



# Full Circle

THE INDEPENDENT MAGAZINE FOR THE UBUNTU LINUX COMMUNITY

INKSCAPE SERIES SPECIAL EDITION



INKSCAPE SERIES  
SPECIAL EDITION



## INKSCAPE

Volume Five    Parts 29 - 35

Full Circle Magazine is neither affiliated, with nor endorsed by, Canonical Ltd.



# HOW-TO

Written by Mark Crutch

Last time I introduced Inkscape's notion of clones – duplicate objects that maintain a link to their parent. I also demonstrated that clones can be independently transformed whilst still maintaining that link, so you can set the basic shape, fill and stroke on a parent object, but then additionally skew, scale and rotate the clone. Keep that capability in your mind, as we'll be returning to it later in this article, but first I need to talk to you about cloning groups.

It's possible to clone almost any type of object in Inkscape. Previously I used rectangles, text and images, but the same rules apply to stars, spirals and paths. The exception is 3D boxes, which don't behave at all well when cloned, and tend to disappear when the parent is modified. You can 'ungroup' a 3D box to convert it into normal paths; if you then group those together again, you can clone that group, but you'll have lost the ability to edit the parent using the 3D Box tool.

Whether created from a 3D box, or via any other mechanism, groups are a prime target for cloning. Having drawn a complex object made up of several different parts, it's useful to be able to clone it in its entirety, rather than having to clone each part separately. Let's use this technique to create a clone army using the snowman we last saw in part 14.



The parent object here is the snowman at the front, which has been cloned multiple times, and the clones scaled. The parent is a group which contains other groups – one for the hat, one for each arm, and so on. It's only when you drill down a couple of levels deep that you finally get to real paths and shapes, but cloning a group

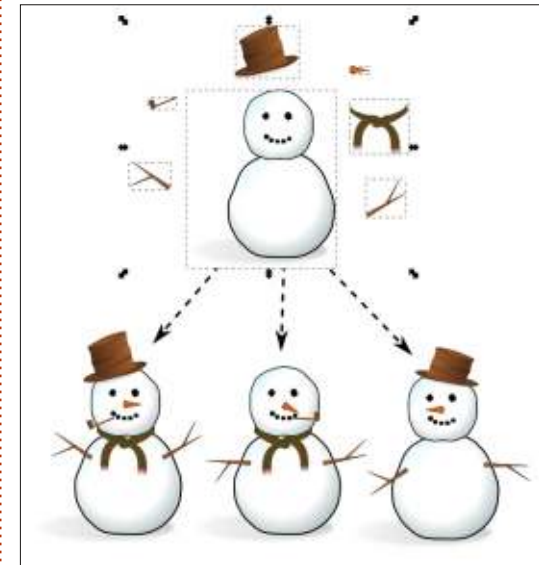
automatically includes all of that structure, no matter how deep it goes.

Creating lots of clones like this can be time consuming, but there are a couple of tricks to speed up the process. The first is to use Inkscape's Tiled Clones feature (Edit > Clone > Create Tiled Clones...) which is a hugely powerful, but extremely complicated, tool. I'll be covering some aspects of it later in this series. The other approach – and the one I took here – is to create the first clone, then drag it around the screen, 'stamping' it in place by pressing the spacebar. Each copy that you stamp is a duplicate of the object you're dragging, and, as we found out last time, a duplicate of a clone retains a link to the original parent object. By working from the back to the front, pausing occasionally to re-scale the clone that's being dragged, it took only a few moments to create all the clones in the image.

One big problem with clones is that they can appear too similar.

## Inkscape - Part 29

Our clone army loses some menace through all being exactly identical, right down to their arm positions. One way to deal with this is to break your group apart into smaller sections, and clone them separately. For example, if we break the snowman apart to separate his hat, arms, pipe, scarf and nose from the main group, we can create an army with a little more individuality by simply omitting or transforming them on some of the characters.



This technique is one that I use frequently when creating comic

strips. A character's body is often cloned directly from one panel to the next, but the arms or legs are cloned separately so they can 'move' between scenes to add a little dynamism to the strip. Often I'll also scale and crop the clones, to give the impression of a camera zooming into or out of the scene. Don't forget that you can still draw extra elements on top of your clones to truly make each one individual. That's how I deal with a cloned character that may be speaking in one panel and silent in another: the original parent has no mouth, then it's drawn on the clones separately for each scene.

