Full Circle

THE INDEPENDENT MAGAZINE FOR THE UBUNTU LINUX COMMUNITY

INKSCAPE SERIES SPECIAL EDITION

INKSCAPE SERIES SPECIAL EDITION

INKSCAPE

Volume Three **Parts 15 - 21**

Inkscape - Part 15

Written by Mark Crutch

Although Inkscape is a vector graphics program, it does have some support for including bitmap images in your drawings. It's certainly not a fully fledged bitmap editor, and neither is it a desktop publishing program. If you want to airbrush a photo, you would be better off using The GIMP, and if you want to lay out a newsletter, then Scribus would be a better tool.

Getting an image into Inkscape is quite simple. My preference is to drag-and-drop it from the desktop or file manager into the main Inkscape window. Alternatively, you can use File > Import to pull your picture into an open document. File > Open will also do the job, but that will create a new Inkscape document with just the image in, and the document's page size set to the dimensions of the image. Whichever approach you take, you will next be presented with an import dialog (left)).

If you select "Embed" then the image will be included as part of the Inkscape file. For the technically minded it is Base64 encoded, which is a means of representing binary data using text. Unfortunately, this encoding has been designed for robustness rather than efficiency, so will inflate the storage space required for your bitmap by about a third. The advantage, however, is that an embedded image is part of the SVG file, so if you move the document to another location, or even another machine, the bitmap will move with it.

Selecting "Link" will include the location of your bitmap in the SVG file, but not the data that makes up the image itself. If you subsequently move the SVG file, you'll need to move the bitmap with it, or fix the link to account for the change in location. One advantage of linking a file is that, if you make changes to the original bitmap image, those changes will be automatically reflected in the Inkscape document. For an embedded bitmap you would have to remove the current version from the document and then embed the modified version.

Which option to choose depends on what you are doing with the bitmap in your Inkscape drawing. If it's only there temporarily—so that you can trace over it, or use it for reference—then linking is probably the best option. For use as a permanent part of your design, then embedding might be better – especially if you plan to move the Inkscape file, put it on a web server, or send it to someone else. If you're unsure, then I would suggest linking the image; you can always embed it later using the Extensions > Images > Embed Images... menu option.

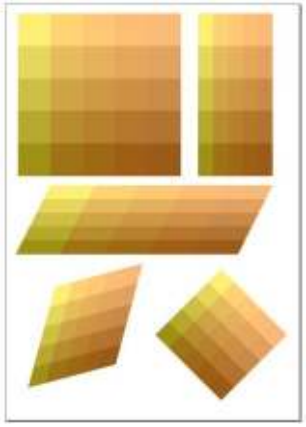
Checking the "Don't ask again" box will mean that future imports will default to your choice of Embed or Link. I prefer to leave this unchecked, as I tend to switch between the two options depending on what I'm drawing. If you do check this and then subsequently change your mind you can switch to the other option, or tell Inkscape to ask in future, via File > Inkscape Preferences... then selecting the Bitmap section and changing the "Bitmap import" option.

Having imported an image, it will appear in Inkscape with the normal selection handles. One thing that might surprise you is the size of the image—imported bitmaps tend to be bigger than you might expect. This is because Inkscape imports them at a resolution of 90dpi, regardless of the size, shape or embedded metadata of the image. At this resolution, a 900 pixel tall image will be 10 inches tall, nearly filling an A4 page.

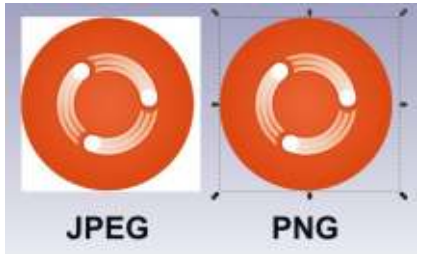
Within Inkscape, it might be useful to think of your picture as being a group of colored squares—albeit a group that you can't enter or un-group. Each

square is drawn at 1/90 of an inch in size, but you can scale it up or down using the selection handles, just as you would with any other object. Scaling like this doesn't change the number of rectangles in the group, it just changes the size and shape of each one. You can also skew and rotate the image, or change its opacity, just as you would with any other group of rectangles. Here's an example of an image made up of only 25 pixels, but copied, scaled and transformed to fill an A4 page. You can see that talking in terms of pixels and dpi quickly stops making sense when you've transformed your pixels into rotated rhomboids!

Sometimes, you don't want the



whole of a bitmap image in your drawing. If your image is a PNG file, then Inkscape will honor any transparency that's present. This can be used to include non-rectangular elements into your drawing. Take the Full Circle Magazine logo as an example: you can clearly see the difference between using importing a JPEG version, which doesn't support transparency, compared with a PNG version, which does.



When protecting sensitive readers from the fleshy expanses of renaissance art, for example, a pair of PNGs makes for a reasonable brassiere, whereas JPEGs lead to overly obtrusive underpants...

Another way to show just a sub-section of an image is to use clipping. This works in exactly the same way as clipping any other object—just draw a clip path on top, select both the path and the image, and then choose Set Clip from the context menu, or Object > Clip > Set from the main menu. Using this with the bottom left corner of our renaissance painting, followed by some rotating and flipping, leads to an image that should be familiar to anyone who has seen Monty Python's Flying Circus.

The hard edges of a clipped image don't suit every requirement, but, as you might expect, masking also works. A simple blurred shape with a white fill, used as a mask, lets you feather the edges of a bitmap image for a softer effect.

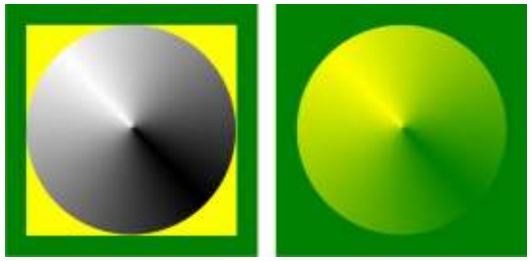
As you may recall from the previous part of this tutorial, masks are just collections of colored pixels—just like bitmap images. Inkscape will happily let you use an imported bitmap as a mask—it's really no different to using a group of rectangles. On first impressions it looks as though using an image as a mask results in something like a photographic negative:

What you're actually seeing is the white of the Inkscape canvas showing through the darker parts of the image, and the color of the masked object appearing where there are lighter parts in the image. By changing the masked object to a lighter color, and using a dark object as the background, relative normality is restored:

It's worth noting that you can mask any sort of object. The examples here all use an image to mask a single rectangle with a flat fill color, but you can use the same technique on an object with a gradient or pattern fill, or even on a group of objects.

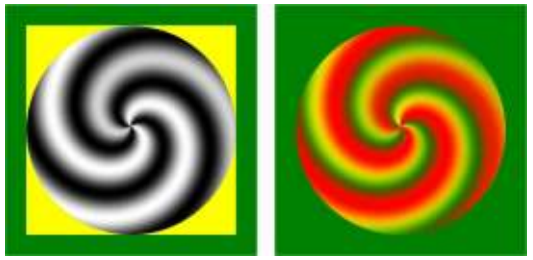
Depending on your source image, you may find that you get better results if you convert it to a grayscale using a bitmap editor such as The GIMP. A bitmap editor will also give you the opportunity to lighten, or darken, or even invert, the colors of the image. If the image was included in Inkscape as a Link rather than an Embed, the effect of your changes will be applied to your Inkscape drawing each time you save the bitmap image in your editor, allowing you to easily experiment with different changes to the bitmap. In my experience Inkscape isn't always perfect at refreshing the screen when the bitmap changes, but scrolling the canvas a little, or changing the zoom level, usually fixes the issue.

Using bitmaps as masks can be another way to get around the limited types of gradients that the SVG format supports. For example, by creating a conical gradient in The GIMP, then using it to mask a yellow square on a green background, it's possible to produce a conical yellow-green gradient that would be difficult to create in Inkscape alone.



Of course this approach doesn't result in a genuine vector gradient, so the accuracy is determined by the resolution of your bitmap. You could get the same effect by simply creating a yellow-green conical gradient directly in The GIMP, and then importing it directly into Inkscape. By using the image as a mask, however, you can still freely change the colors within Inkscape, rather than having to modify the bitmap image each time.

A major limitation of this method is that you can affect the transparency of only one object at a time, so gradients with multiple color stops are a problem. You can work around this by using your mask on a group of objects, or by layering several masked objects on top of each other, but that can quickly become complex. To demonstrate this approach, I've created a bitmap using the “Three Bars sin” gradient from The GIMP, drawn as a spiral gradient. I first applied this to the same yellow square on a green background as used previously, then duplicated it in-place (Edit > Duplicate, or Ctrl-D) before rotating it slightly to give a multi-colored spiral gradient.



As you can see, once a bitmap has been linked or embedded into an Inkscape document, you can pretty much treat it in the same manner as any other object. If you think of it simply as a group of colored rectangles then you won't go far wrong. Don't mistake Inkscape for a bitmap editor or a desktop publishing program, though—The GIMP or Scribus are far better tools for those tasks.

Next time, we'll continue our tour of bitmaps in Inkscape by finding out how to turn them into genuine vectors.

Image Credits

“Venus, Cupid, Folly and Time” by Angelo Bronzino
http://commons.wikimedia.org/wiki/File:Angelo_Bronzino_001.jpg

“La Gioconda” (aka “Mona Lisa”) by Leonardo da Vinci
http://en.wikipedia.org/wiki/File:Mona_Lisa,_by_Leonardo_da_Vinci,_from_C2RMF_retouched.jpg

Mark's Inkscape created webcomic, 'Monsters, Inked' is now available to buy as a book from http://www.peppertop.com/shop/

Inkscape - Part 16

A common reason for importing bitmap images into Inkscape is to convert them to a vector format. This can be done using an automatic tracing process, or by manually tracing the image. I'll cover the automatic process in a future instalment, but for now let's concentrate on the manual approach.

Manually tracing an image is exactly what it sounds like. There's no magic involved, it's simply a case of drawing on top of your bitmap in order to re-create the image using vector objects. It can be a time consuming and tedious process, but, for some images, it's the only practical way to achieve a satisfactory result.

Having linked or embedded your bitmap and scaled it to a sensible size, the first step is to lock it. This will prevent you accidentally moving it as you draw over the top, and can be done in a couple of ways. The best approach is simply to lock the layer that the image is in, using the Layers dialog or the status bar, as described in part 9 of this series. You can also lock an individual object from the Object Properties dialog, which you can access via the right-click context menu, the Object menu, or by pressing Ctrl-Shift-O. The problem is that once you've locked an object you can no longer select it in order to unlock it. The solution is to use the Object > Unlock All menu entry.

For the rest of this article, I'll assume the more sensible approach of locking the layer. Of course you'll now have to create a second layer on which to actually draw. The obvious option is to place your drawing layer above your image layer and then start creating objects. Let's try manually tracing the FCM logo using this approach. If you want to follow along, you can find the image at http://www.peppertop.com/fc/

First drag the image into the Inkscape window, and choose to link it – there's no point embedding it as it will only be in the file temporarily and will be removed once we've finished tracing. Then lock the layer, and create a new one for tracing.

Trying to draw the large background circle immediately presents a few problems. The first is simply that it's very difficult to draw, by eye, a circle of the right dimensions and in the right location. Using the ellipse tool you can force a circle by holding the Control key, but you also need to start and finish at the right coordinates. It would be a whole lot easier if there were some guides to show us where to click.

