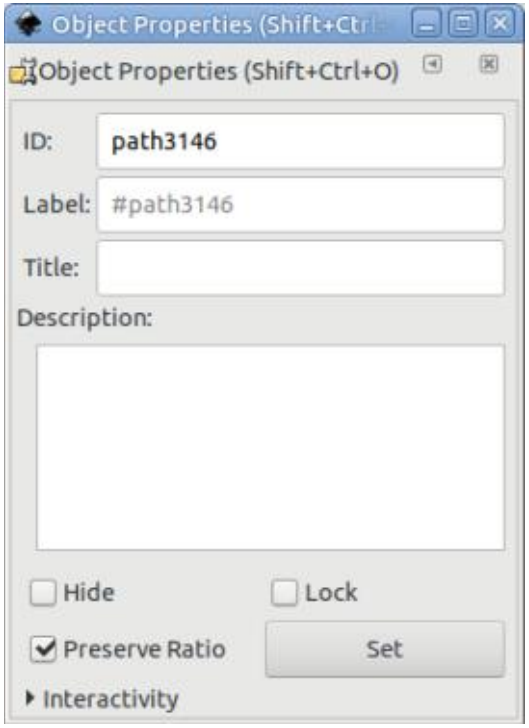
Inkscape - Part 101

A recurring problem that new users face when starting to use Inkscape is the myriad ways of accidentally making something invisible. A few articles ago, I talked about this with regard to Outline View, and the new Visible Hairlines, Split View and XRay modes in version 1.0. But there's one way of making objects disappear even from those tools, and which used to cause a lot of confusion for new users, as their objects vanished into the ether with no obvious way to retrieve them. Version 0.91 made recovery easier, and, with 1.0, we now have several new options on the context menu that make this feature something that might actually be more of a help than a hindrance.

There is an "Object Properties" dialog in Inkscape, which is available from the right-click context menu, making it easy for new users to stumble upon. It allows you to set the ID, Label, Title and Description of your object, but, in practice, most of those items only have any real value when the SVG is loaded into a web browser and manipulated with JavaScript. Small JavaScript snippets can be entered directly into this dialog, via the Interactivity section at the bottom (see part 82 of this series).

There are also a couple of checkboxes in this dialog that can easily tempt an unwary user. The "Lock" option stops you interacting with the object at all. Initially the object remains selected, and can be affected by keyboard shortcuts, but, as soon as the selection is removed – by clicking on the background or selecting another element – the object becomes completely inert. You can't select it, move it, resize it, or delete it. The "Hide" checkbox also makes your object inert – but, in addition, it makes it completely invisible, even to the prying eyes of Outline View and its friends.



Back in the days of v0.48 and earlier, these checkboxes were a real problem for new users. They would naively lock a bitmap they wished to manually trace over, only to find that the means to unlock it when they wanted to delete it was less than obvious. Similarly, a right-click on it would no longer present the Object Properties option – thereby concealing another unlock possibility. A hidden object fared even worse: the behaviour was the same, but you couldn't even see it to be sure you were right-clicking in the correct place!

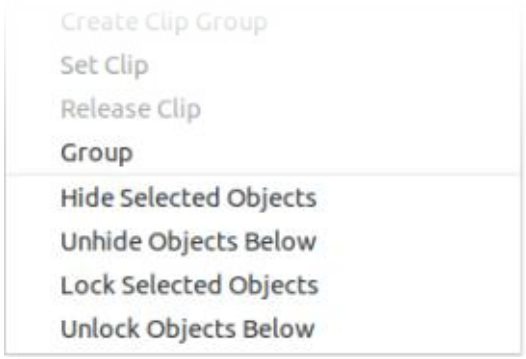
The correct way to unlock or unhide your object was actually to unlock or unhide all the objects in your drawing, via options in the Object menu. There was no UI (other than the XML editor) to let you unlock or unhide individual items, making these capabilities rather useless for managing the state of very specific parts of your drawing.

With 0.91 came a new Objects dialog (Object > Objects… described in part 63). This lists every element in the drawing – including the hidden ones – with handy toggle buttons to (un)lock and (un)hide them. It's a very familiar interface in other graphics software, and turns these properties into genuinely useful features. Personally, I think the checkboxes should have been removed from the Object Properties dialog to avoid further confusion, leaving them available only from the Objects dialog, but they still exist in the dialog to this day.

Inkscape v1.0 adds more UI niceties to work with these capabilities, by providing four new entries on the context menu:

```
Hide selected objects
Unhide objects below
```

```
Lock selected objects
Unlock objects below
```



The Hide and Lock entries are pretty self-explanatory. The one caveat to be aware of is that they hide or lock the objects that are selected, which may not include the one you've right-clicked on. Make sure to select all the target objects before right-clicking.

The Unhide and Unlock options refer to "below" in their titles. In this case they mean "below the mouse pointer". Right-click on a locked object and select the Unlock option to unlock it. It doesn't get much simpler than that, right? But there is a caveat: this operation will unlock any object below the mouse pointer, regardless of what layer it's on, even if the layer itself is hidden and locked! And, as the plural in the menu title suggests, if you have multiple locked items stacked on top of each other below the position where you right-click, they'll all be unlocked.

Unhiding follows the same rules, but it's obviously a little harder to find the correct spot to right-click on. As mentioned, none of the usual options for seeing invisible objects will work, so I hope you've kept track of where everything is in your drawing. To help with this treasure hunt just a little, the Unhide menu option will be enabled only if there is actually a hidden object below the mouse pointer, so you can be sure you're in the right place. If you can't find the object easily then a trip to the Objects dialog is probably a better use of your time than trying to play a hobbled version of Battleships against Inkscape.

The Unlock menu option also becomes enabled and disabled based on whether or not there's a locked item below the mouse pointer, but it gets confused by locked layers, becoming enabled when you right-click over any object in the locked layer, regardless of whether or not the object itself is locked.

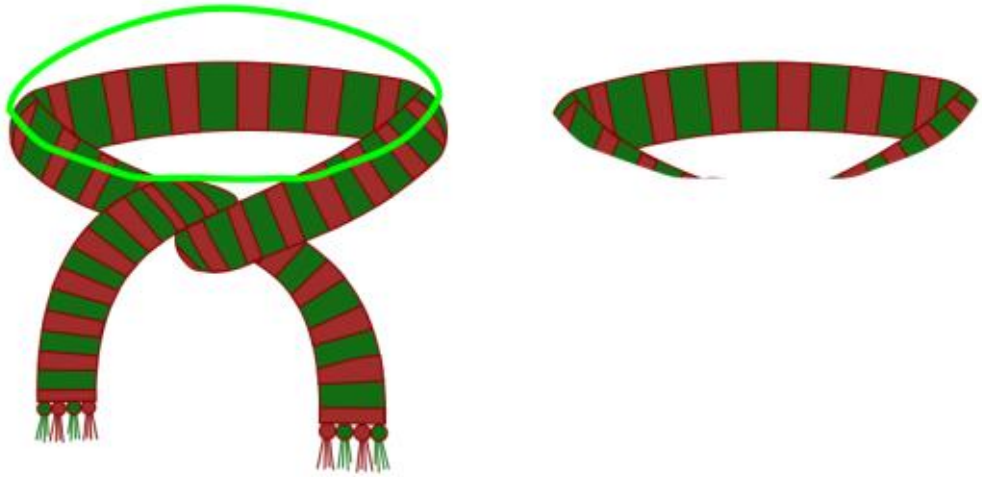
Despite these minor issues, adding the options to the context menu is a welcome change that surfaces these long-standing capabilities of the program in a way that makes them not only more discoverable, but also more usable. Small changes like this may not get the fanfare of the big new features, but, by making users' day-to-day workflows a little easier and more flexible, they're every bit as welcome.

Another small new feature is even more welcome, at least by me: inverse clipping. This is an omission from the SVG specs which has always frustrated me, not least because it solves common problems trivially, but is no harder for a browser or graphics program to implement than the standard clipping routines. For lack of an extra paragraph or two in the spec, users have been forced to spend time reinventing inverse clipping with their own complex paths time and time again.

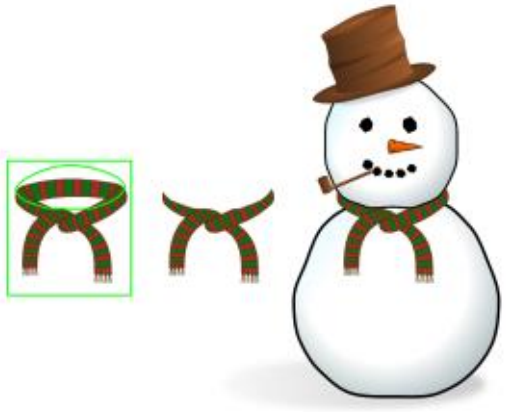
I covered clipping all the way back in part 13 of this series. If you're a very long-term reader, you may recall that this series began with a few articles to help you get to grips with the basics of Inkscape by drawing a snowman. By part 13 it was time to adorn him with a scarf.

I began by drawing a scarf in its entirety, including the part that would normally be hidden behind the snowman's head ("neck" seems a little too generous a term in this case). I first demonstrated how a simple clipping path (in green) wouldn't do the job – it would leave the back of the scarf visible, whilst hiding the part you actually wanted to see.

It's a classic approach: first show the audience what doesn't work,



then follow up with a demonstration of what does work. In this case, it's an "inverse clipping mask", created by using Path > Difference to cut the desired shape out of a larger enclosing rectangle. The resultant complex path is then used for clipping, giving the desired result.



It works, but it's not really beginner friendly. When a new user just wants to cut the center out of a circle to make a donut, being forced to confront Boolean operations and bounding boxes makes Inkscape seem rather unfriendly. If the SVG Working Group had only added an "inverse" parameter to clipping paths all those years ago, Inkscape probably would have included this feature from the earliest versions. As it is, we've had to wait until now – and it's still included only as a user-friendly addition by the developers, not because of any change to the SVG specs itself (this option was proposed for SVG2, but didn't make the cut, much to my ongoing frustration).

So how does this new feature work in Inkscape? As with many "extensions" to the basic SVG capabilities, it's implemented as a Live Path Effect (LPE). But, like several other LPEs, it's available directly from the normal Inkscape UI, so you don't need to deal directly with the LPE to benefit from it. I'll cover the new "Power Clip" LPE that is behind it in a future instalment but, for now, let's just see how we might use it to work with our snowman's scarf.

There's no real trick to it, actually. You just create a path (or an object that can be converted to a path) that encloses the part of the image you want to clip away. Then select both the clipping path and the object or group to be clipped and you're ready to proceed. Whereas the option for a normal clip can be found on the context menu, it takes a trip to the Object > Clip > Set Inverse (LPE) option to trigger the inverse mode. And that's it. No complex paths or bounding boxes. Just a simple interface for what should be a simple task.

Internally, of course, complex paths and bounding boxes still come into play. You can see this by switching to the Node tool (F2), where you'll see that Inkscape has automatically created a complex path consisting of your clipping path and another that hugs the bounding box of your selected object. As usual, you can edit the nodes and lines of these paths for an instant effect on the clipped object – ideal for fine-tuning the resultant shape to make sure your scarf provides a nice, snug fit.

There's also a new inverse mode for masks, though its behaviour is not so obvious. Masking was covered in part 14. It's basically similar to clipping, but uses the color of each pixel in the mask to determine the opacity of that part of the masked object. White parts remain visible, black or transparent parts are made transparent, and values in-between have their opacity set accordingly. Typically it's used where you want to fade the object, rather than produce the abrupt edge that clipping gives. Here's an example where I use a

blurred white object as a mask to remove the outside of a colored rectangle.

This particular result could obviously be produced simply by blurring a colored ellipse, but in a real example you might have a far more complex shape in your mask, or the object you're masking might be a group made up of lots of different elements.

Once again, the SVG spec offers no inverse version of a mask. But Inkscape provides such a feature in v1.0, so to cut a hole in the colored rectangle, surely just requires us to use the Object > Mask > Set Inverse (LPE) option, right? Of course not! That would be far too sensible. Instead the result is just a colored rectangle, as though no mask even existed.

What's happening here is a discussion for another article – probably once I get round to a further examination of the new LPEs in version 1.0. Suffice to say that we can get the effect we're after by drawing the mask in black rather than white, then using the Set Inverse option.