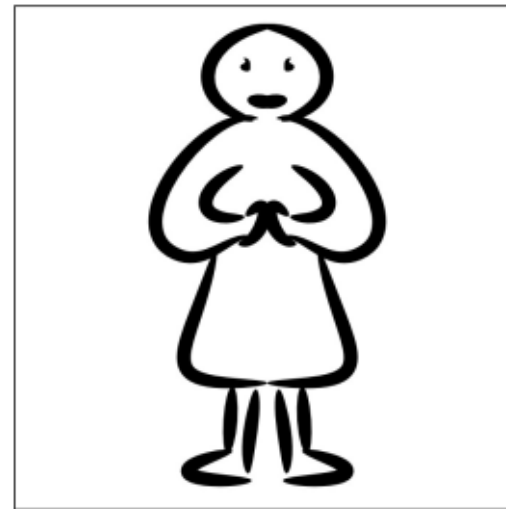
When cloning a group, it's important to realise that the clone is not a group itself. You can't enter the clone to make changes. You can, of course, still enter the parent and make changes there. They will propagate to the clones as usual. When dealing with groups, though, you not only have the option of changing fills, strokes, filters and transformations, but also the possibility to delete objects in the group, or create new ones. Even those changes will propagate to the clones, offering up one rather nice way to use clones which isn't

immediately obvious.

For this technique, we'll need a group. Ideally it would be empty, but Inkscape won't let you create an empty group, so we'll have to put something in it. A good starting point is to draw a large square, almost filling the canvas, with a stroke but no fill. Then immediately group it using the icon on the main toolbar, the Object > Group menu, or by pressing CTRL-G. Select the square and check the status bar: it should confirm that you've got a group of one object.

With the group selected, press ALT-D to clone it. With the clone now selected, press the 'H' key, select the Object > Flip Horizontal menu, or use the toolbar button to mirror the clone horizontally. You shouldn't see any obvious change as you've mirrored it directly on top of the parent. Next, send the clone to the back of the Z-order using the toolbar button, Object > Lower to Bottom, or by pressing the END key. Finally, double-click on the parent (remember, it's now at the top of the Z-order, so you can just double-click on its stroke) to enter the original group. Now switch to the Pencil or Calligraphy tool and draw something.

If you've set everything up correctly, you should find that each time you release the mouse button, the line you've drawn is immediately reproduced as a mirror image at the opposite side of the canvas. All that's actually happening, of course, is that the objects you're adding to your group are being reflected (in every sense) in the clone of the group. Because the square you drew has no fill, the clone shows through as a mirroring of your every stroke.



You're not limited to the Pencil and Calligraphy tools, of course. Anything you draw, regardless of the tool (except the problematic 3D boxes, of course) will be mirrored, making it a useful way of

making symmetric designs. It's surprising how readily a few random paths will become a person, alien, insect or plant once you introduce a little symmetry, making it a great way to get started when inspiration has left you.

As soon as you have drawn another object in the group, there's no need for the square any more, so delete it if you wish. I prefer to leave it in place for reference until I've finished drawing, then remove it as a final step. Either way, don't feel constrained by it – the square is only there to provide some initial content for the group, so don't be shy about drawing beyond its limits.





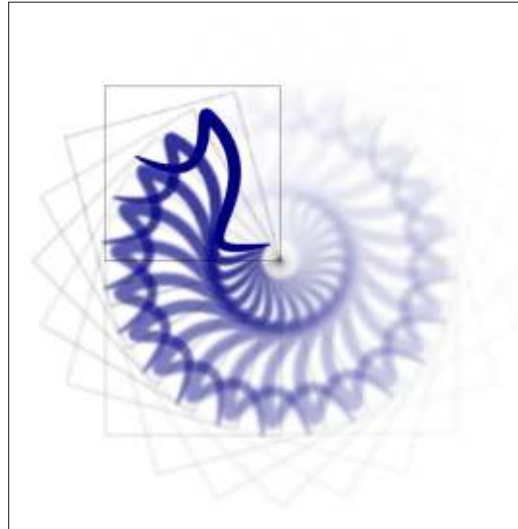
# HOWTO - INKSCAPE

Once you've got the hang of the basic technique, it shouldn't be hard to see that extending it to three clones lets you create drawings that are mirrored both horizontally and vertically.