Inkscape conforms to the conventions of most graphics applications when it comes to creating guides – you simply have to drag them out of the rulers. If you want a horizontal guideline, click and hold inside the ruler at the top of the drawing area and drag downwards, releasing the mouse button to place your guide. Similarly you can drag a vertical guide out of the ruler at the left. If you drag from either ruler, close to the top-left corner, you'll create an angled guide at 45°.

Did you drop your guide at the wrong location? Using the Selection tool you can drag the guide somewhere else, or hold down Shift while dragging to change the angle – hold Control as well if you want to constrain it to the standard rotation angles that are set in the Inkscape Preferences. If you have trouble targeting the thin guideline, you'll find a slightly easier target in the form of a small round handle on the guide at the point where you released the mouse when you first dragged it out of the ruler. This handle is also used as the center when you rotate a guide.

With the Selection tool still active, you can double-click on the line or handle to bring up a dialog

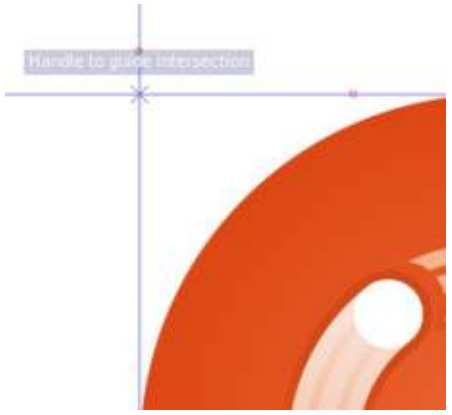
to let you precisely set the location and angle of the guide. You can use absolute coordinates, or enter an offset relative to its current position by checking the "Relative Change" box. This dialog can also be used to change the guide's color, or to delete it completely – although a faster way to do that is simply to place the mouse pointer over the guide and press the Delete key.

With four guides in place, it's a lot easier to see where to start and end when dragging out your ellipse to create a circle. We can make it easier still by using Inkscape's snapping tools to force your cursor to jump to the intersection of two guidelines when it gets close.

The snapping toolbar (shown above) can be confusing – especially as the same icon appears four times! Depending on your setting at the bottom of the View menu, this toolbar may be positioned horizontally at the top of the window, or vertically at the right side. If you can't see it at all, make sure "Snap Controls Bar" is checked in the View > Show/Hide menu. In order to draw a circle, snapping to the intersections of the guidelines, you will need to have at least these buttons pressed:

The first button is used to toggle snapping on and off (you can also use the "%" key). The same icon is also used on other buttons to toggle whole classes of snap targets on and off. To get any useful results from snapping, you'll therefore need to have the first button enabled, plus at least one of the remaining three. In this case, it's the second one that's enabled: hovering over the button to see the tooltip will tell you that it concerns the snapping of "nodes, paths and handles". As we drag out a box to define the size of the circle, the start and end points are treated as handles.

You'll also notice that the last icon is enabled. This tells Inkscape to snap to guidelines. The three enabled buttons on the toolbar therefore simply mean "snapping is enabled – snap handles to guides". Selecting the Ellipse tool and moving the cursor close to the intersection of the guides now will briefly display a small, faint tooltip, telling us that it's snapped the "Handle to guide intersection".



Dragging the circle from here to the opposite corner, we'll see a similar tooltip flash up at the intersection of the other two guidelines. At last we've drawn a large circle, tracing the one from the imported image. Unfortunately it's obscuring the bitmap, making it impossible to trace anything else. It's probably also the wrong color, but we'll come back to that later.

The most obvious way to stop the circle obscuring the image is simply to move it out of the way. It's the approach I tend to use, as there's less fixing up to do afterwards – I just draw each element, then shift it to the side before moving onto the next one. By holding Shift and using the cursor keys to move the object I can ensure that it moves by the same amount, regardless of the zoom level I'm working at. So if it takes 10 presses of the cursor key to move the circle out of the way, so long as I use 10 presses for every other part that I trace they should all end up in the right position, relative to one another.

Another approach is to make your new circle disappear – at least temporarily. By creating a new layer or sub-layer for each object you draw, you can show and hide individual parts of your traced image. This method works well for a few objects, but can become

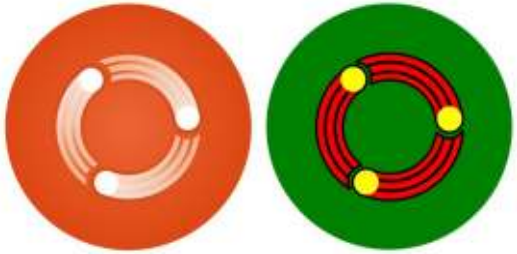
unwieldy when you're tracing something complex. You may also need to move all the objects back to a single layer afterwards, depending on what you want to do with your finished tracing.

Rather than making objects completely disappear you can make them translucent. The Opacity pop-up on the status bar (right-click on the "O:" spinbox) is a fast way to do this – or you could reduce the opacity of the whole drawing layer in the Layers dialog. You'll have to remember to set objects back to their full opacity after you've finished, and even at 25% opacity this approach can sometimes obscure the fine details of the image you're trying to trace.

A final method that's quite common amongst comic artists is to put the imported image layer above the drawing layer, but with reduced opacity. Because the image layer is locked your drawing operations take place on the layer below, so the traced image never obscures the bitmap you're tracing. You can change the opacity of the bitmap layer at any time, to make it easier to see fine details, but because the drawing layer is fully opaque there's no fixing up to do afterwards.

Try all these approaches to see which method you prefer – and don't rule out the possibility of mixing different methods, even when tracing a single image.

After tracing each element of the bitmap this is the end result. You can barely tell which version is mine, and which is the original...



If you're particularly observant, you might have noticed that the colors in my version don't quite match those in the original image. In fact the colors in my version don't even look good together, creating a clashing design compared with the more subtle combinations of the source. This was actually deliberate – I find it much easier when tracing parts which have to lay on top of each other to give them garish and contrasting colors. It makes it more obvious when something is missing or needs to be moved above or below another object, and can help you to keep track of which parts you've traced and which you have yet to work on.

The next task, therefore, is to restore the original colors. Inkscape provides a "dropper" tool for this which is enabled via the "eye dropper" icon on the toolbar, or by pressing F7 or "d".

Before activating the tool you should first select the object(s) that you wish to color. In this case we'll start with the large green circle in the background. Having selected your target object, you can activate the dropper via the icon or keyboard shortcut. Now you simply have to click on any point in your Inkscape drawing – whether on a vector object, an imported bitmap, or even the background – and your selected object will be filled with the color of the pixel you clicked on.

Sometimes you don't want to adopt the color of a specific point, but rather an average of the colors in a small area. This is often the case with JPEG images, where lossy compression can lead to individual pixels being quite different from the overall impression of color your eye perceives. Rather than just clicking with the color picker, you can click and drag in order to define a circle. Your selected object is filled with an average of all the pixel colors within the circle.

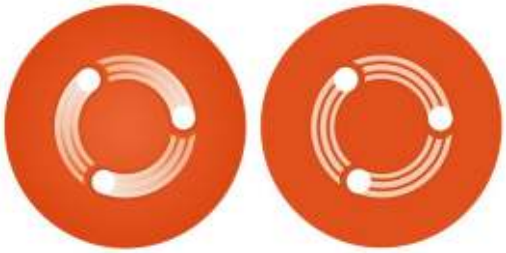
If you need to set the stroke, rather than the fill, of your selected object, you can use the same click or click-and-drag technique, but holding the Shift key. Holding down Alt will fill your object with the inverse of the selected color, whilst holding Shift and Alt will do the same for the stroke. To round out the dropper's tricks, if you press Ctrl-C while the tool is selected, the color of the pixel it's hovering over will be copied to the clipboard as an 8-digit hexadecimal number (RGB and Alpha).

Do also pay attention to the tool control bar when the dropper is selected. There are only two buttons up there which are used to determine whether the alpha level should be included when picking a color, and when assigning that picked color to an object. They have no effect if you're picking an opaque color, but, when dealing

with translucent objects, they can change the results considerably. Usually I leave them both enabled, but, if your dropper seems to be giving you the wrong results, it might be worth experimenting with them.

With the dropper used to pick suitable flat colors our traced design is now a lot closer to the original.

The final step to matching the



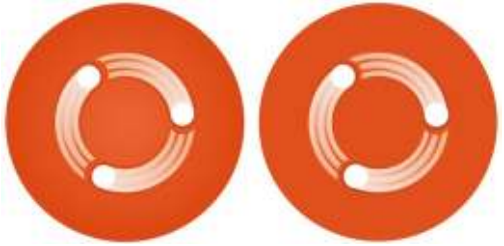
original is to try to reproduce the gradients. Bear in mind when tracing bitmaps that SVG (and therefore Inkscape) supports only linear and radial gradients, so some seemingly simple images and logos can be difficult to trace if they use other gradient types.

Reproducing a gradient is similar to reproducing a flat color: we'll still use the dropper tool to copy the color from one part of our drawing to another. The difference, however, is that instead of using it to set a single fill or stroke color, we'll use it to set the color of each stop on the gradient. In this case the gradients simply have a start and an end stop, but there's no reason why you couldn't also reproduce gradients with several stops.

Select the object you want to modify and give it a gradient on the fill or stroke, as necessary. At this point it's more important to have the right number of gradient stops rather than worrying about the colors, so you might want to choose something garish once again. Switch to the gradient tool, if it's not already active, by using the icon on the toolbar or by pressing Ctrl-F1 or “g”. Now you can drag the gradient stops into the correct positions. At this stage those garish colors can make it look like we've taken a step backwards.

With the gradient tool still active, click on one of the gradient stop handles in order to select it (it should change to a blue color). Now select the dropper tool, and you can pick colors from the drawing just as you did previously, except this time they're assigned to the gradient stop rather than the whole object. With the dropper tool still active, you can click on another gradient stop to select it – you don't need to switch back to the gradient tool each time – and then choose its color from the drawing. Repeat as necessary until all your gradient stops are colored.

After setting a few gradients and picking the colors for their end stops from the original bitmap image, we've finally got a traced version in all its vectorized glory.



Because we started with a fairly high resolution bitmap, the difference only really shows up if you zoom in.

The last step, of course, is to delete the bitmap layer, and with it the source image. At this stage you may want to move all your objects onto a single layer, if you need to, and perhaps group them.

Although this article has nominally been about tracing bitmap images, the information about guides and the dropper tool is applicable to Inkscape in general, even if you use only vector objects. Next time, we'll continue with the same subterfuge – continuing to look at tracing bitmaps, but sneakily introducing more general tools in the process.

Mark's Inkscape created webcomic, 'Monsters, Inked' is now available to buy as a book from http://www.peppertop.com/shop/

Inkscape - Part 17

Written by Mark Crutch

Having manually traced a logo in the previous instalment, this time we're going to trace something a little different: a hand-sketched comic-book character. The basic principles are the same as before, but working from a sketch will allow us a little more freedom than rigorously reproducing a logo.