The irony here is that we've had to manually invert the color of the mask in order to use the automatic feature for inverting the mask! There is some logic to this, but it's tied up to the way the LPE operates, and the default settings that get applied when you select this option. For now, I'll be sticking to creating inverse masks by hand, the old fashioned way, but I can definitely see a lot of inverse clipping paths in my future.

There's one final caveat to using these new features: because they're implemented as Live Path Effects, they won't work directly on bitmap images. Note the word "directly" – these LPEs can also be used on groups, even if the group contains only a single bitmap image. So, if you really want to poke a hole through a treasured family photo, just put it into its own group, then get going with the inverse clip or mask. I usually advise putting a bitmap into a single-object group anyway, as it opens up some other creative possibilities I've covered in the past, so it's perhaps worth getting into the habit of always grouping a bitmap as soon as you place it in your drawing.

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 102

As you may have noticed in the News section of the previous edition of FCM, a new minor release of Inkscape is out. As is often the case with these things, the announcement came only a couple of days after the deadline for my previous article. As it turns out, version 1.0.1 doesn't fix any of the issues I've raised in recent months, or change any of the features that I've described. Phew! No corrections required.

So what is new in the latest release? It's mostly bug fixes and stability improvements, though there is a whole new "Selectors and CSS" dialog that is most likely to be of interest to web developers. I'll cover it in due course, but it's not the topic for this month's instalment. Instead, I'll cover some changes to the 'Path > Stroke to Path' function. To better understand this menu entry, however, I'm going to start with its sibling.

'Path > Object to Path' is a mainstay of any experienced Inkscape user. As the name suggests, it converts your object to a path, and is therefore commonly used when you want to break out of the design shackles imposed by an object's native type. Whilst a rectangle can have only its width, height and corner radius modified, converting it into a path lets you move individual nodes, add more, or delete others. Since paths are so flexible, it can open up a world of design possibilities – albeit at the expense of losing access to the specialized editing tools for the original object type.

There's nothing controversial about Object to Path. The end result maintains its fill and stroke properties, so there's no visual change as a result of the operation. All that's happened is that your native Inkscape object becomes a generic path of the same size, shape and appearance.

Despite the similar name, however, Path > Stroke to Path is an entirely different beast. At its core it converts any stroke that you may have on your object into a new filled path which matches the stroke's original outline. If it sounds confusing, perhaps some examples will help. Let's start with the simplest example possible: a straight line.

The top line is our original path, consisting of two nodes, no fill, and a thick, red stroke. As a rule it's easiest to understand what Inkscape is doing when applied to thick strokes, but everything I describe can also be done with thinner strokes if that suits your requirements.

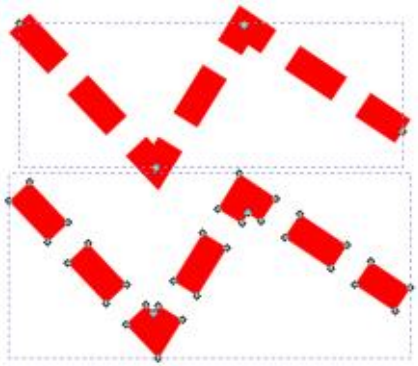
The bottom line is the result of the Stroke to Path operation. You can see that what we now have is a filled path consisting of four nodes arranged to match the original stroke's outline. The fact that it's a filled path is a really important point to grasp: the original shape had a stroke but no fill, whereas the new shape has a fill but no stroke. The fill color of the new shape is the same as that of the original's stroke color in order to produce a result that is visually identical.

Why would you want to perform such a conversion? Consider trying to make a line that looks a little more hand-drawn. SVG has no support for variable width strokes, but you can fake it by converting your stroke to a path and then tweaking the shape.

These days Inkscape can simulate variable width strokes using Live Path Effects, but that wasn't always the case. The LPE approach also supports varying the

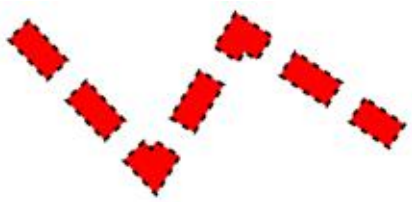
thickness only symmetrically, whereas this manual approach lets you achieve effects such as thickening the stroke on one side of the center line whilst thinning it on the other.

Let's look at a more complicated example. This time we'll jump straight to a multi-segment line with a dashed stroke applied. Once again the original line is at the top, and the bottom shows the effect of converting the stroke to a path.

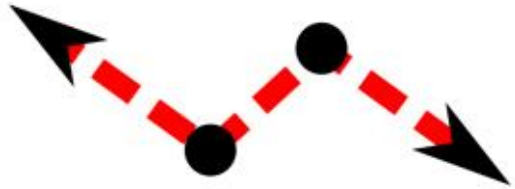


Our converted line is starting to look a bit more interesting. Instead of a simple filled path, we now have a complex path, consisting of a number of filled sub-paths which match the visual appearance of the original line.

At first, this can be a little tricky to get your head around. Because the resultant shape still looks the same, it's not obvious that it's actually now a filled path that has no stroke. Like any other path, you can actually add a stroke to it; here's the same result but with a thin, dashed, black stroke added to the converted line:



Hopefully it's now pretty clear what Stroke to Path does when presented with the simple case of an object with just a stroke applied. But what happens when your object is a bit more complex? Here's a line with a stroke applied, but also with markers at the start, end, and at each intermediate node.



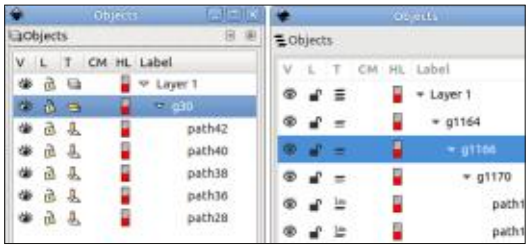
What do you think should happen when Stroke to Path is applied to this object? To my mind there are three possible alternatives:

- Remove the markers and convert the stroke as before.
- Convert the stroke, but also turn the markers into filled paths.
- Convert the stroke, but copy the markers to the new paths, so that each sub-path ends up with multiple markers.

Really old versions of Inkscape went with option 1. Stroke to Path converted the stroke to a path, and ignored everything else. But since version 0.44 (released in 2006), Inkscape uses the second option (no version uses the third option). This, however, is where not-so-subtle differences in behaviour for 1.0.x start to creep in.

Consider the shape above. Up to version 0.92 using Stroke to Path on this would have created a group containing five objects: the complex path generated by converting the stroke, and a separate path for each marker. In version 1.0.x the result is somewhat different. Now the output is a group containing two elements: the complex path, plus a nested group which contains four more deeply nested groups – one for each marker. Each of those marker groups contains two paths, one each for the fill and the stroke of the marker. Yes, I said the stroke of the marker. I know you probably weren't even aware of markers having a separate stroke, but apparently they do and they're now converted into a path of their own.

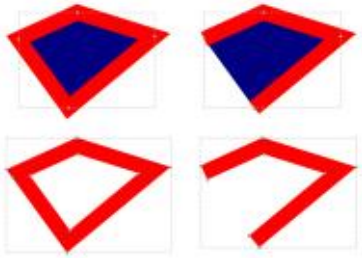
If this sounds a little confusing, perhaps an image of the structure will help. On the left we have the Objects dialog from 0.92, and on the right we have one from 1.0.1 (excuse the different themes – the 1.0.1 snap is still broken with the standard theme, so I'm using the symbolic theme):



As you can see, the structures are substantially different. If you do want to convert a 1.0 arrangement to the old structure, it's quite simple: you just need to use Path > Union on each pair of marker paths (this is easier if you select them in the Objects dialog rather than on-canvas), then select the group that contains the

markers (i.e. not the top-level group, but the one just inside that), then repeatedly ungroup until you're left with a structure that looks like the one in the 0.92 dialog. Yes, "it's quite simple" was sarcasm.

For most users, this may never present a problem. The sort of use-case that requires you to add markers to your path does not generally intersect much with the use-cases for converting the stroke to a path. So, let's look at a far more common scenario: a path with both a stroke and a fill, but no markers. Here's what happens in 0.92.x, again with the original shapes at the top, and the Stroke to Path conversions at the bottom:



The obvious take-away from this is that 0.92 removes the fill entirely before the stroke is converted to a path. I've shown both closed and open shapes to make it clear that the behaviour is the same in both cases. Structurally, you end up with a single filled path, just the same as you would if you'd performed the conversion on a shape with no fill or markers, the same as the first example I showed in this article.

Since version 1.0, however, performing a Stroke to Path operation on a shape with a fill results in a group that contains two filled paths: one is the usual path following the shape of the original stroke, whilst the other is a path representing the fill (i.e. it's just a copy of the original path, but without the stroke). The end result is visually identical to the original object.

As I'm sure you've guessed by now, performing a Stroke to Path on a shape with the holy trinity of stroke, fill and markers, results in a group containing:

- A path for the stroke
- A path for the fill
- A group containing an individual group for each marker, with each of those groups containing a path for the marker's stroke and a path for the marker's fill.

There are a few things to unpick from these changes. First of all, if you have an object with just a stroke then the behaviour is the same as it has always been: you end up with a single filled path, which will have sub-paths if the stroke was dashed.

If your path has a stroke and a fill, however, you'll now end up with a group, rather than the fill being thrown away. This is particularly important to note if following an older tutorial. Many of them either rely on the fill being removed, or instruct you to duplicate the object before applying Stroke to Path so that you don't lose the fill. To get the same result as earlier versions, you can do one of two things:

- Remove the fill before using Stroke to Path.
- After using Stroke to Path, ungroup the result and delete the new path containing the fill (or just keep it, if you need it for subsequent steps).

Despite the confusion this change has caused with some new users, I generally think it's an improvement. Consider the case of drawing a simple cartoon character, with thick black outlines (strokes), and colored clothes and skin (fill). If you want to add some variety to the stroke thickness, the new behaviour makes it easier to do so without having to duplicate every object, then remove its stroke, just to keep a copy of the filled shape.

Markers are another matter, however. While dealing with an unwanted fill just requires a couple of extra steps, getting the 0.92 structure when markers are in play requires several steps – multiplied by the number of markers – with no easy shortcuts. Although the new functionality may technically be more flexible, in practice there are few use-cases for separating the fill and stroke on a marker. It would have been nice to either have the old method available as a preference, or to include a function or extension that would make it easy to union deeply nested paths and recursively pop them out of their groups.

The take-away from all this is that Object to Path still behaves the same way it always did, Stroke to Path may need an extra step or two if you have a fill, but if you have markers as well then good luck to you!

Next time, I'll start to look at the changes to core drawing tools made with version 1.0.

Inkscape - Part 103

Written by Mark Crutch

We start this month with a small correction. Last time, I said that the recent minor revision of Inkscape, version 1.0.1, hadn't fixed any of the issues that I'd mentioned in recent articles. In fact it has addressed one problem that I described in part 100 of this series. In version 1.0, it was no longer possible to reverse a sub-path by selecting a single node and using Path > Reverse. Instead, you had to break the path apart, reverse the path in question, then combine all the paths again. Inkscape 1.0.1 reinstates the previous behaviour. In practice, this is a rarely used feature that really comes into its own only when dealing with fill-rules and self-intersecting paths, as I detailed in part 95, but it's nice to see it fixed nevertheless.