Flipping the clones like this is a good start, but why not try other transformations on them? Rotating them is a great option and holding down CTRL to lock the rotation to the fixed steps defined in Inkscape's preferences can quickly produce kaleidoscopic effects. Try setting different opacities for each group, or blurring some of them. Within a few minutes, you'll be creating works of abstract computer art with just a couple of squiggles of the mouse.

This example was made by rotating the original square about its bottom right corner. By moving the center of rotation for the original, it was already in the correct place as each clone was created. Then I simply cloned the original group and rotated it whilst holding CTRL. I reduced the opacity and increased the blur a little. Pressing CTRL-D created the next clone, and the cycle was repeated until I had completed a full circle of

clones. Bringing the original to the front, double clicking on it, then drawing a single shape with the Pencil tool (with the Shape control set to "Ellipse") produced this abstract image, spiralling away to nothingness.



If you do try this technique and decide to use blurring on your clones as I have done here, you'll soon find that Inkscape can slow down to a crawl. It's not just blurring that has this effect – any of Inkscape's filter functions can result in the rendering engine having to perform copious calculations, slowing it down considerably. If you find this to be a problem, you can try turning off the display of filters using the View > Display Mode > No Filters option.

Any changes you make to filtered elements will still be stored, saved and applied if you export a bitmap – they just won't show up on screen. Use View > Display Mode > Normal to see the page in its fully rendered glory. You can press CTRL+5 (on the numeric keypad) to cycle through the view modes, including the "outline" mode which is great for finding rogue elements that have become invisible due to too much blurring or too low an opacity. This is a useful shortcut to learn if you find yourself plagued by slow redraws as you can press it at any time – even in the middle of a redraw – if you don't need to see the fully filtered version of the image for the particular edit you're making at the time.

Whether you're creating swirly patterns or armies of characters, there will come a time when you want to break the link between a clone and its parent. Perhaps your snowman needs a completely different smile to its comrades, or your soft, pastel spiral needs a single bright red segment to draw the eye. What you really want is a copy of the parent object that you can modify as much as you want without being constrained by that pesky linkage.

You could, of course, just create a copy as normal, but if you've already got a clone in the right place, it seems a shame not to use it. The Edit > Clone > Unlink Clone menu item will do what you want, turning any selected clone into a plain, old fashioned copy. Use it wisely because although it's easy to convert a clone into a copy, you can't go back in the opposite direction.



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



# HOW-TO

Written by Mark Crutch

Using clones makes it easy to create identical copies of objects or groups in your image. As we saw last time, breaking your groups down into smaller chunks to clone lets you add a little variety, and you can always draw extra objects on top of your clones to further distinguish them. But you can't make significant changes to a clone – altering the shape of a path, for example – without first converting it to a normal copy.

Although you can't make significant changes to clones, they're not entirely inert either. I've already shown how they can be rotated, flipped, scaled and skewed independently of their parent objects. But there's one other little trick in the clones' arsenal which requires a bit of effort to set up, but can be well worth it for some situations: clones can change their fill and stroke separately from their parents.

You can't just take any old clone and give it a new fill and stroke, though. Instead, the parent object has to have its fill and/or stroke

“unset”. The quickest way to unset the fill or stroke is to right-click on the relevant swatch in the status bar at the bottom of the Inkscape window. Towards the bottom of the context menu will be an option for “Unset fill” or “Unset stroke”. Select this item and the corresponding color swatch will be replaced with the word “Unset”. You can also unset the fill or stroke using the “?” button in the relevant tab of the Fill and Stroke dialog.