The first step is to obtain a suitable image for tracing. If your drawing skills are as bad as mine, then I suggest finding someone who knows what to do with the pointy end of a pencil to give you a hand. In my case, I've called on Vincent Mealing – the co-creator of my webcomics – to create a headshot of "Frankie", a character from our "Monsters, Inked" strip.

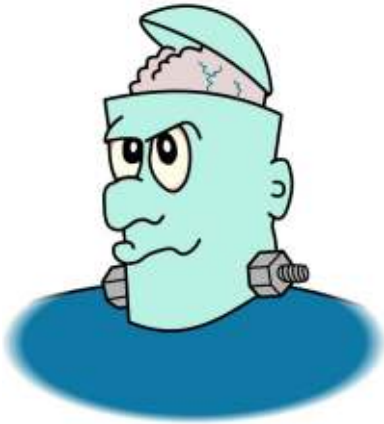
After scanning and saving the sketch as a JPEG image, dragging it into Inkscape presents the familiar import dialog. As usual, I chose to link to the image, because it's only a temporary addition to the file. Locking the layer prevents the sketch accidentally being moved or selected. With that layer locked, we need to create another layer to actually draw on. When tracing a logo or photograph, I prefer to draw on a layer above the bitmap. For tracing a pencil sketch, however, it's often easier to place your drawing layer below.

The obvious problem with drawing below the sketch is that the white background of your scanned image will obscure anything that you draw. The solution to this is to set the Blend Mode of the Pencil Sketch layer to "Multiply". Anything you draw on the lower layer will show through the background, but your pencil marks will still be visible on top to help guide you.

With those preparations in place, it's time to start drawing. Here, I've used the Bézier tool ("B" or Shift-F6) to draw part of Frankie's skullcap. You can clearly see how the pencil marks are still visible, making it easy to trace the lines of the sketch.

You can continue tracing the sketch using the Bézier tool, and tweaking the paths with the Node tool ("N" or F2) to quickly produce an acceptable result. Depending on the style you want for your final image, a simple trace such as this might be all you need to do, or you might want to add highlights, shadows, gradients and textures to give it a little more depth.

Tracing or drawing using simple objects and paths can sometimes feel a bit sterile. Lines that keep the same constant width don't add much character to a drawing, and optical tricks like fading out lines using a stroke gradient quickly lose their effect at large sizes. As is so often the case, Inkscape is constrained by the SVG format, which doesn't contain any notion of variable line thickness. Despite this omission, Inkscape does have a few ways to create more dynamic and variable lines, but each approach comes with compromises in order to maintain SVG compatibility.

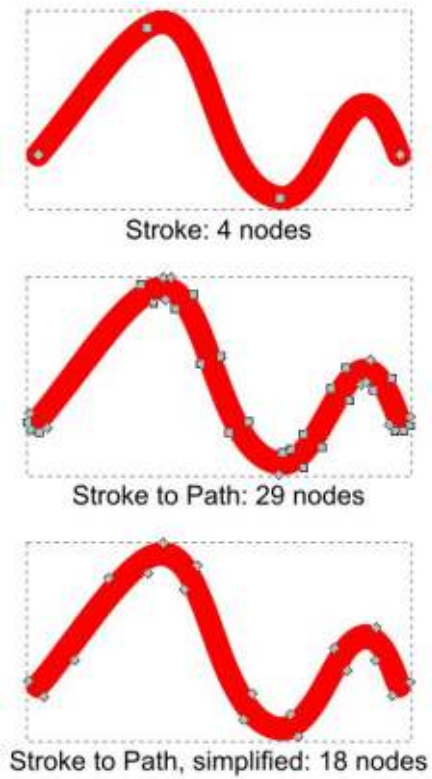


The biggest compromise – shared by all these methods – is

that you can no longer create a filled path with a stroke, and instead have to create two separate objects, one for the outline and another for the fill. Your outline will no longer be a simple stroke, but will itself be a filled path. If you modify the shape of your outline, you'll need to also change the shape of the fill to match, so I recommend just drawing your outlines at first and then only adding fills once you're happy with your final design. This should all become a little clearer as we progress through some examples.

An easy way to tweak the thickness of your outlines is to draw them as strokes, then convert the strokes to paths using the Path > Stroke to Path menu item, or CTRL-ALT-C keyboard shortcut. The effect of this is most obvious if you look at a thick stroke before and after conversion, using the node tool.

The first image shows the original stroke – a simple squiggle with only four nodes, and a constant width. By converting the stroke to a path we end up with a filled object that matches the shape and size of the original,



except now the number of nodes has increased significantly. In this case, it's obvious that some of those 29 nodes aren't really required and can be deleted. You can perform this operation manually if you want precise control over the outcome, but Inkscape also offers an automated option in the form of the Path > Simplify menu option (CTRL-L).

Using Simplify once will try to reduce the number of nodes without changing the shape or size of your path too much. Pressing it repeatedly will try to reduce the number further, taking more and more liberties with the shape as it does so. Sharp corners tend to be the first to suffer, but if you keep pressing CTRL-L frequently enough you'll ultimately end up with something that bears little resemblance to the path you started with. Using the Simplify command is therefore a trade-off between fidelity to the original shape, and the number of nodes you're left with. If the Selection tool is active, you can keep an eye on the number of nodes in the Status Bar at the bottom of the Inkscape window. If you go too far, Edit > Undo (CTRL-Z) will take you back in the opposite direction – press it enough and you'll eventually get back to the original stroke.

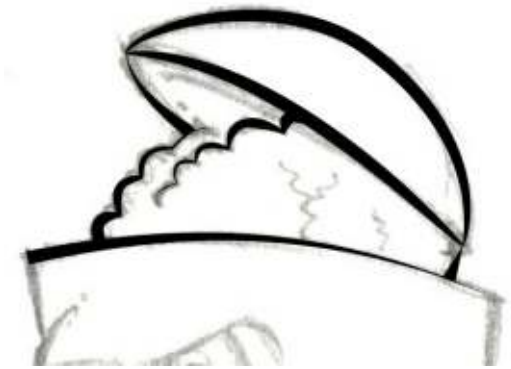
In this case, pressing CTRL-L just once was sufficient to reduce the node count from 29 down to a more manageable 18. Manually adjusting the positions of those nodes gives us the variable width outline we're looking for, as you can see in this second pass at Frankie's skullcap.

As you might imagine, converting your strokes to paths, and then manually editing every node, can be extremely time consuming – however, if you have the patience and skill, it's the best way to have complete control over your drawing.

Inkscape does have a quicker way to achieve a similar result, by letting you select from a few preset path shapes when you draw your line. “Few” is the important word here: currently your choice is limited to three predefined shapes – two of which are essentially the same – although you can use a path from the clipboard if you want something different. To activate this feature, when drawing a Bézier curve, use the “Shape” drop-down on the tool control bar to select “Triangle In”, “Triangle Out”, or “Ellipse”.

The first two will both draw your path as a triangle. The difference is simply whether the wide end is at the start of your line and the pointed tip is at the end, or vice versa. “Ellipse” draws your path as an ellipse – fatter in the center and thinner at the ends. “None” turns off the shaped lines entirely, taking you back to drawing normal strokes.

Unfortunately, there's no simple way to adjust the width of the triangle's base or the ellipse's center, so these shapes can be a bit too heavy for some lines, and a bit too light for others. Using just these shapes gives us another variation of Frankie's head to consider.



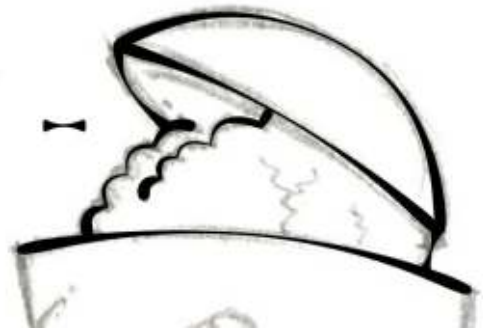
Inkscape's tiny palette of line shapes is a real issue compared with other competing applications. Triangles are okay, but what if you don't want your line tapering to nothing? And whilst an ellipse lets you create lines that bulge in the middle, it's no use if you want one that thins in the middle instead.

Although it's not possible to add your own shapes to the drop-down menu, the “From Clipboard” option does at least offer some additional flexibility that lets you work around the limited list of defaults. In order to use this, you first need to create a path that will be used as your line's shape. In order to create a line that thins in the middle, for example, you require a shape that thins in the middle: a smoothed out dog bone or bow-tie design.

This path will be stretched to cover the length of your Bézier curve, so make sure you draw it at the right sort of scale for your image. When you're ready, you have to put it on the clipboard by selecting it and either copying (Edit > Copy or CTRL-C) or cutting (Edit > Cut or CTRL-X). Now select the Bézier tool again, change the Shape drop-down to “From Clipboard”, and draw your curve as usual.

You can continue to draw new curves, and they'll all use the same shape, until something replaces it on the clipboard. Because of this I prefer to copy, rather than cut, in case I need to put the shape back on the clipboard later. In practice, there's no live connection between the shape path and the Bézier curve, so once you're finished with it, you can safely remove the shape from your drawing without any change to the shaped curves you've created.

Once again we'll use Frankie's cranium to demonstrate the result. I've left the original bow-tie in view to make it clear how the thickness of the final curve relates to the size and shape of the path.



You can, of course, mix and match various shaped paths within a drawing. Looking back at the examples, it's clear that different parts of the image work best with different shapes. Whether you use triangles, ellipses, or the clipboard, you can select the shaped curve and use Path > Object to Path (CTRL-SHIFT-C) in order to create a path that is more suited to manual editing. Note that you don't use Stroke to Path in this case, because the shaped Bézier is implemented as a closed path, not a simple stroke – and don't forget to keep an eye on the number of nodes created, and Simplify if necessary.

As you might expect, the features introduced in this article apply to more than just manual tracing of comic characters. You can Simplify any path, convert any stroke to a path, or use shapes

when drawing a Bézier curve, irrespective of whether you're tracing a sketch, a logo, a photograph – or just drawing freehand, with no reference image to trace.

In the next instalment, we'll continue to outline the image of Frankie using tools that are more suited to freehand drawing using a graphics tablet, rather than the less-than-fluid movements of a mouse. In the meantime, why not try using some of these techniques to trace an image of your own? Or, if you prefer, you can download the sketch of Frankie from www.peppertop.com/fc/ and try to replicate some of the examples shown here.

Mark's Inkscape created webcomic, 'Monsters, Inked' is now available to buy as a book from http://www.peppertop.com/shop/

Quick Review

by Jimmy Naidoo

The Toshiba Satellite C850-F0155 is a low end notebook computer supplied with no operating system. It features a 15.6" LCD display, full keyboard with numeric keypad, Intel 1000M cpu, 2GB DDR3 1600MHz ram, 320GB 5400rpm hard drive, DVD multi drive and a 6 cell 4200mAh battery. It weighs 2.3kg. I recently purchased one for my parents and installed Ubuntu 13.04 on it.

Display

The 1366x768 display is ok when viewed directly but is quite poor when viewed from oblique angles. It performs poorly in bright sunlight.

Hardware

The keyboard is about average for a low-end notebook, as is the touch pad(supports multi-touch). The hard drive is reasonably quick and quiet. Audio is tinny as per usual for a low-end notebook. The biggest surprise is the cpu, which is a lot better than I expected, though it does get rather hot at times. The gpu is quite good also, no problems playing back hd content or basic gaming (SuperTuxKart, Battle for Wesnoth, etc.). The unit is well built and does not flex like some similarly priced units.