The remaining issues described in part 100 – problems with converting text to a path, and other issues with linked offsets – still remain in 1.0.1. As the workarounds to these can be a little tricky to follow, I've made a YouTube video that covers this topic in a more visual way. This is my first Inkscape tutorial video, so please leave a comment if you want to see more.
http://www.youtube.com/watch?v=lx5nRCu7AKk

Now, back to the usual programming, with a look at some of the changes and additions to Inkscape's drawing tools that were introduced in version 1.0:

Reordered Toolbox

Although no new tools have been added in 1.0, the existing tools have been reordered within the toolbox on the left of the screen, in order to group them more logically. Thin dividers are used between the groups, which has the effect of providing a little structure to what was previously just an undifferentiated list. In theory this should make it easier and faster to find the icon you're looking for, and that has certainly been my experience. The tools are grouped as follows:

Edit tools – for editing existing objects: Select tool, Node tool

Shape tools – for creating and editing geometric shapes: Rectangle & Squares tool, Circles & Ellipses tool, Stars & Polygons tool, 3D Box tool, Spiral tool

Primitive tools – for creating basic objects: Bézier Curve tool, Freehand (Pencil) tool, Calligraphy (Pen) tool, Text tool

Color tools – for working with colors and gradients: Gradient tool, Mesh Gradient tool, Color Picker (Eye Dropper) tool, Fill tool (Bucket Fill)

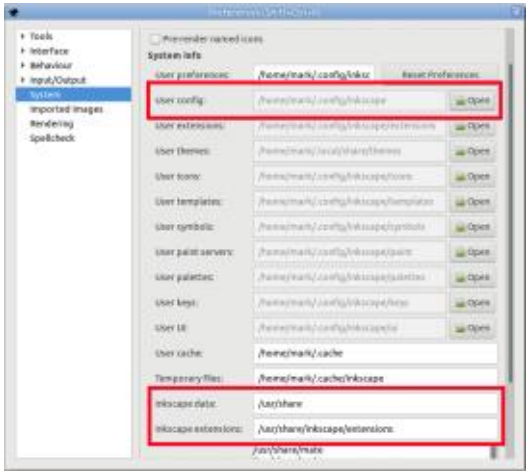
Other tools – miscellaneous tools not included in the other sections: Tweak tool, Spray tool, Eraser tool, Connectors tool

Canvas tools – for manipulating the canvas view: Zoom tool, Measurement tool

These are my descriptions, which vary from the terms used in the official release notes, but which I think are more descriptive and consistent. It's easy to argue about the placement of some items, but generally I think the groupings mostly make sense, and do move the less commonly used icons towards the bottom of the bar. If you don't like the order, however, Inkscape 1.0 introduces a mechanism for changing it… albeit one that isn't exactly user friendly.

Reordering Tools

The order of the tools is now defined by an XML file that is read when Inkscape starts up. You can override this file by creating an edited copy in your user config directory. First you'll need to find the paths for the shared folder (where the original file lives) and your user config directory. You can find the latter, and hints to the location of the former, by opening the Inkscape preferences dialog (Edit > Preferences) and selecting the System panel.



Open a file manager, then navigate to the path shown in the "Inkscape data" field. That may well be a directory that's shared with multiple programs, as is the case with the /usr/share value in my screenshot. Use the search facility in your file manager to find a file named "toolbar-tool.ui", starting from this root directory. In practice that will probably search through far more files than you need to, so you can make the search more efficient by being slightly more targeted in your choice of starting directory. That's why I've also highlighted the "Inkscape extensions" field in my screenshot: that's not the directory you want, as it's a little too specific, but it should give you a good hint as to where the common Inkscape files are stored. In my example, /usr/share/inkscape is a better starting point.

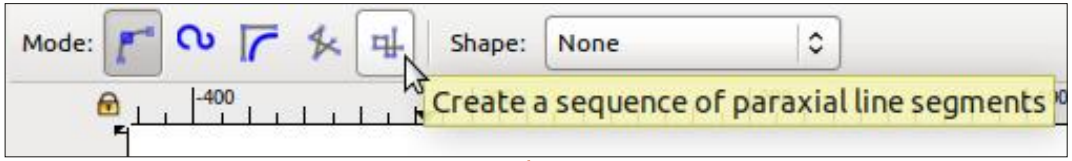
On my machine I found the "toolbar-tool.ui" file in /usr/share/inkscape/ui.

Once you've found the file, open a second file manager with the path from the "User config" field as its location. This should be pretty easy, as the Inkscape developers have provided a handy "Open" button right next to the field.

Create a "ui" folder in your user config location, if one doesn't already exist. Copy the "toolbar-tool.ui" file into it, making sure that you definitely copy rather than move the file.

Quit Inkscape if it's still running, and open the newly copied file using a text editor. It's a fairly flat XML file which should be pretty self-explanatory. To move tools between groups simply re-order the lines in the file; to hide a tool completely, wrap it in "<!--" and "-->" delimiters, similar to those used for comments in the file. Save your changes and launch Inkscape 1.0 to confirm that your new tool arrangement is working. If you have any problems and get completely stuck, you can quit Inkscape and delete the new file in order to return to the default arrangement.

This new capability may be particularly useful when using Inkscape on a machine with a smaller screen. If there's insufficient height to draw all the tools in the toolbox, Inkscape moves any excess tools into a pop-up menu at the bottom of the box. By moving the tools around in this configuration file, you can ensure that less useful ones end up in the pop-up while those you use commonly are always just one click away.



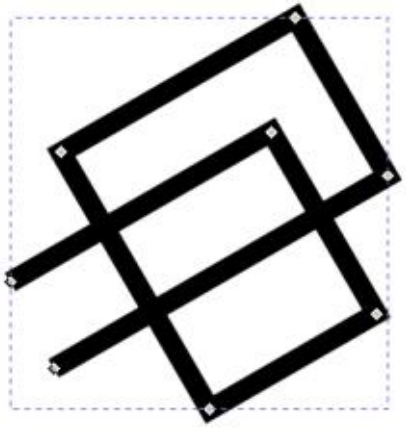
Here's a quick bonus tip: there's also a "toolbar-commands.ui" file in the same directory which can be used to re-order and hide entries in the main Inkscape toolbar.

Bézier Tool

The Bézier tool's control bar (above) has a button with the confusing tooltip of "Create a sequence of paraxial line segments".

In layman's terms, this was the "draw only horizontal and vertical lines" mode. With this mode enabled, Inkscape would allow you to draw only an alternating sequence of horizontal and vertical lines. You could switch modes mid-path – if you wanted to switch to the "straight lines" mode to add a single off-axis segment, for example – but any parts drawn under the influence of this control could be only horizontal or vertical.

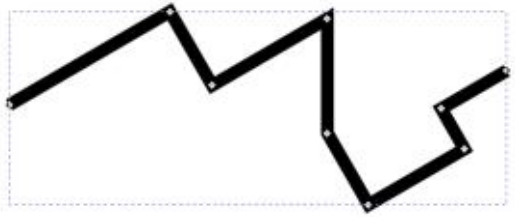
With 1.0, the layman's term for this button would now be the "draw perpendicular lines" mode. Now the segments are constrained by the first segment you draw: the second segment will be perpendicular to it (i.e. at a 90° angle to the first segment), the third segment will be perpendicular to the second (i.e. at the same angle as the first), and so on. The first path segment can be drawn at any angle, essentially turning this into a version of the previous paraxial mode, but with built-in rotation.



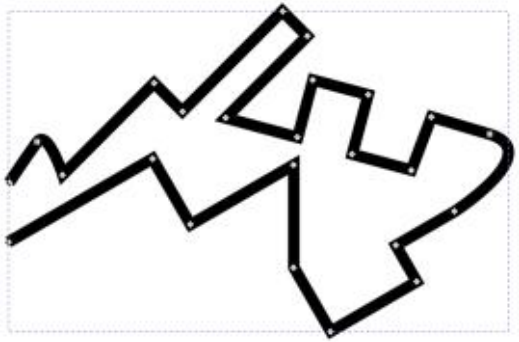
If you still require precise horizontal and vertical segments, make sure to hold the Ctrl key when drawing the first segment. That will constrain the initial line to one of a fixed series of angles, defined in the Behaviour > Steps section of Inkscape's preferences, and defaulting to every 15°. The previous image was drawn using this method to fix the initial segment at a 30° angle.

Once again, you can switch to other modes mid-way through drawing a sequence of path segments, but the behaviour might not quite be what you expect. Consider drawing a series of paraxial paths at, say, a 30° initial angle. If you require a series of horizontal and vertical paths to continue the sequence, you might think that you can switch to the "straight line" mode, hold Ctrl to create your initial horizontal or vertical line, then switch back to paraxial mode to continue with further horizontal and vertical line segments.

In practice, the paraxial mode remembers the initial path angle you used, not the most recent one. So, after switching back to paraxial mode, you would end up with further lines at 30° and 120°, not at the 0° and 90° you wanted. You can see this effect in the following example where the fourth segment was drawn vertically, but the fifth and subsequent segments are still constrained by the angle set with the very first segment.



The workaround is to end your path, then start a new one. If the previous path is still selected, the Bézier tool lets you continue by starting your next line segment at the end node of the existing path. Each time you do this with paraxial mode enabled, the first segment you draw will be the reference segment for the remainder of that path. You can repeat this as many times as required to produce complex lines with differently oriented paraxial sections, interspersed with curves or lines at arbitrary angles.



Calligraphy Tool

There are two things that every sentence in this article has in common. Two basic rules of written English. They all start with a capital letter, and they all end with a punctuation character – usually a period ("full-stop" in British English), but often a question mark, exclamation mark, or colon. What all these characters have in common is that they require the ability to draw a dot. Given this fundamental requirement of written communication, it's surprising to note that Inkscape's calligraphy tool previously had no practical way to make a single dot.

You could kind of fake it by drawing a small circular shape with the tool, but make the circle too small and Inkscape would ignore it, while too big resulted in a large misshapen splodge. With 1.0, the developers have added the ability to directly create a dot – but in a rather odd way that, in my opinion, doesn't really address the underlying requirement.

To draw a dot, you just have to click with the primary mouse button, as opposed to click-drag when drawing a calligraphic stroke. As that mouse button usually also maps to pressing the tip of the stylus on a graphics tablet, anyone trying to write some calligraphy using such a setup just needs to press the stylus down and up to draw a dot. It's as simple and intuitive as can be.

You can also hold down the Shift key while performing the same operation to create a larger dot. Larger, in this case, means exactly twice the diameter of a small dot.

The fact that I can easily tell you it's exactly – not roughly – twice the diameter reveals the first problem with this new feature. Whereas calligraphic strokes are actually created as filled paths, these dots are created as circles. If you want to edit a stroke, double-clicking on it allows you to drag the individual nodes around. To do the same with a calligraphic dot, however, first requires a trip to the

Path > Object to Path function. Without that step you're limited to the changes that can be performed with the Circle tool.

By rendering a circle, your dot is a pure shape, with no lumps, bumps or character to it. It doesn't matter what your Calligraphy Tool settings are: you could be using the Wiggle or Splotchy preset, or have some custom values to create a frantically random stroke, but your dots will always be circular. Using the Dip Pen preset for a classic calligraphy style, with angled lines that would suit a diamond-shaped dot? Nope, you still get a circle.

A far larger – or rather, smaller – problem is the size of the dots. They're tiny! Even the large ones. Here are four examples: each is drawn using the Marker preset, with the width set to 25, 50, 75 and 100. In each case, I've drawn a single calligraphic stroke, followed by a standard dot, then a large dot.

As you might imagine, these don't make for great titles with your punctuation marks, should you be using the Calligraphy Tool for, you know, actual calligraphy. In an example about as far from classic calligraphy as you can get, here's some mouse-drawn text to demonstrate just how useless these dots would be for terminating a sentence.

It seems obvious to me that the dot sizes should be far closer to the width of the line. Perhaps 75% of the width for a small dot, and 150% for a large one. As it stands, this feature is mostly useless. You could scale the dots up after drawing, but they'll still be pure circles with none of the character of your selected pen. And quite honestly, if you have to manually scale the circles anyway, then you may as well just draw them using the circle tool after completing the rest of your lines.

Circle Tool

On the subject of the circle tool, there's been one small but welcome addition. As you probably know, you can move the round handles on a circle or ellipse to open the shape out, forming arcs (when the mouse is released inside the shape) and segments (when the mouse is released outside the shape). You can toggle between these, and a completely closed shape, from buttons in the tool control bar.

The new addition is the ability to create chords – arcs where the two ends are joined with a straight line. There doesn't appear to be an on-canvas method for doing this, but if you create an arc or segment, you can switch it to a chord using a new button on the tool control bar. It's a small addition, but good to have nevertheless.

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 104

This month, we'll be continuing to look at the changes in Inkscape's drawing tools that arrived with version 1.0.