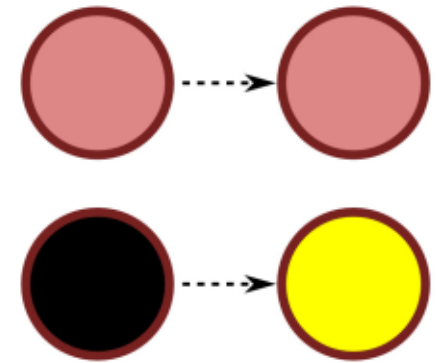


Unfortunately, unsetting the fill

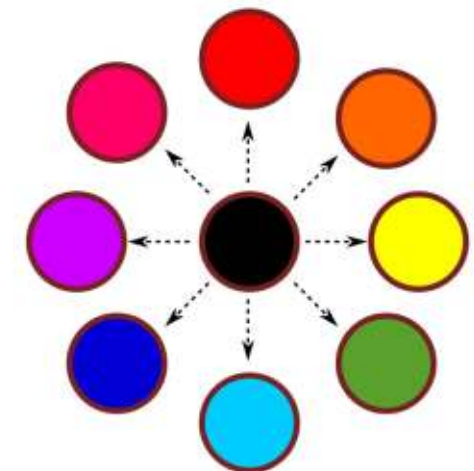
or stroke on your object has side effects. An unset fill is rendered in black which is often bad enough, but an unset stroke doesn't render at all which can be disastrous if the object you want to clone is all stroke and no fill – making it effectively disappear from the screen! Because strokes are trickier to illustrate (an invisible object doesn't make for a great screenshot), we'll start by just considering unset fills, and come back to strokes later in the article.

As a demonstration I've drawn two identical circles, then unset the fill in the bottom one. You can easily spot it because the fill has been drawn as solid black. It's important to note that “black” and “unset” are not the same thing, though, even if they appear that way on screen. Next, I've cloned each circle, then set the fill color for each clone to yellow. As you can see, the top clone ignores the fill that's been set, just like all the clones we looked at previously. The bottom clone, on the other hand, has replaced the black “unset” fill with the color that's been set on

the clone itself.



The yellow clone has inherited its shape, size and stroke from the parent object, but carries its own fill color. We can extend this further by creating additional clones from that one parent and giving each of them their own color.



When dealing with something as simple as a circle, there's probably no real benefit in creating clones like this compared with simply duplicating the parent and changing the fill. But the parent object is rarely as simple as a circle, and these clones can still also be rotated, flipped, scaled and skewed independently. Furthermore, the fills don't have to be simple colors: you can use patterns and linear or radial gradients, too. Admittedly the Inkscape UI struggles a little with anything other than simple colors as the gradient editing handles tend to appear out of place, and the pattern scaling handles don't appear at all – but SVG itself allows for all these possibilities.

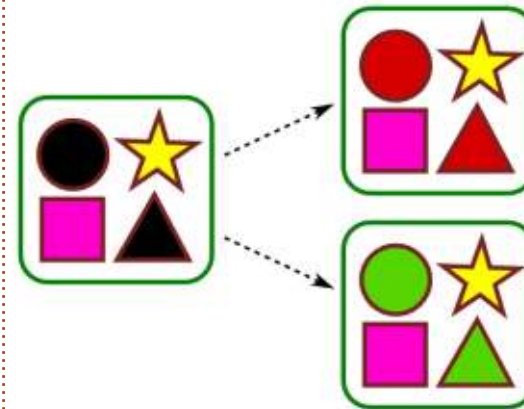


Taking our previous example, let's mix it up a little to demonstrate some of these capabilities. First I've converted the parent circle to a path, then tweaked its nodes to give a more interesting shape. Then I've squashed, skewed and rotated some of the clones, and given others different fills or transparency. They're all still clones – a change to the parent path will affect them all – but combining transformations with an unset fill lets them each take on a distinctly different appearance.

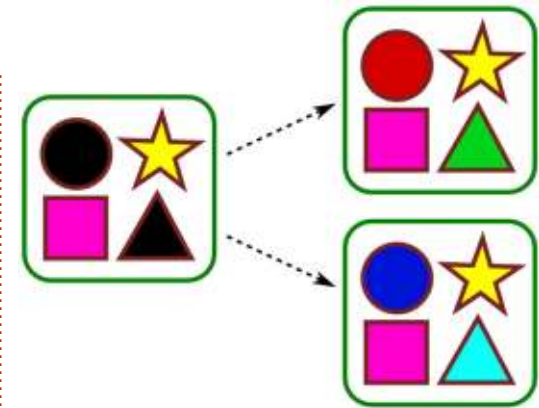
In the previous article, I was extolling the virtues of cloning groups rather than single objects, so you'll be pleased to hear that you can use unset fills in cloned groups as well. Any objects within your group that have their fill unset will be given the clone's color, whereas other objects will be cloned as normal, inheriting their fill from the parent. This ability to mix normal and unset fills within the parent can be very useful if you require a few similar copies with just part of the design changing color with each clone – consider creating some characters for a crowd scene, each of which has a

different colored T-shirt.