Battery

The Lithium-ion battery offers around 2.5hours of normal, continuous use. Recharging is swift, under an hour from totally flat to fully charged.

Ubuntu

Ubuntu 13.04 installed quickly and has worked flawlessly so far. All hardware is natively supported. The system boots and shuts down very quickly and the Unity interface seems perfectly suited to this portable device.

Summary

The Toshiba Satellite C850-F0155 is hard to beat at the price. Most similarly priced notebooks have weaker cpu's, smaller screens, or lower build quality. The only negative seems to be the rather short battery life.

Inkscape - Part 18

Last time I traced a sketch of my comic strip character, "Frankie", using the Bézier tool. First I used the standard SVG stroke, which creates a smooth outline of constant width. Then I converted the stroke to a path in order to vary the width of the outline manually. Finally I used Inkscape's "Shape" option to create variable width outlines.

In addition to the Bézier tool, Inkscape offers a couple of other methods for drawing variable width outlines. I've split these out into their own article because, in my experience, they both particularly benefit users of graphics tablets and can be difficult to use effectively with a mouse. The first is referred to variously as the Pencil tool or the Freehand tool. The keyboard shortcut, on English versions of Inkscape at least, is either F6 or "P" – so "Pencil tool" is a better mnemonic.

The Pencil tool (left) is best thought of as a freehand version of the Bézier tool. With the latter you place nodes at specific locations and Inkscape draws a path that connects them. With the Pencil tool, however, you draw a path and Inkscape places the nodes for you. The few items on the Pencil tool's control bar are very similar to those of the Bézier tool: you still have the Shape pop-up, with its limited list of options, if you want some variability in the path's width; but you also have a "Smoothing" control that's specific to this tool.

When you draw a line using the Pencil tool, the Smoothing value determines how accurately the final path will follow your scribbles. Setting this to a low value will create a path with lots of nodes which accurately records every bump and hiccup that you make as you draw your line. Conversely setting this all the way up to 100 will result in a path that only honors the start and end points of your line, with a curve that vaguely corresponds to the direction of your movements.

This example shows my efforts to trace Frankie's nose using a mouse, with the Smoothing set to 1, 25, 50, 75 and 100. The paths clearly become smoother with each increase in value, but in doing so, the finer detail of the shape is lost. By checking the status bar when each path is selected, the reason becomes clear: increased smoothing results in fewer nodes in the path. In this case the paths have 548, 70, 8, 4 and 2 nodes respectively.

I find that it's usually easier to produce a smooth line with a graphics tablet than with a mouse, so stylus users may be able to get away with a lower smoothness value and still produce acceptable results. Remember that you can always simplify the path using CTRL-L, so it's often better to keep the smoothness value a little on the low side, and tidy up the results afterwards.

One very satisfying use of this tool – at least for those of us who can't draw particularly well – is to set the smoothness fairly high and the shape to "ellipse", then freely sketch with a stylus or mouse. The smoothing turns your jittery paths into swooping impressions, and the ellipse shape gives something of a brushwork feel to the image. You probably won't sell the results at a gallery, but it does make for a quick and easy way to create a rough sketch that you can then refine using other tools. Here you can see the results of two minutes

spent playing with a graphics tablet – I'm sure many readers will be able to do much better.

Whereas all the options for the Pencil tool can be used with a mouse, the Calligraphy tool (CTRL-F6 or "c") has options that are available only if you use a graphics tablet. To be able to use all its features requires a tablet and stylus that measures both pressure and tilt angle – effectively limiting the tool for anyone who doesn't have an expensive Wacom tablet. The tool can still be used with a cheaper tablet, or even with a mouse, but not all the features will be available.

The Calligraphy tool (left) does not produce simple, clean Bézier curves. Instead, it produces complex filled paths that often have hundreds of nodes. Making a change to a path drawn with this tool is not for the faint hearted. The control bar

(above) has a lot of buttons, sliders and options, reflecting the complexity of this tool and the range of effects it can produce.

A good starting point is the selection of presets. These can be found in the pop-up menu on the left, offering presets called "Dip Pen", "Marker", "Brush", "Wiggly", "Splotchy" and "Tracing". The last option, "Save..." allows you to add your own choice of settings as a preset. As you can see from this image, the first four presets give quite different effects. These were scribbled using a cheap graphics tablet that tracks pressure but not tilt angle. You can see that increasing the pressure as the line moves from top to bottom results in thicker lines for three of the presets – "Marker" pays no attention to pressure.

I've omitted the "Splotchy" preset from that collection because I think it's fairly useless with its default settings. In particular, the width slider is set all the way up at 100, which results in extremely thick line ends that are too large for practical use. Dragging the width down to a lower value produces a far more usable result – but then it's no longer "Splotchy" and you may as well just save your own preset with a different name. In this test image, the black line uses the standard Splotchy width of 100, whereas the red line is set to 25. One thing to note is that this preset doesn't use the tablet's pressure – the thinning of the lines is dictated by the speed at which you draw.

The final preset, "Tracing", does something a little different to the others. Rather than the thickness of the line being dependent purely on how you draw, it also depends on what else is on the canvas when you draw. As you draw over darker objects, the line will become thicker, and over lighter objects it will be thinner. Unfortunately it never quite drops to zero, so you'll always have thin wispy lines even on the whitest of backgrounds. Here's an example in which I've just scribbled over my scan of Frankie. While my scribbles are over the white background, the line width is at its minimum, but as soon as I hit part of the character the line width thickens, resulting in a blobby approximation of the image beneath.

Although the presets give a good idea of the range of effects that can be produced with the Calligraphy tool, you can, of course, tweak the controls to create your own individual pens and brushes, then save your favourites as presets. Unfortunately, there's no way to delete a preset within Inkscape, but if you really need to remove one you can edit the Inkscape preferences file in a text editor to delete the relevant section of XML data. On a Linux system, the file is ~/.config/inkscape/preferences.xml, and if you search for the name of your preset you'll find a section that looks something like this:

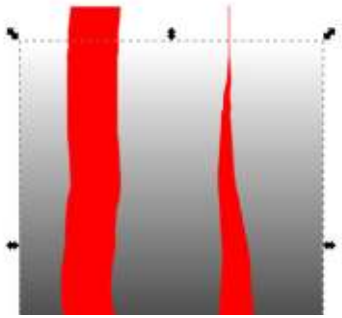
```
<group
      id="dcc7"
      width="44"
      mass="0"
      wiggle="0"
      angle="30"
      thinning="30"
      tremor="10"
      flatness="0"
      cap_rounding="1"
      usepressure="0"
      tracebackground="0"
      usetilt="1"
      name="Medium Splotchy"
/>
```

With all copies of Inkscape closed, first make a backup of the file, just in case. Then remove everything from the opening "<group" string to the closing "/>". Make sure not to delete parts of the other "group" sections around it. Finally, save the file. When you next start Inkscape you should find that the preset has gone.

After the presets pop-up, the next widget on the tool control bar is a slider to alter the nominal width of the line. As with other sliders in Inkscape, there's a right-click context menu with some presets, but usually it's easier to just drag the slider to approximately the width you want. The reason this is only a nominal value is that the line's thickness can be changed by any or all of the next three controls on the bar.

The first button switches pressure sensitivity on, which is useful only if you have a tablet that reports pressure. With this enabled, light pressure on the stylus will produce thinner lines, and strong pressure will produce thicker lines – but only up to the value set by the width slider.

The second switches on the feature we saw with the "Tracing" preset whereby the line thickness changes depending on the darkness of the objects you're drawing over. Tracing over light objects will create a thinner line, while tracing over dark objects will create a thicker line. Tracing over a black object will result in the thickest line – the size set by the width slider. This image shows a line with a width of 50 drawn over a white-to-black gradient with the tracing option first disabled and then enabled.



The "Thinning" control could equally have been called "Fattening", as it can produce either effect. For any non-zero value, the line thickness is changed according to how quickly you are drawing. Positive values will reduce the thickness, negative values will increase it. Again there is a context menu with some sensible values. This feature is most commonly used with positive values to simulate a brush or fountain pen, where fast strokes tend to produce thinner lines as less paint or ink is deposited on the page.

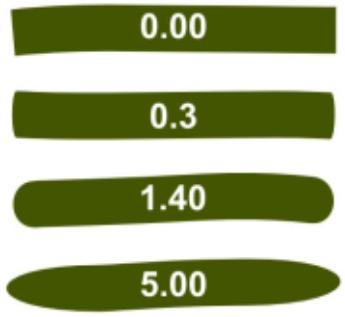
The next three controls on the toolbar all need to be considered together. They alter the angle of the simulated calligraphy nib that is at the heart of this tool. Think about the wide, flat shape of a broad nibbed fountain pen, and about the effect that has on the line you draw. By changing the angle of the pen, you affect the shape of the line, and these controls attempt to mimic that.

The first directly sets the angle of the nib between -90° and +90°. If you have a suitably professional tablet, you can enable the button to the right of the Angle control to allow the nib's angle to be set by the tilt of the stylus. But it's the third control, "Fixation", which affects the line the most. With this set at zero the angle is constantly changed to match the direction you're drawing in – resulting in a fixed thickness of line. With Fixation at 100 the angle is entirely set by the Angle spinbox and Tilt button – resulting in a line that is thick when drawn perpendicular to the nib angle, and thin when drawn parallel to it. Setting this control to

values in-between affects how much of the angle is governed by the direction of the stroke, and how much by the Angle and Tilt controls.

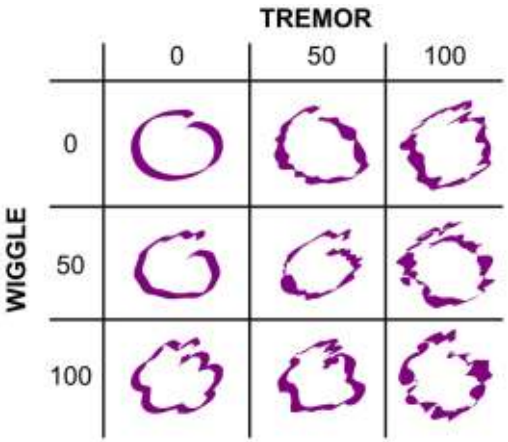
The best way to get a feel for the Angle and Fixation controls is simply to experiment with them. As usual they have context menus, and selecting the menus' default values of 30° angle and 90 for the fixation produces a passable fountain pen style for signatures or calligraphic flourishes.

The Caps control allows you to define how rounded the ends of your lines will appear. A value of zero will produce a square line end, with increasing values making the ends bulge out to semicircles, and ultimately to long ellipses. As usual the context menu has some sensible default values labelled, the effects of which you can see in this example.



Finally we come to the Tremor, Wiggle and Mass controls. Let's get Mass out of the way quickly, as it's the least useful of the three. Essentially it tries to simulate some inertia in your calligraphy pen, by making the line you draw lag a little behind your mouse or stylus movements. It can help to smooth out erratic hand movements, much like the Smoothing control of the Pencil tool. Unfortunately, the range of values far exceeds anything that is likely to be useful. Although you can drag this control all the way up to 100, anything over about 10 results in so much lag that it's almost impossible to produce the shape you want, rendering 90% of the scale essentially useless!