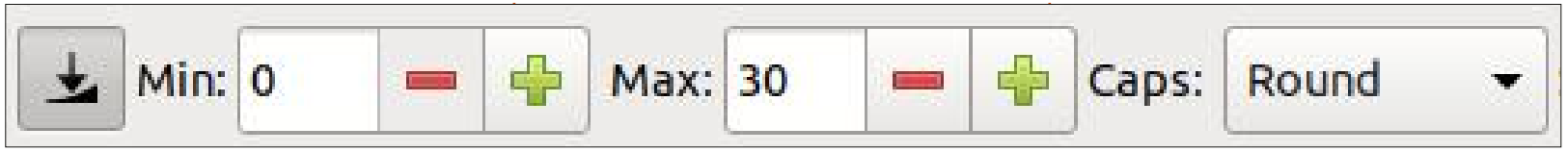
Pencil Tool

Last time, I covered changes to the Bézier and Calligraphy tools, so let's start this article with the third of the line-drawing tools, the Pencil or Freehand tool. The release notes mention only one change to this tool (albeit one that comes with controls for several parameters), so I'll begin with that. In Inkscape 1.0, the tool control bar has gained an additional button in the "Mode" section, outlined in red in this screenshot:

Whereas the first three entries in this section act as radio buttons, the new addition acts as a toggle. This seems a little odd to me, because toggling it 'on' immediately disables the other three buttons. It's still only possible to have one option exclusively selected (the hallmark of radio buttons), so I don't really understand why this wasn't just included as an extra radio button, rather than implemented as a toggle. What it does mean, though, is that switching to a different mode requires the additional step of turning off this toggle to re-enable the older buttons.

Minor UI gripes aside, what does this button actually do? It enables pressure sensitivity for the tool, which is of no practical use to mouse users, but may be beneficial to tablet users. Time to dig out my 'cheap-but-does-the-job' Monoprice branded tablet; it's no Wacom, but it works out-of-the-box on my Ubuntu Mate system and supports pressure sensitivity.

With the toggle enabled, the tool control bar changes to look like the image below, providing additional controls labelled "Min", "Max" and "Caps":



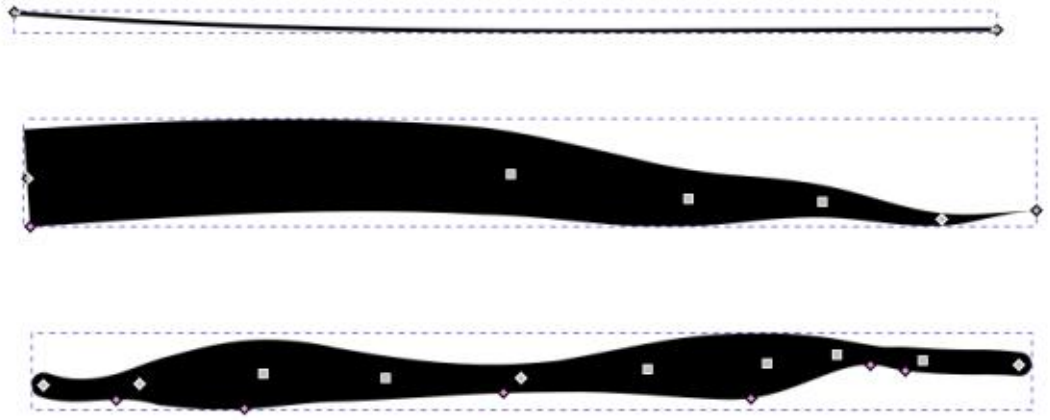
Before describing what those controls do, a brief recap of the way that the Pencil tool works is probably useful. Historically, it was just a tool for freehand drawing that would create simple paths. Starting and ending at the same point would create a closed path, whereas a different ending point would result in an open path. The thickness and style of the resultant path was trivially based on the fill and stroke options, as you might expect.

More recent releases added a "Shape" pop-up menu. With that set to "None", the behaviour remained as described above, but by selecting a different option in that menu, the behaviour of the Pencil tool changes dramatically. The shape you draw is no longer the stroke of a path, but rather it is a filled path in its own right. In 1.0, the shape is the result of applying a Live Path Effect onto the skeleton path you've drawn. The triangle-in and triangle-out shapes are created using the Power Stroke LPE, whereas the ellipse and clipboard-based shapes are produced using the Pattern Along Path LPE. As you can see, this tool relies heavily on LPEs for its more advanced features.

This brings us back to the new pressure-sensitive mode. This is also implemented using the Power Stroke LPE, but whereas triangle-in and -out just have a single control point for setting the thickness of the triangle's base, in pressure-sensitive mode, Inkscape creates multiple control points along the length of the drawn line – whenever there's a significant change in pressure. The result is a line that moves from thin to thick and back, based on the pressure

you apply, but with the ability to tweak the thickness of each part of the stroke by switching to the Node tool and manipulating the purple control points.

You can see the effect in the image below. The top line was created with the shape control set to "None": it's a simple path with nodes at the ends, but no way to adjust the stroke thickness along its length. The second line used "Triangle-out", and you can see that there's a single purple handle at the left that is used to adjust the stroke thickness set by the Power Stroke LPE. The third line was the result of some random pressure adjustments using a graphics tablet: this one has even more handles for adjusting the skeleton path, plus additional purple LPE handles at each pressure change.



This makes it easy to compensate for poor pressure control by adjusting the line thickness after drawing.

Knowing that this mode enables the Power Stroke LPE makes it easier to understand what the various controls on the tool control bar do. Min and Max set the minimum and maximum values for the purple control points. The Caps control sets the shape of the end caps, and mirrors the corresponding control within the LPE itself.

For my tablet, setting a Min of 0 and a Max of 20, with round end caps and a small amount of smoothing (around 10 on this scale), gives a nice "marker-pen" effect that responds quite nicely to the pressure I tend to apply to the stylus. Potentially, this can provide a much more naturalistic feel to cartoons and sketches – though it still requires a more artistic hand than my own to produce something impressive.

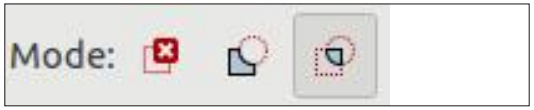
Eraser Tool

I make no secret of the fact that the Eraser is one of my least favourite tools in Inkscape. Not because it has any inherent problems or limitations, but because it's deceptively familiar for users coming from bitmap editors. Too many words have been expended on forum threads in which new Inkscape users bring their bitmap preconceptions to the vector world, and the presence of an eraser that goes some way towards mimicking its raster equivalent makes it even harder to explain why Boolean operations or clipping paths are a usually a better solution to their task.

Nevertheless it does exist, and it does have its uses, so it's nice to see it gaining some additional features in 1.0. The bulk of the additions are to make it operate in a similar way to the Calligraphy Tool: there's a toggle for pressure-sensitivity, which causes the "Width" field to become a maximum value, but with the actual value used depending on the pressure applied to the stylus. It also gains controls for Thinning, Caps, and Tremor, mirroring those in the Calligraphy tool.

I won't spend any more time discussing these: you can read my description of the Calligraphy tool in part 78 if you wish. I still believe a more useful approach would actually be to draw the erasing line you want using the Calligraphy tool (or some other tool, if you prefer), then either perform a Boolean operation or use it as the basis of a clip or mask.

In that vein, however, the second addition to the Eraser is more welcome: a clipping mode. Previously this tool could either erase objects entirely, or cut away parts of the shapes by effectively performing an immediate Boolean operation with the drawn line. The clipping option adds a third button to the Mode section of the tool control bar:



With this enabled, the "erasing" is actually performed by creating a clipping path that is immediately applied to the object. Where more than one object is affected, each one gets its own clipping path, independent of the others, even if the original objects overlapped.

A very nice touch is that only a single clipping path is created for a given object, even if you erase using several separate strokes. This makes it easier to remove large parts of an object with a thick eraser, then reduce the tool's width for subsequent passes to refine the shape being removed, without ending up with multiple clip paths to manage.

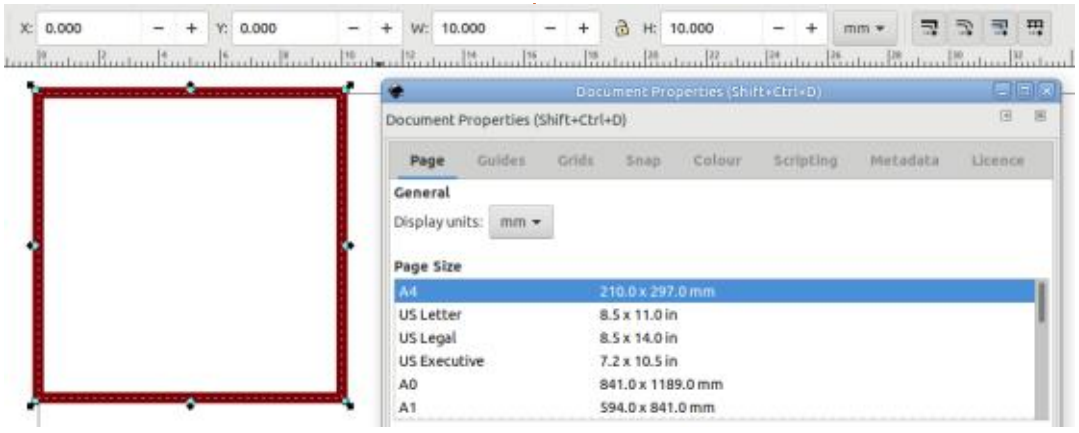
It's important to note, however, that the results produced by clipping will not always be the same as those created using Boolean erasing, particularly if the target object has a visible stroke applied. Consider the image below, with a deliberately thick stroke to make the point. On the left is the original shape, whilst the middle and right-hand images show the results of the Boolean eraser and the clipping eraser, respectively. Notice that the Boolean eraser results in separate path objects, each with a complete stroke around them. The clipped shape, on the other hand, is still a single object, so the cut faces are not "closed" by the path.

The clipping mode is a great addition to the Eraser tool, creating a non-destructive edit that can later be refined, or reverted entirely, with ease. When creating my comic strips, I often have to apply clipping paths around very precise shapes. Usually I block them out with straight-line paths, then use the Node tool to fine-tune the corners and curves. But this new feature promises to get me 90% of the way to a finished clipping path in a fraction of the time. In fact, this addition alone may have just promoted the Eraser tool from being one of my least used tools, to being a clear favourite.

Measurement Tool

The Measurement tool is another that I rarely use, but which has also seen some small but significant additions in Inkscape v1. This month, I'll talk about the first of these: an extended "tooltip" that appears when hovering over a path (or an object that can be trivially converted to a path, such as a rectangle, ellipse or star). The tooltip shows the length of the path, as well as the X and Y coordinates, width and height of the object's bounding box. In theory, this could provide some useful at-a-glance information about a path… if it wasn't for the fact that the data displayed aren't always correct. Let me prove this assertion with an example. Here's a square, drawn so that its top-left corner is at (0, 0), with sides of 10mm. The document properties are set with mm as the display units, and I'm using the geometric bounding box so that the stroke width doesn't factor into the dimensions.

Switching to the measurement tool and hovering over the square produces this tooltip:

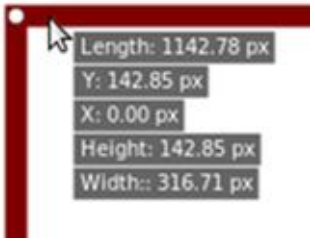


The first problem here is that the values displayed are in pixels, even though I specified mm as my display units. That's easily explained: the Measurement tool has its own toolbar, with a pop-up menu to select the units it displays. Changing that pop-up to "mm" results in this tooltip instead:

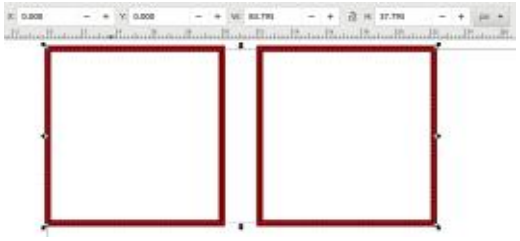
A quick comparison with the previous screenshot shows that although the units are now correct, the actual numbers are still the pixel values! According to this tool, my 10mm square now has sides that are nearly four times as long.

And what of that Y value? I drew the rectangle from top-left to bottom-right, the selector tool shows the Y value as zero, and even the XML editor agrees. Yet, for some reason, the measurement tool wants to take its Y value from the bottom of the shape, not the top.

Now let's give it a more complex path to work on, by duplicating the square, moving the copy, then using Path > Union to convert the two squares into a single path comprising two sub-paths. Since each sub-path's perimeter is 151.18px, surely the Length in the tooltip for both paths must simply be double that value, right? Wrong.



The Length displayed for two identical paths combined is over 7.5 times the value shown for a single path! The Height and Width values also have me scratching my head: they might make sense if the squares were arranged along a diagonal, but they're side-by-side. Here's what they look like when using the Select tool, together with the tool control bar's interpretation of these values (in px):



Having the Select tool declaring a width of 83.795px and height of 37.795, while the Measurement tool claims values of 316.71px and 142.85px, leaves me thoroughly confused as to how the values in the tooltip are actually calculated. This situation leaves me uneasy about the measurement tool in general; I wonder what it has to say when used in the traditional click-and-drag mode?