In this example, I've cloned a group of five objects – four shapes inside a larger rounded square. The circle and triangle have their fills unset, whereas the star and square have them set to specific colors. You can see that in each clone the shapes with the fixed colors appear the same as the parent, but those with the fill unset use the color that's set on the clone itself.



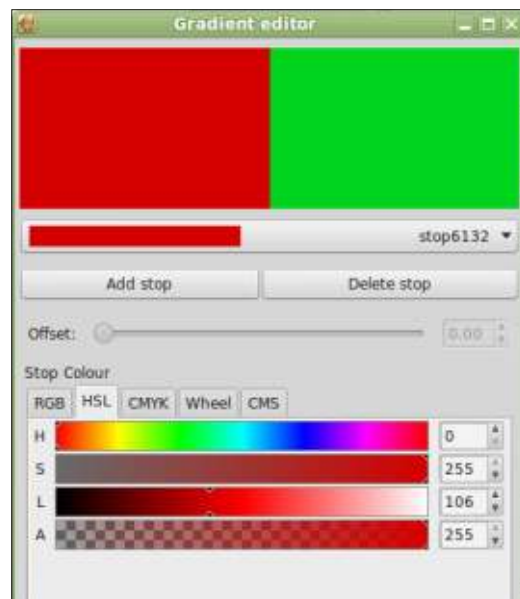
You'll notice that both the circle and the triangle take on the same color in each clone. One major limitation of this approach is that SVG considers all unset fills in a clone to be identical. There's no way to apply two colors to your clone and have one affect the circle whilst the other affects the triangle. Or is there...?



If you really want to get extra colors into your clones, there are ways to do it by being underhanded and devious. One approach is to stack identical clones on top of each other, each set to a different fill color, then use clipping paths to only show the relevant parts of each clone. Another technique I've used in the past is to create a filter in the parent that "rotates" the color of one of the unset objects – more on that when we get onto filters later in this series. For this example, however, I kept it simple: I managed to use two colors in the unset objects by using a linear gradient as the fill. By creating a couple of extra stops in the gradient and setting them to the same colors as the start and end points, I created gradients like this, to allow me to fake the appearance



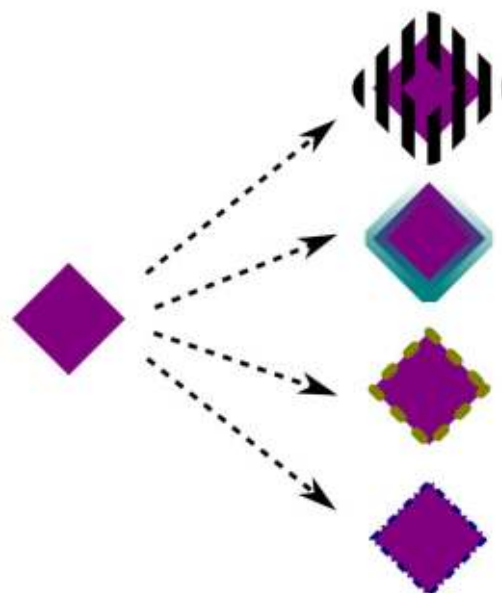
of having two separate fill colors.



At the start of the article I did promise to come back to unset strokes. Whereas an unset fill is rendered in black, making it easy to see and work with, an unset stroke is also rendered in black, but with a width of zero! If your object has a fill – even an unset one – it will still be visible on screen. But, if you unset the stroke of an object with no fill, it will disappear from view entirely. For this reason my first rule of working with unset strokes is to unset them as the last step. Work with a colored stroke while you're creating your parent object or group, and unset it only at the last minute.

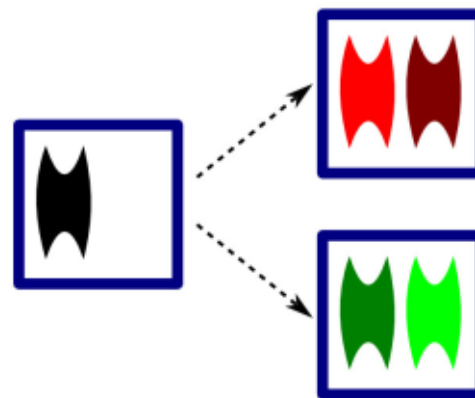
The second rule is not to panic if you do lose track of your object. Just use the View > Display Mode > Outline menu to switch to a mode where even the most invisible of Inkscape objects appears as a skeletal outline. Even in this mode you can still select and manipulate your objects, and they will remain selected when you switch back to another display mode.

