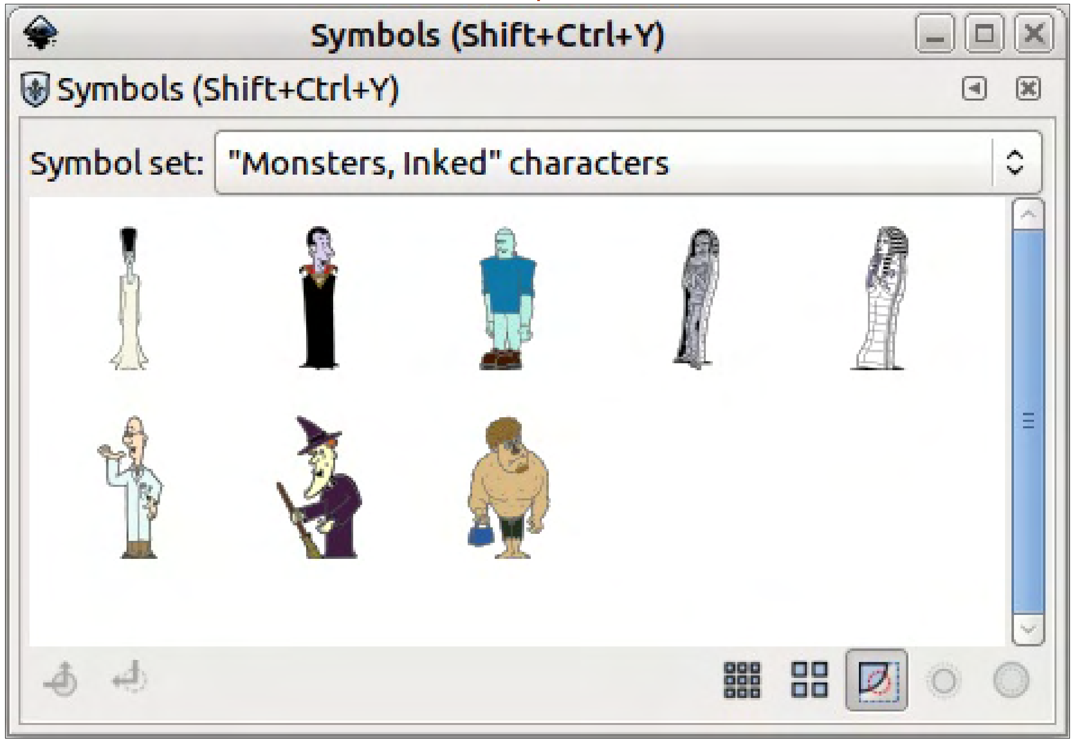
You can now enter the group and edit as you would normally. You can add extra elements into the group, change the shape of the path, or explicitly set (or unset) the fill and stroke to determine what parts of your symbol can be overridden when it's used in a document. If you examine the Object Properties for the enclosing group, you'll find that it's got a title and ID (as well as some other properties). Leave the ID alone, unless you want to break the link to any existing instances of the symbol in your document, but feel free to change the title to distinguish your newly edited symbol from the original. Finally, with it still selected in your document, use the bottom-left button in the Symbols dialog ("Add symbol from the current document") to insert it back into the <defs> section of the document. It also converts the copy you've been editing into a symbol – so you'll need to manually delete it if you don't need another one on your page. Once again, the early builds had a simple menu option ("Convert to symbol"), but with the release version we're stuck with confusingly tooltipped buttons in the dialog.

In practice you can turn any group into a symbol. Inkscape automatically creates an ID for your group, but it's probably best to open the object properties and set something more human-readable, as well as giving it a title. Then just select it and use the bottom-left button in the Symbols dialog to convert it. Don't forget to unset fills and strokes if you want to be able to alter them for each instance of the symbol in your document.

Once you have got to grips with creating one or two symbols, you might feel that you need an entire set, whether it be symbols for electronic components, crochet stitches or, in my case, a handy place to keep some cartoon characters. All you need to do is create all the related symbols in a single document, then give it a title in the File > Document Properties dialog (Metadata tab). Save the document to /usr/share/inkscape/symbols (for those you wish to share with other users on the machine), or ~/.config/inkscape/symbols (for use by a single user – though you'll probably have to create the "symbols" directory). The next time you start Inkscape, you'll find a new entry in the Symbols dialog's pop-up, showing the title you gave to your document. Select it to gain access to your own symbols.

As this image shows, symbols aren't restricted to simple shapes or single colors. Here I've got complex groups of paths with a variety of colors in each. It does also show, however, that there are still limitations: the sarcophagus on the right is supposed to be a shimmering gold color, but the repeating gradient of the original failed to make it into the symbol version. In general you may find similar issues with anything that links to a definition in another part of the document – such as gradients, patterns, filters or other symbols.

But the Symbols dialog is really intended for simpler shapes – for domain-specific tags, markers and building blocks that might be needed to annotate or create an informative image, rather than an artistic one. The standard symbols shipped with Inkscape are okay as a starting point, but somewhat limited in scope. So if you do create any symbol libraries that are generally useful, then please consider sharing them with the wider community – it's a great way that you can give back to the project without having to write any code.



Symbol set: "Monsters, Inked" characters

Mark uses Inkscape to create three webcomics, 'The Greys', 'Monsters, Inked' and 'Elvie', which can all be found at http://www.peppertop.com/