For clarity, I've copied and enlarged the top and right values, and put them inside the boxes. The width and height are reported as the expected values of 83.79px and 37.79px.

The tooltip mode has another feature which I'll mention for completeness: when used on a group of paths, it will show values for the width, height and position of the whole group, but you can hold Ctrl (incorrectly described as Shift in the release notes) to see the data for an individual path. But given that the values in the tooltip clearly can't be trusted, this capability is somewhat moot.

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 105

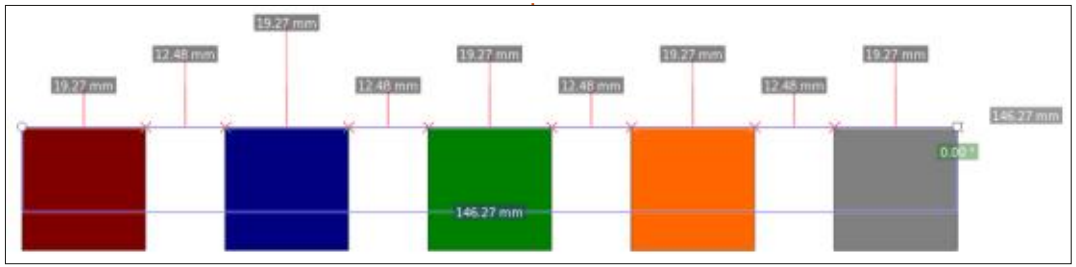
Last time, I described the (unreliable) tooltip mode that was introduced to the Measurement Tool in version 1.0. That wasn't the only change to this tool, but the other addition is, thankfully, a little more reliable.

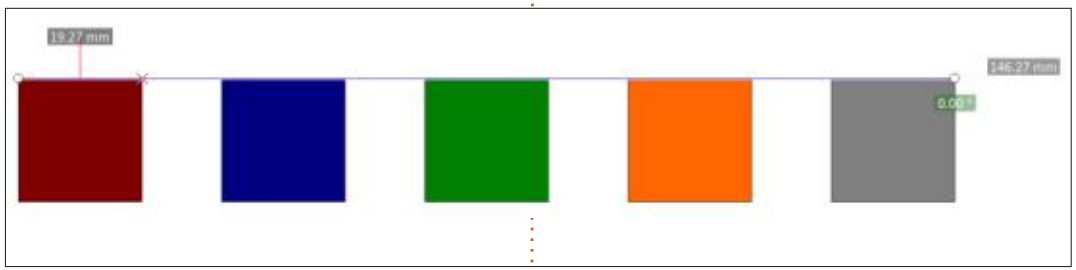
Measurement Tool

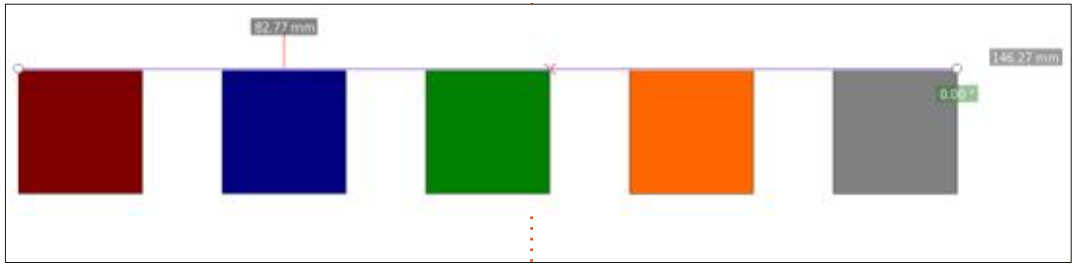
For this simple example, I have five squares – identical apart from their colours – arranged in a line. I've got the "Ignore first and last" option selected on the Measurement tool's control bar (the button outlined in red), and have dragged a measurement line from the top-left to the top-right of the arrangement. As you can see, I'm presented with a single measurement for the length of the line.

But what if we also wanted to see how wide the boxes are? For that we can enable the "Show measures between items" button (the one to the right of the red outline). This shows the distances between each line or point that the measurement line crosses.

That's good, but we already know that the boxes are all the same size, so it would be sufficient to just label one of them. That's where the new button – to the left of the red outline – comes in. The tooltip describes it as "Measure only selected". With this toggle enabled, you'll get additional measurements displayed for selected objects, as well as the overall dimension of your measurement line. By selecting the first box, then drawing the same top-left to top-right measurement



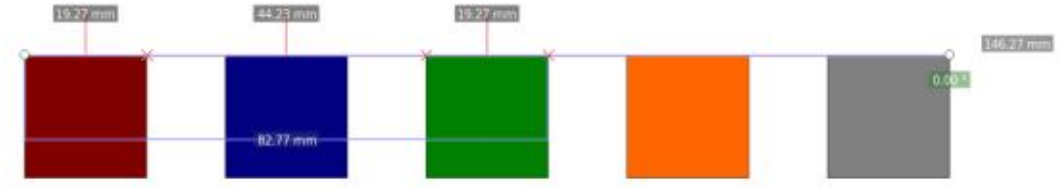




line, we can get a clearer view of the two values we're interested in.

With a single element selected the state of the "Show measures between items" button has no effect. But when you select two or more objects within the path of the measurement line, it changes the result quite significantly. By selecting just the red and green boxes, and with the "measure between items" toggle turned off, this is the display that Inkscape produces:

It's a little hard to make out, but there is a small red cross at the top-right of the green square. The displayed value of 82.77mm represents the distance between the start of the measurement line (the small circle at the top-left of the red square), and that red cross



– that is to say, the entire width of the selection, irrespective of the unselected blue box that happens to lie in the middle.

With the "measure between items" toggle enabled we get a different collection of numbers.

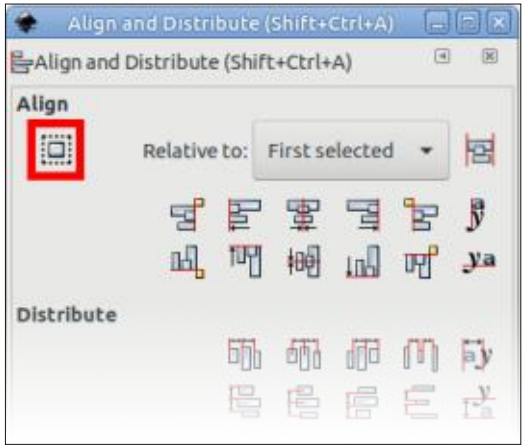
There are now two additional red crosses – at the top-right corner of the red square, and the top-left corner of the green one. In other words, the points that intersect with the measurement line, but only if they're from the selected items. The numbers then show the distances between each pair of marks along the line, as well as our 82.77mm total for the selection, displayed further down (appearing over the blue box in this image). And we still have a total length for the whole measurement line displayed at the right.

This is a great new feature for this tool, giving you the ability to more precisely indicate which parts of a complex drawing should be measured, and which should be ignored. It does have a slight problem in that changing the selection can cause all the additional measurements to disappear. Redrawing the measurement line brings them back though, so it's a small annoyance rather than a major bug.

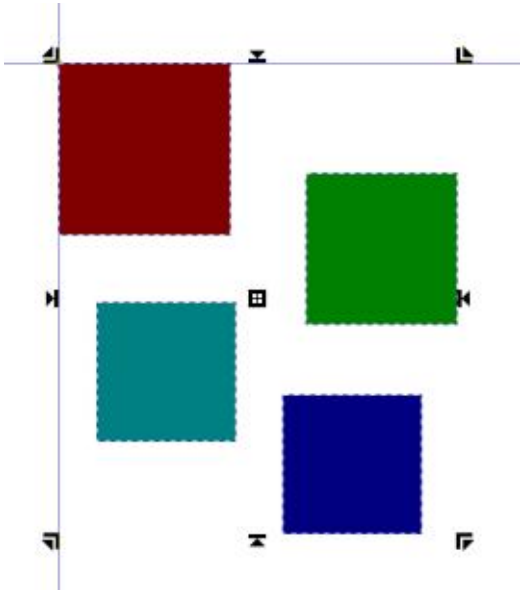
You may have noticed that none of the squares in these images give the appearance of being selected: you can't see the usual dashed line and handles that you would expect when the Selection tool is active. These are suppressed when using the Measurement tool – a good thing, too, otherwise they would clutter and confuse the layout. But it might leave you wondering which objects are selected and which aren't. Worry not! The new tooltip feature, although rather broken in the numbers it displays (see last month's column), does reliably state that each element is either "Selected" or "Not selected" as you move the mouse over them. Although, in most cases, those red crosses are also a bit of a give-away.

On-canvas Alignment

Another new feature in version 1.0 is the ability to align selected objects on the canvas, rather than via the Align & Distribute dialog. Oddly, however, you do need to make a visit to that dialog to enable the feature, even though its implementation is entirely based around the Selection tool. Let's first turn the option on, using the new toggle button which is just hanging out on its own at one side of the Align & Distribute dialog (outlined in red in this image). You can safely close the dialog once you've enabled the mode though, as the setting does persist.



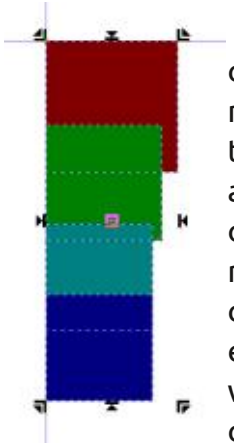
With this toggled on, the Selection tool acquires a third mode. Even the most novice of Inkscape users is familiar with the first mode, indicated by double-ended arrows that allow you to resize selected objects. Once in that state, clicking on a selected object will cycle to the second mode, where a new set of double-ended arrows are used to rotate and skew. With this new toggle button activated, a third click will cycle to the new mode, where a radically different set of handles will greet you.



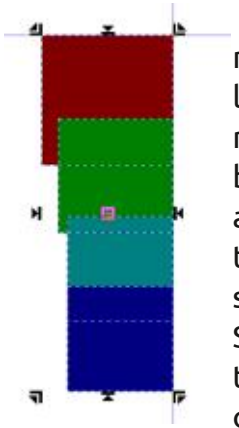
The two thin blue lines are just guides that I've added to make it

clearer where the top and left edges of the selection group lie, to help orient you in the screenshots to follow. Their position on the canvas doesn't change, so, by treating them as fixed references, it should be clearer to see how the items move around on the page.

The handles in question are the eight black icons around the outside of the selection, and the one in the very center, but the first thing to note is that these aren't really handles – not in the sense that the term is used for the other two modes. You can't drag these around. You can try, but all that occurs is that they disappear from the screen until you release the mouse button. These 'handles' are really just buttons that happen to be positioned where the selection handles usually live.

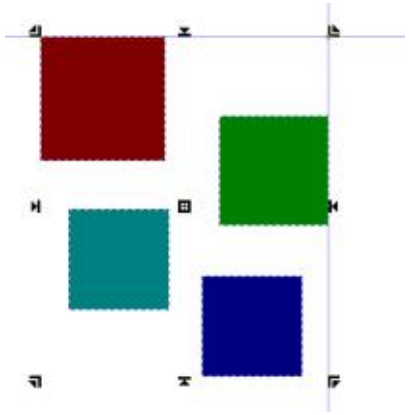


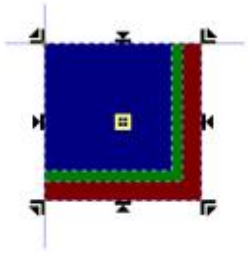
Ignoring the central button for now, clicking any of the others will align all the selected objects to the relevant edge or corner. As an example, here's what happens if you click on the button that's halfway down on the left.

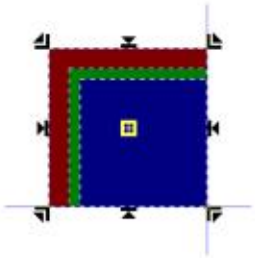


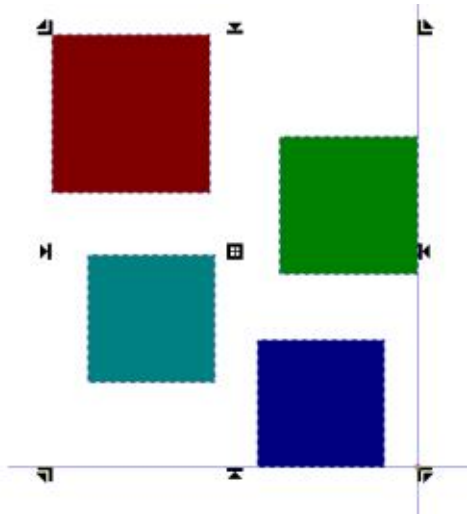
The objects are moved so that the left edges of their respective bounding boxes are all aligned along the left edge of the selection. Hold Shift while clicking the same button, on the other hand, and the right edges of the bounding boxes are aligned along the left edge of the selection instead.