Tremor and Wiggle are far more useful controls. On the surface they both produce similar results – adding some degree of randomness to your line. But whereas Tremor introduces randomness in the thickness of your line, Wiggle randomizes the position a little – wiggling the line up and down. The two are both dramatically affected by the speed with which you draw and can, of course, be combined to introduce a large amount of randomness.



The Calligraphy tool can produce some interesting artistic effects, especially when used with a graphics tablet, but the results can be very difficult to edit. Consider something as simple as a little swirl, drawn with a stylus:

The version on the left is a simple Bézier path with a stroke, drawn using the Pencil tool. The one on the right was drawn with the Calligraphy tool, and is a filled path. The difference becomes clear if you look at the status bar, or switch to node editing mode – whilst the Pencil tool has produced a simple path with only four nodes, the Calligraphy tool has resulted in 42!

The Pencil and Calligraphy tools both add more options to your arsenal when using Inkscape to manually trace an image. They can, of course, also be used as creative tools in their own right – especially if you have a graphics tablet and some artistic talent. For those of us with very little artistic talent, however, I'll be delving into Inkscape's tools for automatically tracing bitmaps in the next instalment.

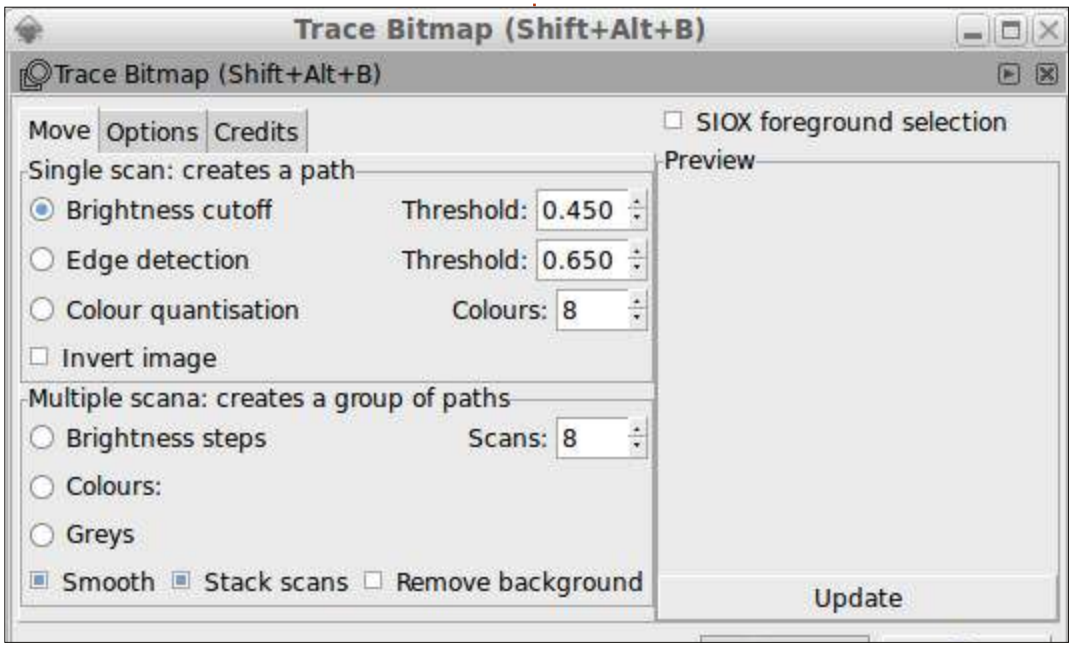
Mark's Inkscape created webcomic, 'Monsters, Inked' is now available to buy as a book from http://www.peppertop.com/shop/

Inkscape - Part 19

The past few articles have presented various techniques and tools for manually tracing a scanned sketch in order to create a vector outline. All that manual work can produce some impressive results, but it does take a while. Fortunately, Inkscape also has an automated tracing tool that can often produce acceptable results in a fraction of the time.

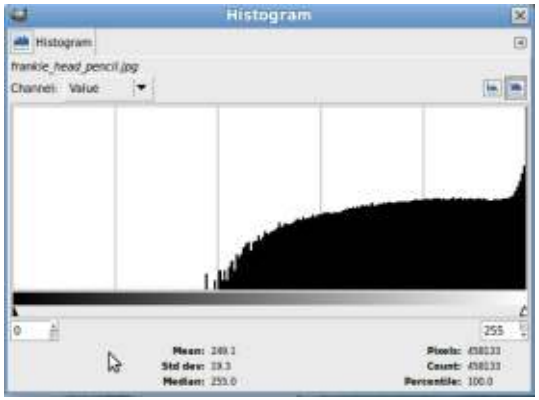
Inkscape's tracing code is based on the venerable Potrace command-line tool, but does some additional pre-processing of bitmaps before they're passed on to the underlying algorithm. You can open the Trace Bitmap dialog using the Path > Trace Bitmap… menu entry, or by pressing SHIFT-ALT-B.

This is one dialog in Inkscape that could really do with a little UI love. It's cramped, unintuitive, contains typos, and the spinboxes don't have the nice context popups of most similar controls in Inkscape. But with a little explanation of the various options, it becomes functional enough even if it's not going to win any prizes for design.



The first thing to note is that the "Move" tab has a pair of groupboxes, "Single scan" and "Multiple scans". As the titles indicate, these result in different traces. The former creates a single path and is useful when you want a clean, hard trace. It's ideal for creating a solid outline from a sketch, or for reproducing a single color logo. The latter creates multiple paths which are grouped together, and is better for converting color or grayscale logos or photographs. Continuing our efforts to create a nice vector version of "Frankie" from the sketch that was introduced in Part 17, I'm going to concentrate on the Single Scan options in this article.

With the sketch imported into Inkscape and selected, clicking the Update button in the Trace Bitmap dialog fills the preview area with… a rather disappointing white rectangle with just a few speckles of black. The problem is that our pencil sketch is made up of shades of light gray which fall below the default threshold required by the Brightness Cutoff option. This method of pre-processing the trace simply converts the dark pixels that fall below the threshold to black, and converts any that fall above it to white. By looking at a histogram of the sketch in The GIMP, it's clear that there is little content to the darker left-hand end of the scale.



There are two fixes for this: either the source image has to be made darker, or the threshold needs to be raised. Taking the latter approach, increasing the threshold to 0.90 (it runs from 0.00 to 1.00) gives a much better looking preview after clicking on the Update button.

Picking the right threshold level for any given image is largely a matter of trial-and-error. Usually, images with strong dark colors require a low value, whilst those with lighter hues will need a larger value. In an ideal world, this dialog would show a histogram of the image, and let you set the threshold by dragging a marker along it, dynamically updating the preview as you go. In lieu of such niceties, you have little choice but to tweak the threshold, update the preview, tweak it a little more, update it again, and so on until you achieve an acceptable result.

When you do finally get a preview that looks promising, you simply have to click the OK button to start the tracing process. Most of the time this completes almost instantly, but if you're tracing a particularly complex image, or using a particularly slow machine, you may have to wait a little longer. During the tracing process the OK button will be disabled; the best indication that it's finished is when the button becomes enabled again.

At this point you will have a new path in the main Inkscape window, positioned exactly on top of the bitmap image. It will also be automatically selected and the raster image below will have been deselected. Unfortunately, this means that if your trace doesn't look right, you can't just change the threshold and hit OK to try again. Instead you have to move or delete the bad trace, then re-select the bitmap, and then finally you can adjust the tracing parameters in the dialog and try once more. It's only a few steps, but when you're trying to hone in on a suitable threshold by trial-and-error, it's a few steps too many.

You should always ensure that you move the final trace away from the bitmap image when checking the result to avoid the original image obscuring any holes or gaps in your trace. Here's how the finished Frankie trace looks:

It's not too bad, but there are several areas where the lightness of the pencil marks and the grain of the paper have conspired to break up the outline. A closer view of the eyes shows this effect quite clearly.

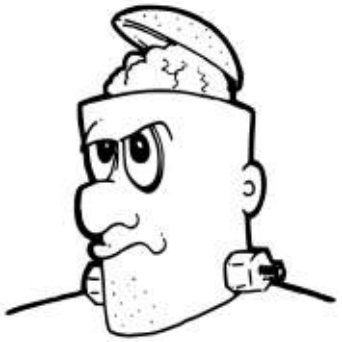
Sometimes this very rough appearance is exactly the right artistic effect, but, more usually, the point of creating a vector image is to give you something a lot smoother. You can try increasing the threshold before tracing again, but often this results in lines that are too thick and heavy – although it does usually fill some of the problem gaps in the process.

Practically, though, if you want good results from auto-tracing, you have to have a good source image to begin with. This means blocks of flat, contrasting colors rather than tints, gradients, and thin lines. A few minutes spent in a bitmap editor can save you a lot of time in Inkscape later. In practice, I never use auto-trace on a pencil sketch. A sketch might be suitable for manual tracing, but there's just not enough contrast and clarity for Inskcape and Potrace's algorithms to do a reliable job. Here's how Vince and I actually create the vector outlines for our 'Monsters, Inked' comics:

- Create a pencil sketch to decide on the shape and position of the characters and objects.
- Ink over the pencil sketch using black ink and marker pens.
- Erase any pencil marks as thoroughly as possible.
- Scan the image.
- Load the scan into The GIMP, and adjust the contrast even further to give a clear distinction between black and white.
- Use the eraser tool in The GIMP to remove any stubborn pencil lines that were picked up when scanning.
- Trace the image in Inkscape using the Brightness cutoff option with a suitable threshold (usually the default of 0.45 is fine, given the preparation above).

After going through these steps, the result is much better

than the Swiss-cheese trace we had earlier. There are still some areas that need to be manually tidied up – mostly where the tracing process has filled small areas – but, overall, we've got a vector that's crisp and clean, and is a good representation of the artist's original intentions.



You can download an inked copy of Frankie's head from the link at the end of the article if you want to try for yourself.

Now, let's take a look at the other two algorithms in the Single Scan section of the dialog. I find these to be less useful than the simple Brightness Cutoff method, but the results vary greatly from image to image, so it's always worth giving them a try if you're not getting the result you want.

The Edge Detection method, unsurprisingly, runs the bitmap image through an edge detection algorithm before vectorising the result. Edges are defined by changes in brightness within the image – a transition from dark to light or vice versa. The Threshold value sets the amount of change that is needed in order for a pixel to be considered to be an edge. Higher values mean that only really obvious edges are counted, which can lead to broken lines. Set it too low, however, and almost any color change is counted as an edge.

It may seem obvious but edge detection works best on images with strong edges. Boldly colored logos or black-and-white line art can give good results, but, as always, you may get better results if you tweak the source image in a bitmap editor first. For example, trying this method with the Full Circle Magazine logo worked reasonably well but kept losing the shape of the inner circles as the difference in brightness between those and the gradients was too small. By using The GIMP to convert the logo to grayscale and to adjust the color curve for better contrast, I was able to create a trace that preserved the outline of the circles.

The Color Quantisation algorithm takes a fundamentally different approach. In this case, the bitmap image is first reduced to the specified number of colors, each with an index number. This simplifies gradients and soft edges down to solid blocks of color. Then, a black-and-white image is created, splitting the colored blocks between black and white depending on whether the index of the color is even or odd. It's this black-and-white image that is finally passed to the Potrace code to produce a path.