Holding Shift+Ctrl while clicking the handle provides one final alignment option: the entire selection is moved so that the right edge of the selection aligns with its previous left edge. In other words, it's shifted by the width of the selection.





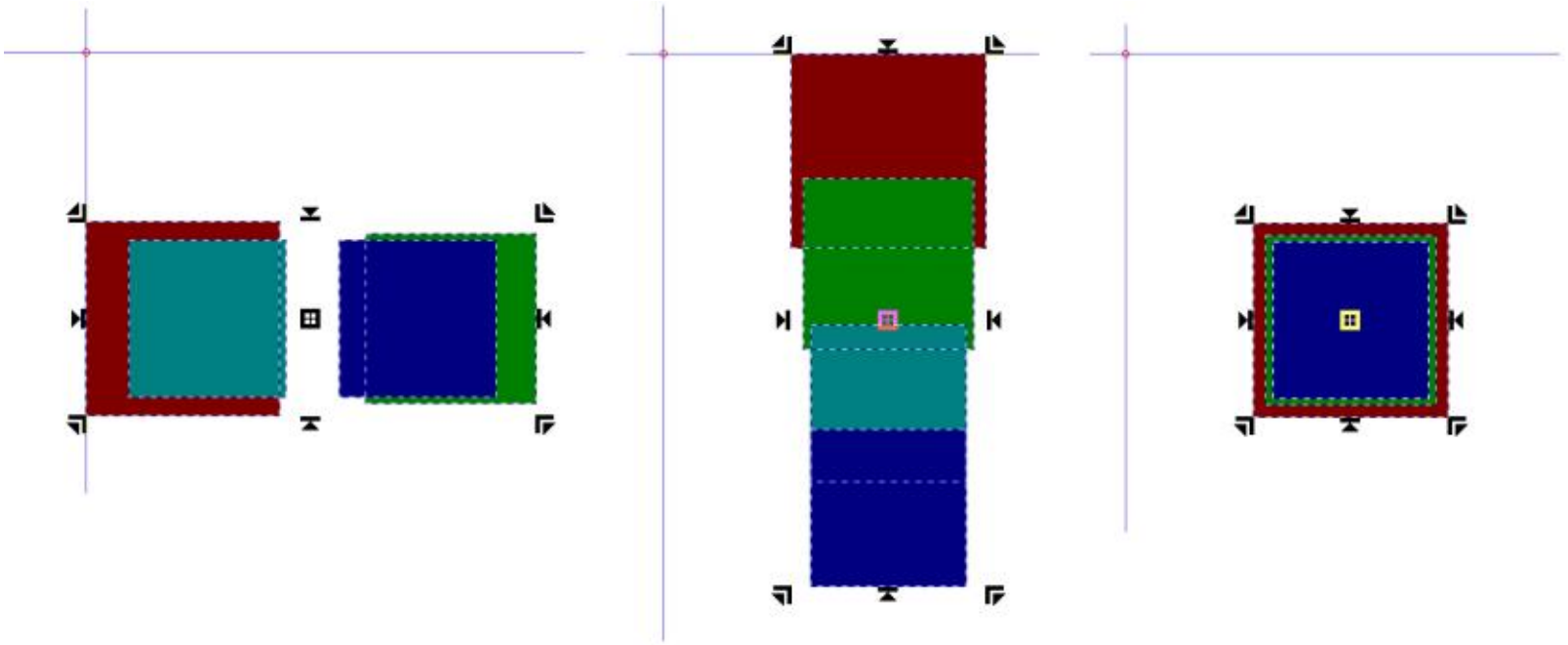




Equivalent movements take place, with the direction changed accordingly, for each of the four buttons on the sides of the selection. The four corner buttons also behave similarly, but align two edges at once. For example, the three images below show what you get when clicking on the top-left button, clicking the same button while holding Shift, and finally when holding Shift+Ctrl.

Clicking the top-left corner button is equivalent to clicking the left edge button, followed by the top edge button (or vice-versa). The same goes for the Shift and Shift-Ctrl variants. The corner buttons are therefore just a shortcut for aligning both horizontally and vertically with one click, but you can still perform this operation as two steps if you wish. This is especially important to note if you wish to use different alignments for the two axes – for example, a click on the left edge button, but a Shift-click on the top edge button.

The center button will move objects vertically so that they're aligned on the horizontal axis of the original selection group, while a Shift-click will move them horizontally to center them on the vertical axis. One useful trick is to perform a click followed by a Shift-click (or vice versa) to center the objects along both axes (i.e. stack them up with a common center point). These three options are shown in the image below. Note that Shift-Control clicking on the



center button has no effect.

All of the alignment capabilities offered by the new mode are also available via the Align & Distribute dialog, with the "Relative to" pop-up set to "Selection Area", but the on-canvas buttons are probably a bit more obvious and intuitive.

In my opinion there are three things missing from this new feature which would have improved it:

- A toggle button on the Selection tool's control bar as well as (or instead of) the one in the Align & Distribute dialog. As the Selection tool is the way in which you interact with the new capabilities, it doesn't make sense to me that you need to open the dialog every time you want to toggle this feature on or off.
- A keyboard shortcut for toggling this mode on and off.
- A way to use other types of "Relative to" alignment. I most commonly use "First selected" or "Page", and it would be great to have easy access to those modes via the on-canvas handles. Perhaps a duplicate pop-up in the tool control bar, or leveraging the unused Alt key to provide a second set of alignment options.

Nevertheless this is a nice addition to the standard tools which, hopefully, will gain in features and prominence with future releases.

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 105

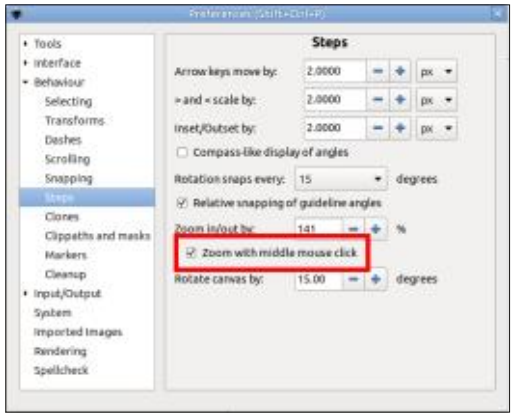
Yet again, last month's News section of Full Circle Magazine managed to sneak in some Inkscape announcements that arrived after the deadline for this column, so apologies for parroting information you probably already know.

First, Inkscape 1.1 alpha is available for testing. This is the initial alpha release, but there may be others – not to mention betas and release candidates – before version 1.1 sees an official build. If you want to help make that release as stable as possible, then please consider downloading the alpha version, testing it with your typical workflow, and reporting any issues. Rather than provide a link directly to this alpha, which may be outdated by the time this magazine hits the wires, I suggest visiting the News section of the Inkscape website (https://inkscape.org/news/). and following the links in the latest relevant article.

Alongside 1.1 alpha we also saw the release of version 1.0.2. This is another bugfix release, addressing several issues that appeared in 1.0 and 1.0.1. At the time of writing, there doesn't appear to be a 1.0.2 snap package available, but there is a PPA, as well as AppImage and Flatpak versions available from the Download section of the Inkscape website. If you wish to use Inkscape on a non-Linux system, there are, of course, builds for Windows and MacOS. Since FCM is primarily a Ubuntu-oriented publication, I will mention that I had problems using the AppImage version on Ubuntu Mate 18.04. Although it initially appeared to work, any operation that opens the file picker – such as loading and saving files – caused the application to immediately crash. As usual, I had the best results with the PPA.

There are only a couple of new features in 1.0.2, but they're ones that a lot of users have been clamouring for. Both are actually new preferences, so to find them you'll of course have to install 1.0.2, then open the Edit > Preferences dialog. The first new option can be found in the Behaviour > Steps panel, and allows you to turn off the default behaviour of zooming in when the user clicks the middle mouse button on the canvas (and zooming out when doing the same with Shift):

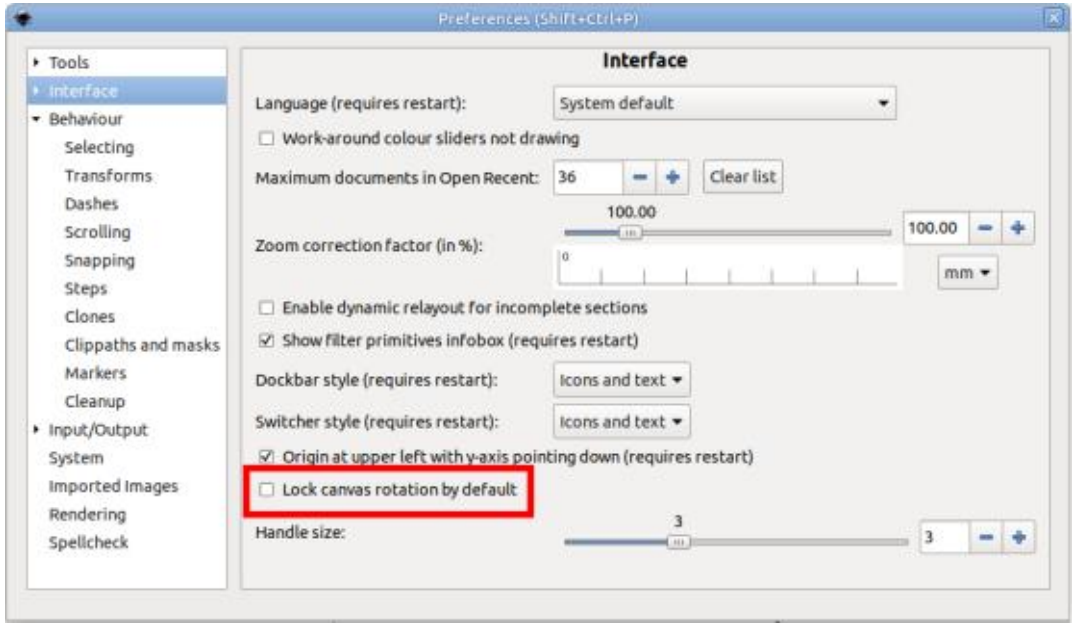
I can't say I've ever triggered this behaviour by accident, despite



being a frequent user of middle-click-drag to pan the canvas, but if you are plagued by this problem,\ you can now disable the feature entirely. It may well be that this problem occurs more for frequent users of graphics tablets; certainly that's the case with the issue that has given rise to the second new option.

You may recall that one of the headline features for 1.0 was the ability to rotate the canvas. For mouse users the best way to trigger this is to hold Ctrl-Shift and move the mouse wheel, however that will rotate in steps (15° per 'click' of the wheel, by default). For an unconstrained rotation you also have the option of holding Ctrl whilst click-dragging with the wheel or middle mouse button. The middle mouse button is often exposed on the stylus of a graphics tablet, where it's useful for panning the canvas, but many users of such devices found they were accidentally triggering the canvas rotation by mistake, due to sloppy timing when releasing the Ctrl key from a previous operation. As a result the Inkscape forum has seen a flurry of requests for the ability to turn off the canvas rotation feature.

If you wish to disable this feature, there's a new option for it in the Interface panel of the Preferences dialog. This doesn't, however, remove the "R:" field from the bottom right of the Inkscape window, so you can still force rotation by changing the value in there, by typing, rolling the mouse wheel over the field, or



right-clicking to bring up the context menu.

That's it for the new features in 1.0.2, with everything else being bug fixes, mainly for issues that the majority of users won't ever encounter. But there is one class of fixes that definitely warrants a mention: this release fixes the text-to-path regressions that were introduced with version 1.0. For the full rundown on these problems, see part 100 of this series, or the subsequent YouTube video I created (http://www.youtube.com/watch?v=lx5nRCu7AKk), but here's a brief reminder.

Originally the behaviour of Path > Object to Path, when applied to a text object, was to create a single complex path. This was subsequently changed to create a group of paths, one for each letter – yet internally the ability to convert text to a single path remained. It was even exposed via the UI, whether by accident or design: using Path > Union on a text object would convert it to a single path.

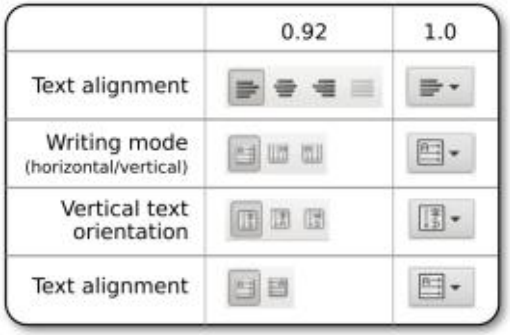
A rewrite of the path operations code for 1.0 broke this internal ability. Path > Union now behaved the same way as Object to Path. Furthermore, other features which required an implicit conversion to a single path stopped working in a variety of ways. You could no longer trivially apply an inset, outset, dynamic offset, or linked offset to a text object.

With 1.0.2, normality is restored. Path > Union once again creates a single path, and the various offset functions work once more. In my opinion, this fix alone makes it worth upgrading to 1.0.2 if you've already made the leap to the 1.x series, and removes one of the biggest reasons for sticking with 0.92.x for any readers who have been reticent to move on.

All this talk of text objects leads nicely into the next topic for this series. We're returning to the new and updated features that arrived with version 1.0, of which Inkscape's text support received more than its fair share of changes. We'll start with the most obvious of the user-facing changes: the revamped Text tool control bar.

There's no doubt that the Text tool control bar in version 0.92 was a little unwieldy, featuring many, many buttons. The new version simplifies this clutter by collapsing several sets of radio buttons into drop-down menus. For example, the four buttons used for text alignment now take up less than the width of two buttons. The old discrete buttons and their new combined drop-down replacements are shown in the table below.

The tool bar has also lost a

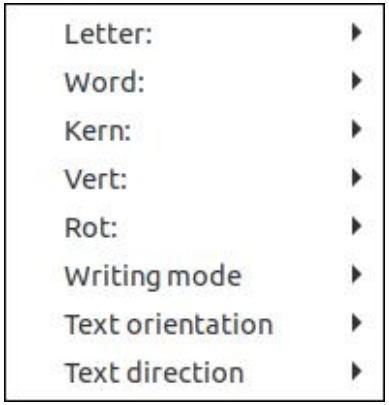


couple of icons completely. The confusing "Show style of outermost text" button has been relegated to the waste bin. The existence of this toggle was the result of an attempt to better represent the underlying complexities of SVG text. In practice very few people understood the implications of using this button, and most users would either ignore it entirely, or randomly turn it on and off while trying to reset the other controls on the bar. Also gone is the "?" button which was only relevant in the relatively rare case of the user mixing different line spacings within a single block of text (more on this later). Good riddance to them; anyone who really needs the

control they offered probably has the technical skills to make their changes via the XML editor, or by editing the raw text of the SVG file.

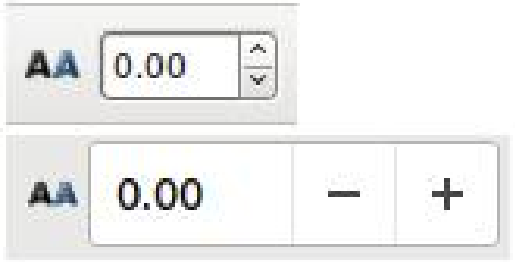
With two buttons removed, and several others reduced to drop-downs you might expect the toolbar in 1.0 to be more compact than its predecessor. Unfortunately the opposite is true. Despite having fewer controls, the new toolbar occupies even more horizontal space than the old one, barely fitting onto the width of an HD monitor. Anyone trying to run Inkscape on a narrower screen, or just with a reduced window size, will find that the controls inevitably overflow into a horrible pop-up menu, with unnecessary abbreviations and inconsistent colons.

The reason that the toolbar takes up so much additional space



is due to the general trend towards big, chunky UI widgets that are "touch friendly" for use on tablets and phones. In the case of the Text toolbar, this change is particularly exacerbated due to the number of spinbox widgets it holds. Here's a comparison of how spinboxes look on Inkscape 0.92 compared with 1.0:

This is a definite case of "you can't please all the people all the time". As a mouse user, I never



experienced any problems with the smaller buttons of the old widgets, and much preferred being able to get to the justification options with a single click. A tablet user might disagree, considering the extra step of opening a drop-down to be a small price to pay for spinboxes with large, easy to hit buttons. I'd love to see a future release of Inkscape address this by offering a preference to switch between the two styles.

Ironically, this change of style for the Text toolbar is presented in the version 1.0 release notes under the heading of "More Compact Tool Controls Bar". Clearly the Inkscape developers use a different definition of "compact" than the one I'm familiar with!

As mentioned earlier, the removal of the "?" button was one of the changes implemented to make the toolbar more "compact". In earlier releases, this button would be enabled if you created a multi-line text object, but then changed the line spacing for a subset of the lines in the block (e.g. by dragging to select a single line before changing the value in the spinbox). Toggling the button back to its off state would remove the line-specific override, returning the whole block to the same default line spacing.

Without this button, it's still possible to reset all the lines back to a single value. It's as simple as clicking in the text to place the caret (without selecting anything), and then changing the value in the spinbox. If you just want to reset it to the existing value, press the plus button immediately followed by the minus button to nudge the line height up and then down by the same amount.

If, rather than just clicking within the text, you select part of a line so that at least one character is highlighted, changing the line height value will affect that whole line. This also works with a selection that spans multiple lines. This ability to mix your line heights within a block is exactly the same as in 0.92, all that's been removed is a one-click method to revert all the lines back to a single value, which is no great loss.

Flowed Text

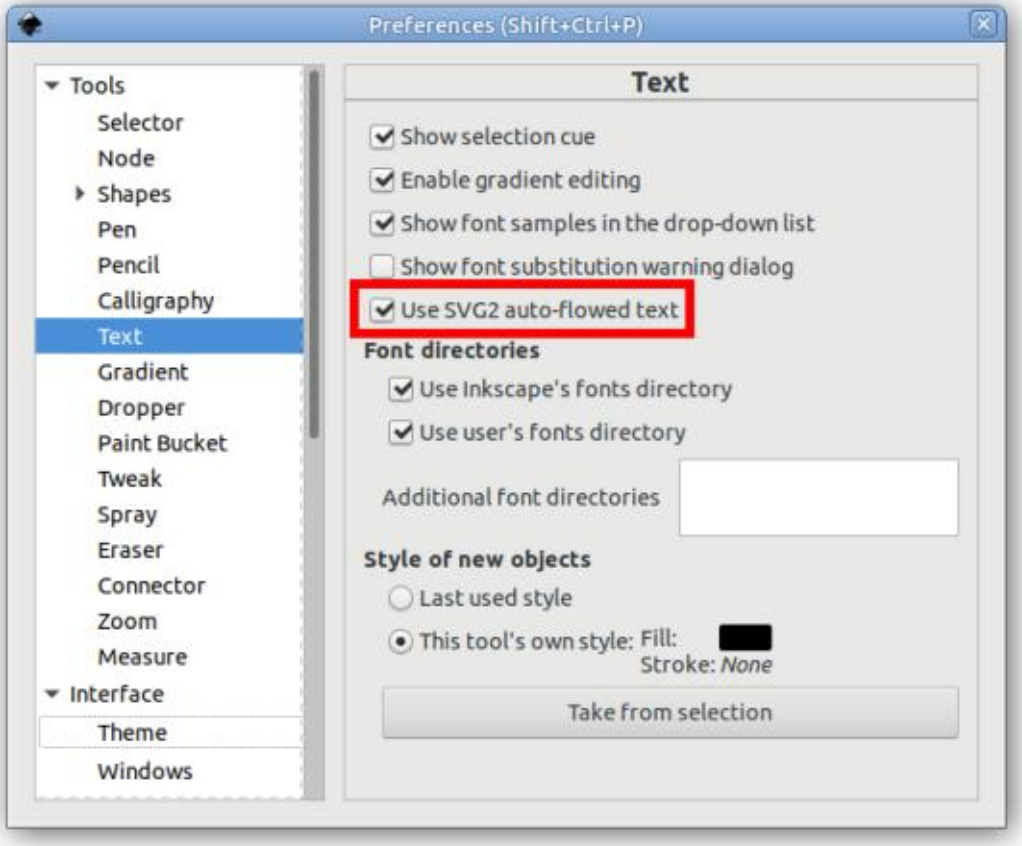
Flowed text has long been a source of consternation for veteran Inkscape users. Support was added many years ago, based on a proposed implementation for SVG 1.2 which looked like it would become part of the official spec. That proposal was not accepted as part of the SVG specification, however, leaving Inkscape with a flowed text implementation that was not supported by the vast majority of other programs, including web browsers.

With the advent of SVG 2.0, however, flowed text has seen

something of a renewal. In part this is due to the SVG Working Group's decision to defer to the CSS standard for many parts of the language, rather than trying to re-implement their own variations. Inkscape 1.0 revamps the existing flowed text support completely, in order to produce something that is compatible with SVG 2.0.

The downside is that it's no longer compatible with the implementation in earlier versions of Inkscape. For most users, this isn't something to worry about, but if you do need to create files that will be edited in 0.92, you can switch back to the older approach by turning off an option in the Inkscape preferences. The quick way to find it is to double-click on the Text tool, which will open the Preferences dialog with the correct panel already selected. Un-check the "Use SVG2 auto-flowed text" option and the flowed text you create will once again be compatible with Inkscape 0.92… and almost no other programs.

There's yet more to describe about Inkscape's new flowed text implementation, as well as other text features that are new to version 1.0, so next month's article will continue to explore this subject.



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 107

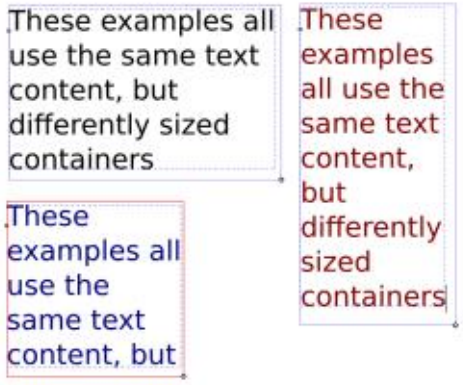
As promised last time, I'm going to continue describing the flowed text improvements in Inkscape v1.0. Previously, I showed you how to turn off the new SVG 2.0 flowed text implementation, in order to produce files that are compatible with Inkscape 0.92, but, unless you've got a specific reason for doing that, I strongly recommend sticking with the SVG 2.0 variety for compatibility with web browsers and future Inkscape releases. This time I'll dig into the gory details of the new features, both from an Inkscape user's perspective, and also with regard to the way flowed text appears in your SVG file.

First, some definitions: flowed text is the sort that will automatically wrap its lines to fill the available area. Usually that area is defined as a rectangle (though Inkscape can also flow text into arbitrary shapes), and the text will be rendered so that each line fills the width of the rectangle as fully as it can. Changing the dimensions of the rectangle causes the text to move around – to "flow" – in order to best fill the width. Ordinary (non-flowed) text refers to text objects that aren't constrained inside a rectangle or other shape, and don't flow to fill the space. Line breaks are explicitly entered, and have to be manually adjusted if you want the layout of the text to change. Whereas flowed text gives layout control to Inkscape, ordinary text keeps that control solely with the creator.

In earlier versions of the program, it was possible to create flowed text by dragging a rectangle with the Text tool selected, then typing your content into it. Ordinary text was created by just clicking on the canvas with the Text tool in order to position the text entry caret, and then typing. Those were your only two options: flowed or ordinary. With Inkscape v1.0, however, there are now two different varieties of flowed text to understand.

As with previous releases, you can click-drag to create a rectangular text box on the canvas when the Text tool is selected. Typing into this box will produce flowed text which, in practice, behaves the same way that flowed text did in earlier Inkscape releases, but with greater compatibility outside the program.