It sounds more complex than it is, so I've simulated the process using The GIMP – although the end result isn't quite the same as that produced by Inkscape as the exact details of the algorithm it uses are different. Starting with an image of the Mona Lisa, the first step is to reduce the number of colors. I've chosen to reduce it to 8 colors – it's this value that's set by the Colors spinbox in the Trace Bitmap dialog.

Our color-reduced image now has a fixed palette, with each color being identified by its index – a count of its position in the palette.

Because the Potrace code expects a simple black-and-white image, the final step before tracing is to reduce this palette further. This is done by converting all the odd indexes to black and all the even indexes to white.

That's the approximate process – now let's see what Inkscape actually makes of an 8-color quantisation trace of the Mona Lisa...

The results from the Color Quantisation method can vary wildly as you change the number of colors. A lower number tends to produce larger filled areas, losing details. Higher numbers preserve the details a little better, but result in a path with lots of nodes. Some values will result in the indexes being changed so that the black-and-white image appears inverted. You can see this clearly in the preview when you click on the Update button. In this case, simply check the Invert Image checkbox and hit Update again. This checkbox can also be used with the other two tracing methods, and can be particularly useful for tracing a light image on a dark background.

Automatic tracing works well for some images, and poorly for others. In almost every case, there will still be some manual cleaning up to do, so knowing how to use the node editing tools, and how to trace by hand, will still be invaluable skills. Most importantly, try to get a good image to trace from in the first place, even if that means some work in a bitmap editor.

Links:

Potrace:
http://potrace.sourceforge.net

"Frankie" and other images:
http://www.peppertop.com/fc/

Mark's Inkscape created webcomic, 'Monsters, Inked' is now available to buy as a book from http://www.peppertop.com/shop/

LibreOffice Special Editions:

http://fullcirclemagazine.org/libreoffice-special-edition-volume-one/

http://fullcirclemagazine.org/libreoffice-special-edition-volume-two/

http://fullcirclemagazine.org/libreoffice-special-edition-volume-three/

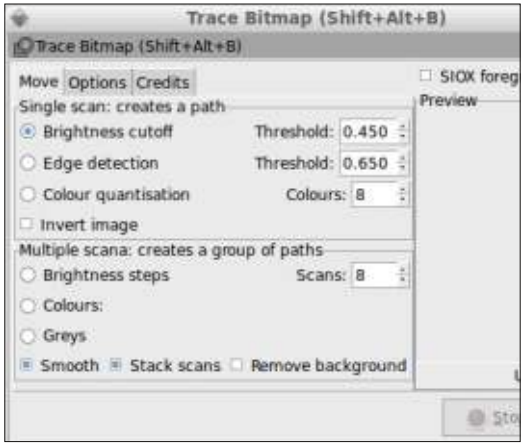
Inkscape Special Editions:

http://fullcirclemagazine.org/inkscape-special-edition-volume-one/

http://fullcirclemagazine.org/inkscape-special-edition-volume-two/

Inkscape - Part 20

In this instalment I'll be continuing to look at the Trace Bitmap dialog. Previously I covered the Single Scan algorithms that are used to create a single path from a bitmap image, and which tend to work best on simple line art. This time it's the turn of the Multiple Scans section of the dialog, which creates more than one path, and can often be a better option when dealing with colored images and logos.

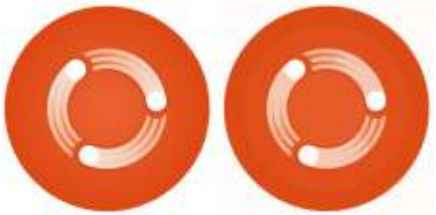


You may recall from last time that the Potrace code within Inkscape only traces a simple black and white image. The three Single Scan algorithms represent different ways to reduce an image down to a single collection of black and white pixels. The Multiple Scans option, on the other hand, creates a collection of different black and white images, and then passes each of them to the Potrace algorithm individually. The multiple paths that are created are then assembled into a single group before being inserted into your Inkscape document.

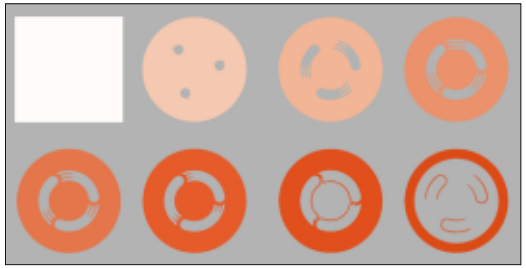
This part of the dialog has radio buttons for selecting one of three algorithms. Based on the top half of the dialog, you would be forgiven for thinking that the Scans spinbox applied to only the Brightness Steps algorithm, but it actually sets the number of paths that will be created regardless of which algorithm you choose. Similarly, the checkboxes at the bottom apply to all three algorithms.

Let's begin by tracing the Full Circle Magazine logo. For this example, I'll use the "Colours" algorithm, with 8 scans and the "Smooth" and "Stack scans" checkboxes enabled – I'll explain all these options a little later.

Selecting the logo and clicking the Update button suggests that the final result will be okay, so I can go ahead and click on the OK button to perform the trace. The result looks like this, with the original PNG image on the left and the traced version on the right.



It's not a perfect trace – though that was never to be expected, given that we've reduced it to 8 colors. But it's not too bad, and does represent the original image quite well. By ungrouping the 8 paths and separating them out against a gray background we can get a clearer view of the paths that have been created.



There are a couple of things to note about these paths. The first is that there's a square path which is used as the background for the final trace. This obviously results in our round logo becoming square, even though the original image was a PNG with transparent corners. The second observation is that the paths stack up on top of each other: each scan sits on top of the one before it, obscuring the lower one and just letting it show through where holes have been left – the final image is created by successively hiding parts of the lower objects.

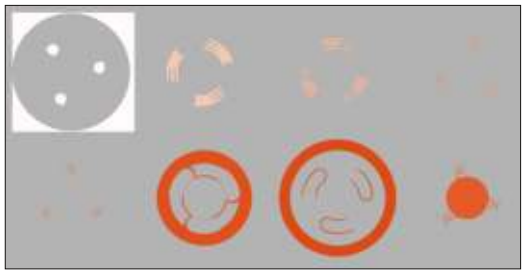
Consider the three white circles in the final logo. As you can see, there is no single trace that contains three white circles. What you see in the final trace is the white background showing through the holes that have been left in each subsequent layer in the stack. Suppose you wanted to tidy up those circles a little. You would potentially have to modify seven of the eight paths!

And what of the square

background? You can remove it by checking the “Remove Background” option in the Trace Bitmap dialog, but that just creates 7 paths rather than 8. You get all the same paths as before, but the background path is removed. However, because the three white circles are the result of the background path showing through, we actually end up with three holes in the traced image instead.

In this case, it's not too tricky to fix up the results, regardless of whether you've chosen to automatically remove the background or not. As a general rule, if you're tracing a rectangular image with no transparency then there's little to be gained from removing the background. However, for tracing images where transparency is important, it's usually best to either check that box, or to manually remove the background layer afterwards. Be aware that you might have some fixing up to do, though.

The way in which the paths sit on top of each other, with later paths obscuring those below, tends to give the best visual results. But not all Inkscape users are using it for artistic reasons; there's a vibrant community of people who use it to create files for vinyl cutters, plotters and laser cutters. For these users the stacked traces would result in some lines being cut and re-cut up to eight times. In the previous example the circular outline of the FCM logo is repeated across most of the paths. Not only is this inefficient, but it could also result in damage to the work being produced, or even to the machine itself. By un-checking the “Stack Scans” option, you will end up with a series of paths that fit inside each other without overlapping.



There's no hiding of lower paths by those further up the z-index. In fact you could re-order the z-indexes and still get the same visual result. The circular outline of our logo now appears only twice – once as an interior path of the background trace, and once as the outside of the large orange path.

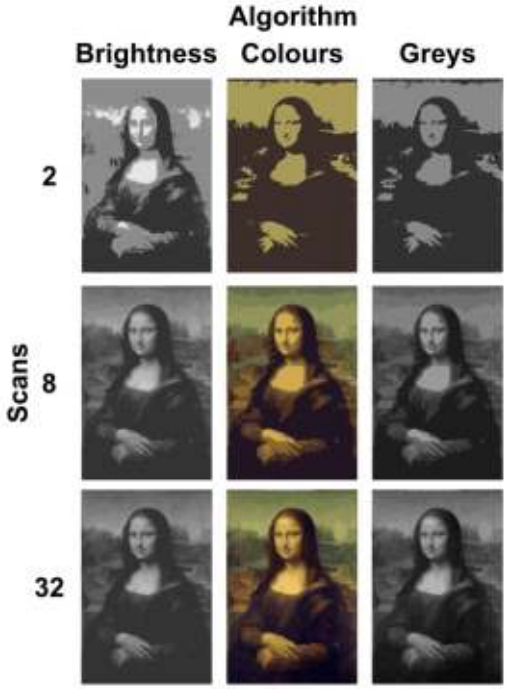
For some tasks – even those that don't involve blades or lasers – a trace made in this way can be easier to edit than one made with “Stack Scans” enabled. As always, the only way to know for sure is to try both approaches and see which works best with your particular source image. Be aware, however, that disabling this option can leave small gaps, blobs and other artefacts between the traced paths. A close up of one of the white circles in the FCM logo reveals this issue all too clearly (the gray part is the background color showing through gaps between the paths).

The last of the three checkboxes, “Smooth”, specifies whether the bitmap image is traced in its original form, or after slightly blurring it. The purpose of blurring it is to smooth out some of the minor color changes that often appear in an image, usually resulting in a trace that is less broken up and has fewer nodes. Because the Potrace algorithm produces only monochrome paths, the resultant trace will still be crisp and sharp, even with this option enabled. Usually I find it's better to leave this checked and benefit from simpler paths, but, if you want to preserve the fidelity of the original image as much as possible, then you may prefer to uncheck it.

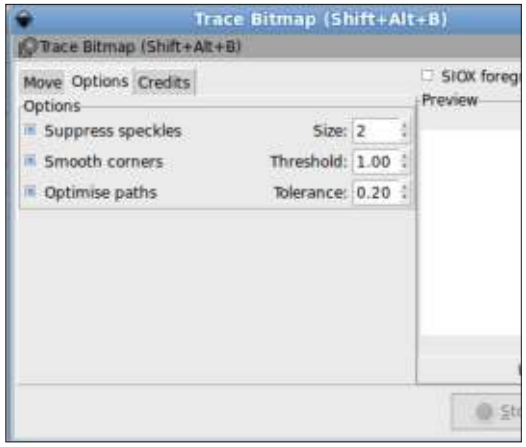
Now that I've explained the options, it's time to take a look at the other two algorithms. All the examples so far have used the “Colours” algorithm. The “Greys” algorithm is a trivial one to explain: it's exactly the same as the “Colours” algorithm except that the resultant paths are converted to shades of gray.