The image below shows a single piece of flowed text that was duplicated twice (and the text color changed). You can see that each container has a small diamond-shaped handle at the bottom-right corner: dragging this resizes the container and re-flows the text, as seen in the black and red text versions. The one with the blue text was achieved in the same way, but the handle was deliberately dragged such that the container size was too small for the text it needs to contain. You can see that

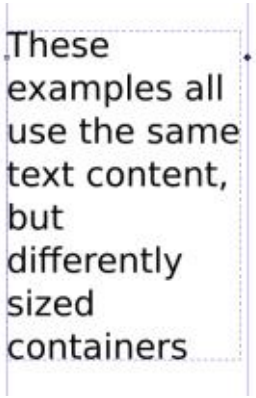


Inkscape draws the container in red in this situation, as a visual indicator that the text has overflowed outside of the allowable space.

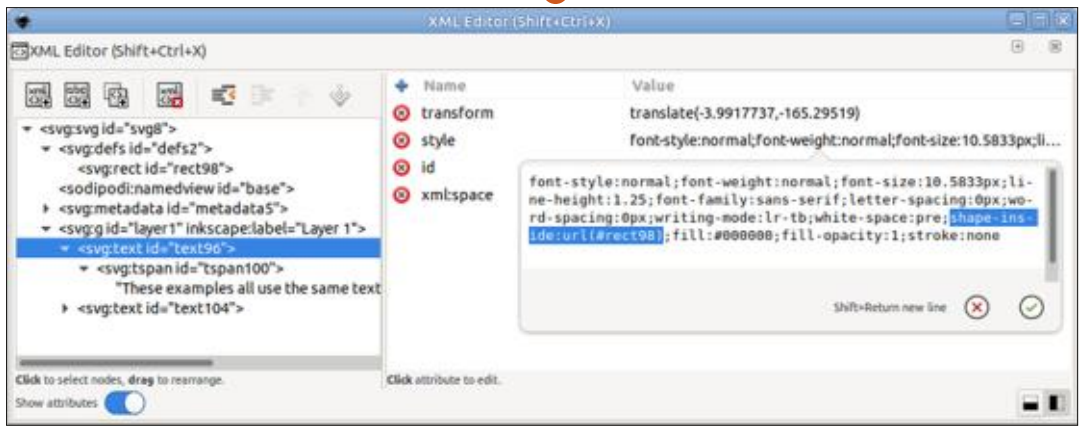
In this example, it's pretty obvious that the text has overflowed the last container, but that's not always the case. When dealing with large amounts of text in a bigger container, you may not notice that the border turns red when you add a word into the middle of the prose. The second flowed text type, referred to as "Column mode", goes some way to addressing this possibility.

Column mode begins life as ordinary, non-flowed text. Just click on the canvas with the Text tool, and begin typing (or paste in some already written content). But, instead of beginning the tedious task of manually entering line breaks, look at the right-hand edge of the text box, where you'll find a small diamond-shaped handle. Start dragging that handle and you'll see a pair of vertical blue guides appear (representing the left and right edges of the "column"), and your

text will be flowed to fit between them.



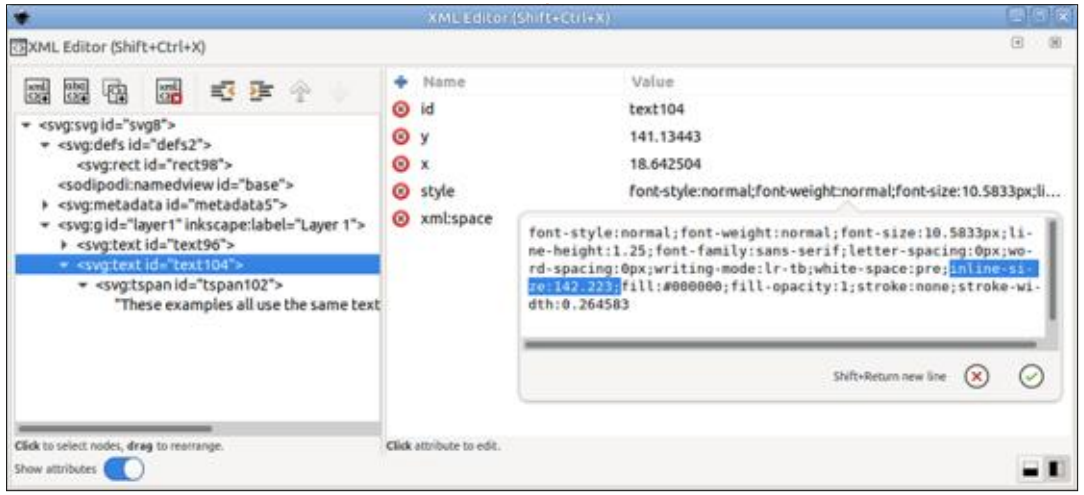
What's important about this mode is that it sets the width of the flowed text, but doesn't constrain the height. This has the advantage that you'll never accidentally clip the end of your text with the flow container, but it also means that your text can flow to be taller than you intend, possibly leading to it interfering with other parts of your design. Where this type of flowed text comes into its own, therefore, is in loose designs where you want to tweak the column width dynamically to see what works best for your particular layout (think posters and leaflets), rather than designs which already have well defined areas of a specific height in which the text has to fit (such as magazines or newsletters).



Technical Details

With two different ways to create flowed text, let's take a look at the technical details behind these new features. Feel free to skip this section if you just want to use them as an artist, and aren't interested in what happens in the XML code.

The first thing to know about both of these approaches is that Inkscape adds an SVG 1.1 compatible fallback implementation when you save your file. This can potentially mean that Inkscape's XML editor is a little misleading: look at it just after creating some flowed text, and you'll see a different structure than if you'd saved it and re-loaded the file. Let's start by looking at the structure that is used as soon as you create your content.



The click-drag type of flowed text consists of two parts: the text itself, and the rectangle that constrains it. The latter is created as a hidden object in the <defs> section of the XML, and is then referenced via the "shape-inside" CSS property, within the style attribute of the svg <text> element. In the image below, you can see the highlighted text on the right which references "rect98", the ID of the rectangle near the top of the left-hand pane.

As you can also see on the left, the <text> element just contains a single <tspan> as a child, which contains the entire text string.

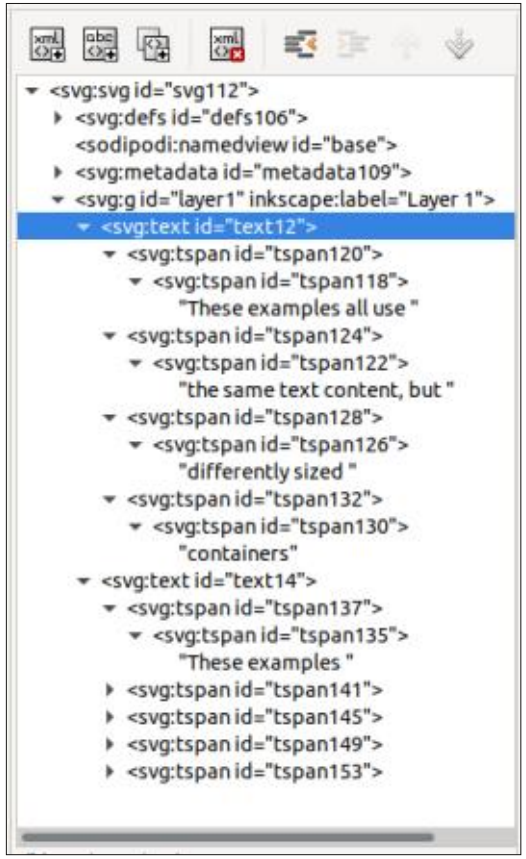
Column mode is fairly similar, but there's no need to reference a hidden rectangle. Instead of the "shape-inside" CSS property, Inkscape inserts an "inline-size" property which defines the width of the column. Once again the text itself lives as a single line within a solitary <tspan> child element.

Unfortunately, both of these approaches are broken in web browsers today. Whilst the "inline-size" method is well supported in

modern browsers, in my testing it appears to have an effect only on HTML content, and does not work for SVG files. The "shape-inside" CSS property, on the other hand, is not currently supported by any browsers. It is part of the CSS Shapes specification, but was dropped from Level 1 of the spec (the one the browsers have implemented), and moved to Level 2 (the one they haven't, yet).

On the surface, therefore, it might seem that flowed text hasn't really advanced much. We've gone from a non-standard, unsupported technique to a pair of nearly-standard, but still unsupported techniques. Yet, if you save a file with flowed text from Inkscape 1.0.x and load it into a browser, the text is visible – line breaks and all. So what's going on?

You can see the reason for this apparent support if you examine the file via the browser's developer tools, or if you load the same file back into Inkscape and look at the XML editor. At first all seems to be as expected: the <text> elements are present, with their corresponding "shape-inside" or "inline-size" properties, but when drilling further down to the text itself, we're faced with a structure like this.



The single <tspan> element, containing the entire text, has been replaced with a separate <tspan> for each line in the resultant output – the same sort of structure you would expect to see if you had manually entered the line breaks. These elements are used for positioning each line, but then a second level of <tspans> is used to style the text. We've gone from a single child, one-level deep, to a collection of two-level children. This is the SVG 1.1 fallback content that allows the current batch of browsers to display the content as it appears in Inkscape.

This change of structure has repercussions. Because it produces the expected visible result in a browser, it will probably serve the needs of 99% of users. But it also changes the nature of the text. What was previously a single string is now broken into separate blocks: this could potentially have an effect on screen readers and search engines, depending on how they treat <tspan> elements. It would definitely have an effect on JavaScript programmers who want to dynamically find or change the text on the page, or had hoped to use code to alter the column width or rectangle size with the content flowing automatically to suit.

This does raise some questions about how these files will fare in the future. If browsers do start supporting "inline-size" on SVG text, or add support for CSS Shapes Level 2, how will they behave when faced with text that already has line-breaks? It's possible that this could result in additional automatic line breaks being inserted which conflict with those that Inkscape has already provided, breaking the intended layout.

But these concerns are purely hypothetical at this point, whereas the problems with flowed text in v0.92 are very real. The changes in version 1.0 are a definite improvement, and make it an obvious choice if you wish to use flowed text in your designs. The details above really matter to only a minority of web developers, so shouldn't prevent you making the switch.

If you really don't want the SVG 1.1 fallback text included in your file, it can be turned off in the Preferences dialog (Input/Output > SVG Export pane). Note, however, that this will affect only newly created flowed text – any previously saved file that already contains multiple <tspan> elements will still maintain that structure.

Other Changes

There are a couple of other changes to mention regarding flowed text, though they're relatively minor points.

One feature of 0.92 was the ability to convert flowed text into ordinary text via the Text > Convert to Text. This would "apply" the line-breaks by converting the content to a SVG 1.1 compatible <text> element with multiple <tspan> children. This no longer works in version 1.0, and has no effect on the structure of the XML. Saving the file will insert an SVG 1.1 compatible version of the text however (as discussed in the previous section), but it also inserts some CSS which will cause Inkscape to still treat such content as flowed.

Essentially if you actually want to convert from flowed text to fixed line breaks in a way that Inkscape will recognise, you have little choice but to manually insert the breaks, or to remove the new CSS from a saved file. As far as I can tell, this now makes the Convert to Text menu option completely redundant, as any attempt to use it simply puts a "No flowed text(s)" message in the status bar, and has no effect on the text or SVG structure.

Rather than fix the line breaks, what if you want to un-flow your words, to revert them to a single line of ordinary SVG text? In this case, the program is oddly contradictory as to the method you use. Version 0.92 offered the Text > Unflow menu option, which still works for click-drag style flowed text in version 1.0. It's always had a nasty habit of moving the un-flowed text quite a distance from the original flowed version for some reason, so if you do use this, and your text seems to disappear entirely, try zooming out and panning around.

For reasons best known to the Inkscape developers, this approach doesn't work for text that has been flowed via column mode. To revert this to a single line, you need to Ctrl-click on the diamond handle used for adjusting the column width. To further add to the confusion, this same technique does not work with the diamond handle at the bottom right of the rectangle used for click-drag style flowed text. Come on devs, how about a little consistency!?

The final change to mention is a small but important usability improvement. If you use the Text > Flow Into Frame option to flow text into multiple shapes on your canvas, the order in which the shapes are filled is now based on the order in which you select them. Previously it used the selection order in reverse, which is less than intuitive, so it's good to see this change make its way into the program. It makes the behaviour more familiar to anyone who has ever used a desktop publishing program, such as Scribus – though I still maintain that Inkscape is a poor substitute for a real DTP application for anything but the most basic of page layouts.

Next time, we'll conclude this part of the series by looking at the support that has been added for new font types in Inkscape v1.0.

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