The “Brightness steps” algorithm separates the bitmap image based on the brightness of each pixel, and produces a group of grayscale paths. The number of paths created is always one more than the “Scans” value, giving you a minimum of three paths. I've found this algorithm to be more

problematic than the other two: on my machine it will create a preview for the FCM logo, but won't actually trace it at all! Fortunately it seems happy to trace renaissance paintings, so I've used an image of the Mona Lisa to create a sample grid of the same image traced using each of the three algorithms, with three different values for the "Scans" spinbox. I've only gone as far as 32 scans here – larger values will give more accurate traces but can take a long time to complete and result in a lot of paths and nodes to deal with.



There are some other options in the Trace Bitmap dialog that apply to all the tracing algorithms, whether they are of the Single Scan or Multiple Scan variety. These are gathered together into their own tab in the dialog.



Each can be enabled or disabled using the checkboxes to the left, and each also takes a single parameter which is set in the corresponding spinbox to the right. I tend to leave these all enabled with the default settings, however you may wish to disable or edit them for particular traces.

"Suppress speckles" removes all paths with a size that is less than the number specified. Turning this off can lead to some very long tracing times. Setting it to a higher value can speed up tracing, but lose fine detail in the process.

"Smooth corners" creates paths with rounded corners. Disabling this, or setting the threshold to 0 results in sharp corners instead. The threshold value determines the amount of rounding that is allowed. The effect of this option is most visible on images with sharp corners – if you want to trace a pixel image or a QR code and preserve the individual squares then you may want to disable this option.

"Optimise paths" allows Inksape to reduce the number of nodes by simplifying the traced paths in a similar manner to the Ctrl-L shortcut. The resultant paths aren't a precise match for the original, but are usually close enough for artistic purposes. The Tolerance spinbox sets the amount of variation from the original path that is allowed, with higher values allowing more variation and resulting in fewer nodes.

There's one final option in the Trace Bitmap dialog that sits on its own in the top-right corner: "SIOX foreground selection". This uses the Simple Interactive Object eXtraction algorithm (http://www.siox.org/) to separate a foreground object from the background prior to tracing. With this enabled, the dialog requires two objects to be selected: the image to be traced, and a filled path that roughly covers the foreground object to be extracted. You can see the effect with this

trace of a picture of Horatio Nelson's statue on top of his eponymous column in London.

Although this mode might appear useful, in practice it can be difficult to get a good result. The previous example only worked well when I reduced the original high-resolution image down to something with fewer pixels. In my experience, you would be better off separating out the foreground using The GIMP (which has its own, more interactive, implementation of the SIOX algorithm) and then tracing the result.

Automatic tracing of images is no magic bullet. It's not a practical way to convert a bitmap to an infinitely scalable vector unless you're also prepared to accept a loss of color depth and fine detail. And whilst it can be useful for tracing logos and line art, the resultant paths are likely to still require additional editing. Often the time spent trying to tidy up the results of an automatic trace would have been better spent manually tracing instead: it's better to manually trace a round logo as a circle than to automatically trace it and end up with an approximately circular path, for example. Don't rule out the possibility of mixing both methods, though. As always, you really need to experiment with your own images in order to decide which approach works best for you.

Image Credits

"La Gioconda" (aka "Mona Lisa") by Leonardo da Vinci
http://en.wikipedia.org/wiki/File:Mona_Lisa,_by_Leonardo_da_Vinci,_from_C2RMF_retouched.jpg

Statue of Admiral Nelson, at the top of Nelson's Column, London.
http://commons.wikimedia.org/wiki/File:Admiral_Horatio_Nelson,_Nelson%27s_Column,_Trafalgar_Square,_London.JPG

Mark's Inkscape created webcomic, 'Monsters, Inked' is now available to buy as a book from http://www.peppertop.com/shop/

LibreOffice Special Editions:

http://fullcirclemagazine.org/libreoffice-special-edition-volume-one/

http://fullcirclemagazine.org/libreoffice-special-edition-volume-two/

http://fullcirclemagazine.org/libreoffice-special-edition-volume-three/

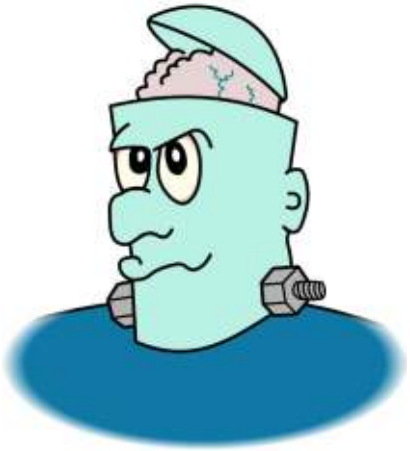
Inkscape Special Editions:

http://fullcirclemagazine.org/inkscape-special-edition-volume-one/

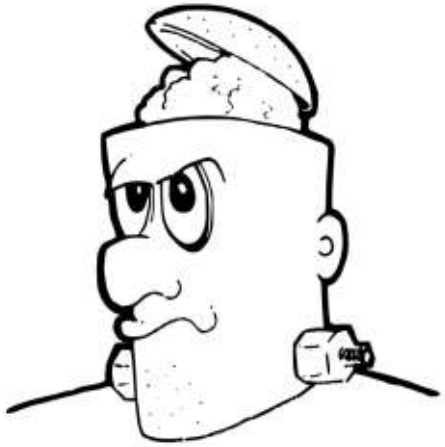
http://fullcirclemagazine.org/inkscape-special-edition-volume-two/

Inkscape - Part 21

Over the past five instalments, we've used Inkscape to trace bitmap images, both manually and automatically. In part 17 I introduced a sketch of a cartoon character, "Frankie", and proceeded to demonstrate a few ways to manually trace it. The first method was simply to draw over the outlines using the normal Bézier Path tool, which resulted in a drawing like this:



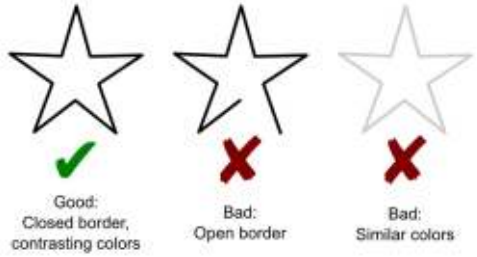
Seeking some variation in the outlines, I then proceeded to introduce a number of ways to trace the sketch, but they all suffered from one obvious omission: color. Although they introduced some dynamics to the image, they all resulted in the outline being rendered as a single filled path. The best looking result arguably came from automatically tracing an inked and cleaned-up version of the image (see part 19 for details), but that still produced only a nice looking outline consisting of a single complex path with over 1000 nodes.



Changing the fill on this image would just result in a colored outline. In order to color the drawing itself, it's necessary to create each area of color as a separate path that can be placed below the outline. In other words you have to manually re-draw each area of the image to create a set of paths that can be filled. If it sounds like a lot of extra work, that's because it is, but Inkscape does have a Paint Bucket tool that can help.

The Paint Bucket – sometimes referred to as a Fill or Flood Fill tool – is commonly found in bitmap editors such as The GIMP, so you may already be familiar with its basic operation. In Inkscape, it's present in the main tool palette and can be selected using the Shift-F7 keyboard shortcut, by pressing the "u" key on the keyboard (easier to remember if you imagine it as a sort of bucket shape), or by clicking on the icon.

Once selected, clicking the mouse inside a "bounded region" in your drawing will create a path that fills it with the currently selected fill and stroke. In simple terms, a bounded region just means an area of a single color that is completely surrounded by a differently colored border, with no breaks. In practice the color of the border must be significantly different from the area you are filling.



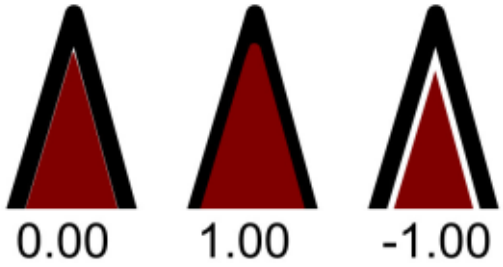
The region to be filled is actually calculated based on the color of each pixel in the area, so it's a bitmap operation rather than a vector one. The pixel you click on is taken as the starting point, then the algorithm tries to expand outwards by considering adjacent pixels. If a pixel's color is close to that of the starting pixel, then it gets added to the fill region and the algorithm continues by considering the adjacent pixels of the newly enlarged region. If the pixel's color is significantly different from that of the starting pixel (i.e. it's the boundary color), then it's not added to the fill region and the process stops trying to grow in that direction. This is repeated until the fill region can't grow any further because it has hit

the boundary on all sides. Finally Inkscape creates a path that surrounds most of the pixels in the fill region, converting the bitmap-based search into a vector result.

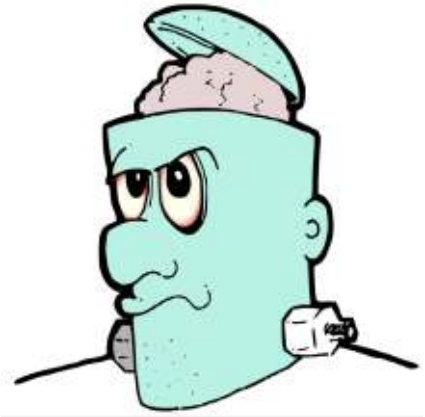
If all that talk of algorithms has got you confused, a simpler way to imagine it is as though you're trying to pour ink into a shallow dish. The ink will spread out along the base of the dish – but only where the base is flat and even enough – and stop when it reaches the edges. Similarly the bucket-fill algorithm tries to spread the fill region out – where the colors are similar enough – and stops when it reaches the contrasting boundary.

Taking a closed star as our object to be filled, zooming it to fill the screen, then clicking anywhere in the white interior, will produce something like this:

In this case the tool was set to a dark red fill and no stroke, and we've managed to create a new path which approximately fills the outline. It's only approximate because the new path doesn't actually reach the edges and corners of the star, leaving a thin gap that shows up when zooming in within Inkscape. This is a common problem with the bucket fill tool, but it can be alleviated to some extent by setting the "Grow/shrink by" option on the tool control bar to a positive number. This will cause the calculated path to grow outwards so that it overlaps the boundary a little. Set it too high and it will extend beyond the boundary, so a little trial and error is often needed. You can also use a negative number in order to shrink the path away from the boundary if you want to. Here's a close-up of the top point of the star with different grow/shrink values.

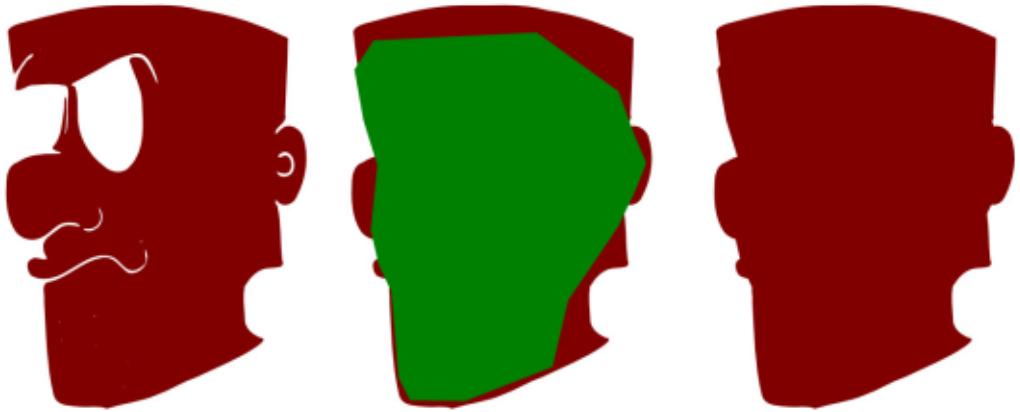


As our aim is to color a comic sketch, it's the middle option – a positive grow/shrink value – that interests us at this stage. As you can see, the new path extends well into the border but we can send it to the back of the z-index, bring the outline to the front, or draw our color fills on a lower layer, to give us our original outline with the appearance of a filled interior. It's a quick and easy way to fill some of the larger areas of a character like Frankie.

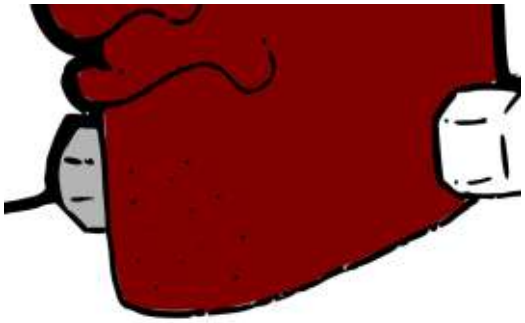


It's not a bad start, but there are a few problems. Some of the fills don't quite get into all the corners, or leave gaps near the outlines, and the bucket fill didn't work at all on the jumper and the nut on the right hand side. Let's start by looking at the fill for the face in isolation, temporarily changing the color so it stands out a little more.

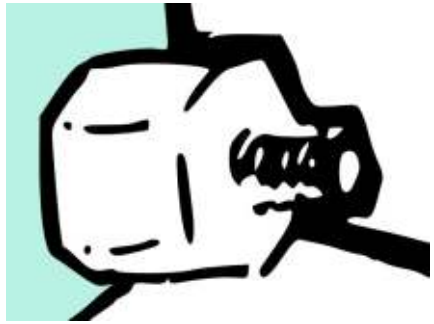
There are 96 nodes in this path, but many of them aren't really necessary. We don't need it to follow the outline of the mouth and nose, and we certainly don't need to trace every bit of stubble. Even the eyes aren't necessary as we can simply stack their own bucket-filled paths on top of the face path. With a bit of node editing it's easy to simplify this complex path. Adding and subtracting rectangles, ellipses or other paths using the Boolean operations is a quick way to deal with lots of nodes at once. In this case I'll simply draw an approximate path using the Bézier tool (shown in green), then use Path > Union to combine it with the face.



Putting this path back into place shows that there are still some gaps around the edges. We can correct this just by dragging a few nodes and handles into position – with the path now significantly simplified this is a much easier job than it would have been previously.



The bucket tool often has a problem with concave corners. One way to reduce the issue is to zoom in on the object you're filling. This results in more pixels being used in the flood algorithm, giving a more accurate fill. Unfortunately it only works for small objects because the whole, unbroken boundary has to be visible in the Inkscape drawing window, or at least only slightly outside it. Otherwise you can fill at a low zoom level, then get in closer and fill again to finish off the corners. If the first fill is still selected you can hold Shift to cause the second fill to be added to the same path, or you can simply create them as two paths then use Path > Union to merge them into one. Most commonly, though, it's easiest just to do a little manual node editing to get the path to fill the corners.

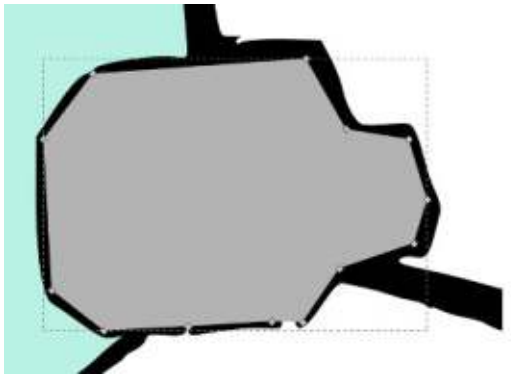


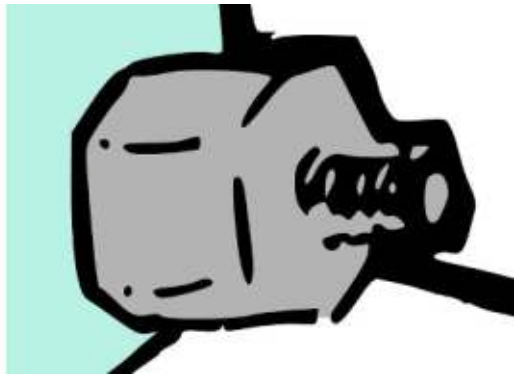
Taking a closer look at the nut and bolt, it's clear that the problem here is a gap in the sketch which means we don't have a completely bounded region to fill.

If your boundary has small gaps in it, then don't despair: the bucket fill tool has a “Close gaps” option on the tool control bar that allows it to automatically deal with such issues. This feature can be set to close small, medium or large gaps, or turned off entirely. Be aware that closing gaps might also prevent it filling legitimate parts of the drawing if you have a complex boundary that pinches together in some areas. Even the “large” setting only closes gaps of a few pixels in size, so you may find it works only if you zoom out a bit – which in turn gives you less accuracy in the corners.

An alternative approach is to manually close the gaps. Remember that the fill algorithm is concerned only with how different each pixel's color is from the initial start point. That allows you to use a contrasting color to draw lines or other objects to plug any gaps before filling. I usually draw these plugs in a color that also contrasts with the boundary so that they stand out afterwards and are easy to find and remove. In this case I've drawn a red triangle using the Bézier tool as my plug: it takes only a few clicks to create, but the size and color makes it obvious that it's an object to remove once it's served its purpose. The color is different enough from the white background that the algorithm will just consider it to be part of the boundary.

Manually plugging gaps lets you use the bucket fill while still being zoomed in close enough for it to get into the corners. In this case, however, there are enough separate areas that some extra manual work would be needed to color it all anyway, so it's just as easy to manually fill the whole shape. By using the Bézier tool to



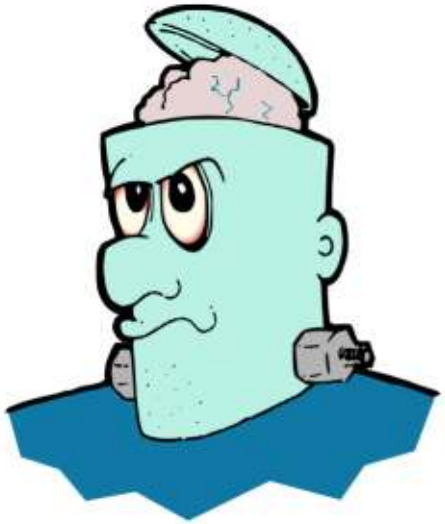


lay down straight lines that follow the center of the outline it's quite simple to color a small region such as this. Once the Bézier path has been drawn, sending it to the back will hide the straight edges behind the outline to retain the hand drawn final appearance.

The bucket fill tool also has a few other tricks up its sleeve. Because it works on pixel values, it can be used to fill areas of a bitmap image, even without tracing it first. In that case the background color can often be less than uniform due to artifacts introduced with JPEG compression, so the bucket fill has a Threshold setting on the tool control bar to adjust the amount that a pixel can deviate from the initial starting color for it to still be considered as part of the background. Adjusting this setting lets you fine tune the fill to only very specific colors, or broaden it to allow quite a range. As well as filling bitmaps, this can be handy when you want to fill an area that has a gradient or other color variations.

You can also change the basic rule of the algorithm entirely using the "Fill by" pop-up menu. Instead of looking for general changes in pixels' colors you can choose to focus on only the red, green or blue components, the hue, saturation or lightness, or even the alpha channel. These options are rarely used, but could be invaluable when your background and boundary aren't distinct or contrasting enough for the standard algorithm to notice the difference between them.

After another manually drawn path for his jumper, and a little node work to remove and re-create the veins in his brain, it's finally time to reveal the finished version of Frankie. Having to separately color each section of a sketch can certainly be time consuming, but if you compare this version with the simple manual trace presented at the start, I hope you'll agree that the extra effort required to maintain some variation in the outline has been worth it. Whilst the manual trace has a decidedly vector feel to the image, this final version retains a lot more of the hand-drawn style.



Mark's Inkscape created webcomic, 'Monsters, Inked' is now available to buy as a book from http://www.peppertop.com/shop/

ubuntu uk podcast

The Ubuntu Podcast covers all the latest news and issues facing Ubuntu Linux users and Free Software fans in general. The show appeals to the newest user and the oldest coder. Our discussions cover the development of Ubuntu but aren't overly technical. We are lucky enough to have some great guests on the show, telling us first hand about the latest exciting developments they are working on, in a way that we can all understand! We also talk about the Ubuntu community and what it gets up to.

The show is presented by members of the UK's Ubuntu Linux community. Because it is covered by the Ubuntu Code of Conduct it is suitable for all.

The show is broadcast live every fortnight on a Tuesday evening (British time) and is available for download the following day.

podcast.ubuntu-uk.org

HOW TO CONTRIBUTE

FULL CIRCLE NEEDS YOU!

A magazine isn't a magazine without articles and Full Circle is no exception. We need your opinions, desktops, stories, how-to's, reviews, and anything else you want to tell your fellow *buntu users.
Send your articles to: articles@fullcirclemagazine.org

We are always looking for new articles to include in Full Circle. For help and advice please see the Official Full Circle Style Guide: http://url.fullcirclemagazine.org/75d471

Send your comments or Linux experiences to: letters@fullcirclemagazine.org
Hardware/software reviews should be sent to: reviews@fullcirclemagazine.org
Questions for Q&A should go to: questions@fullcirclemagazine.org
Desktop screens should be emailed to: misc@fullcirclemagazine.org
... or you can visit our site via: fullcirclemagazine.org

FCM#116

Deadline:
Sunday 11th Dec 2016.
Release:
Friday 30th Dec 2016.

Full Circle Team

Editor - Ronnie Tucker
ronnie@fullcirclemagazine.org

Webmaster - Lucas Westermann
admin@fullcirclemagazine.org

Special Editions - Jonathan Hoskin

Editing & Proofreading
Mike Kennedy, Gord Campbell, Robert Orsino, Josh Hertel, Bert Jerred, Jim Dyer and Emily Gonyer

Our thanks go to Canonical, the many translation teams around the world and Thorsten Wilms for the FCM logo.

For the Full Circle Weekly News:

You can keep up to date with the Weekly News using the RSS feed: http://fullcirclemagazine.org/feed/podcast

Or, if your out and about, you can get the Weekly News via Stitcher Radio (Android/iOS/web):
http://www.stitcher.com/s?fid=85347&refid=stpr

and via TuneIn at: http://tunein.com/radio/Full-Circle-Weekly-News-p855064/

Getting Full Circle Magazine:

EPUB Format - Most editions have a link to the epub file on that issues download page. If you have any problems with the epub file, email: mobile@fullcirclemagazine.org

Issuu - You can read Full Circle online via Issuu: http://issuu.com/fullcirclemagazine. Please share and rate FCM as it helps to spread the word about FCM and Ubuntu.

Magzster - You can also read Full Circle online via Magzster: http://www.magzter.com/publishers/Full-Circle. Please share and rate FCM as it helps to spread the word about FCM and Ubuntu Linux.