With an unset stroke in your parent object or group, you can now set not only the stroke color (or pattern, or gradient) on each clone, but also its width, join style, end caps and dashes. In fact, at a minimum you have to set the color and width if you want the stroke to



be visible. Unfortunately, you can't set markers on a per-clone basis – if you want arrows or chevrons marking the nodes of your path, they have to be set on the parent object.

With unset strokes, you now have another method of getting an extra color into your clones. With a little lateral thinking, you can even usurp the stroke to provide a second fill color, if that suits your needs better. In this final example, I've used an unset fill on the left hand shape, that's clear enough. But where is the right hand shape coming from?



You've probably already guessed that there's an unset stroke involved, but how does that turn into a shaped fill in the clones? The trick was to draw a single vertical line in the parent and give it a really large width value – 40px

in this case – effectively creating a 40px wide rectangle. Then I used a path to clip that “rectangle” to the shape I wanted, before finally unsetting the stroke. On each clone I have to only set the stroke to the color I want, with a 40px width, and my second “filled shape” appears. You can also perform a similar trick using masks, which can be particularly useful for faking a gradient in your clones.

The ability to use different fills and strokes on clones can make them extremely versatile, at the expense of leaving you with black areas or invisible lines in your parent object. Being able to set only two “parameters” on each clone can be limiting, but hopefully you've seen how, with a little lateral thinking, the use of masks, gradients, clipping and filters can let you break that restriction to some degree.



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



# HOW-TO

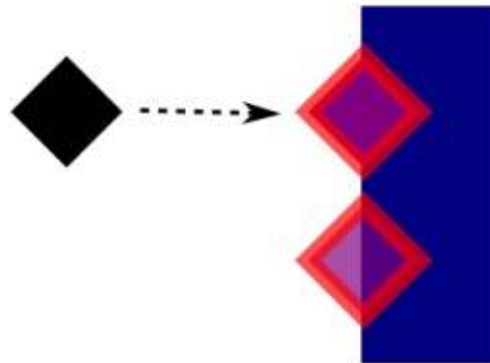
Written by Mark Crutch

## Inkscape - Part 31

After the previous instalment of this series had gone to press, an interesting problem was raised at [www.inkscapeforum.com](http://www.inkscapeforum.com) that directly relates to the use of unset fills and clones. So, before moving on to the next topic, I think it's worth drawing attention to this issue, and how you can deal with it.

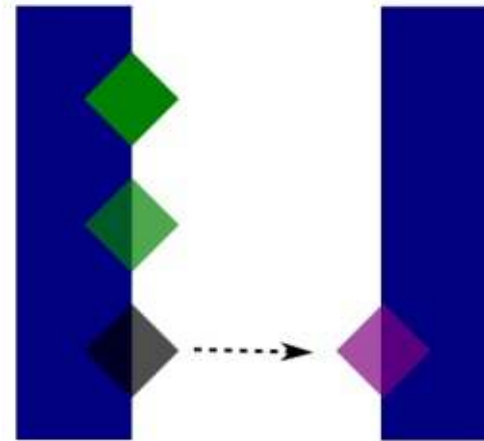
Let us suppose that you create a parent object and unset both its fill and stroke. As you know from the previous instalment, you can now set the fill and stroke on any clones independently. I demonstrated using colors, patterns and gradients for both the fill and the stroke, but it seems that one thing I missed was setting a non-opaque color – i.e. one with the alpha (A) channel in the Fill and Stroke dialog set to something other than 255. It turns out that doing this with the stroke works perfectly well, but the opacity of the fill color is completely ignored. In this example you can see what I mean. Both the fill and stroke opacities on the clone have been set to 177, but only the stroke actually appears transparent (the bottom

diamond shows how the clone should appear).



It turns out that there's a bit of a bug in Inkscape (issue 1183400 in Launchpad). When you unset a fill, the program fails to remove the “fill-opacity” attribute in the SVG. Any clones made from that object are then stuck with the opacity that the parent fill had before it was unset. As a demonstration of this, I created a clone and filled it with an opaque green color. Then I set the alpha channel for the green fill to 177. Next I unset the fill altogether. Finally I cloned the object and gave the clone a fully opaque purple color.

What I would expect to see here is that unsetting the fill should also



unset the opacity, making it default to the SVG standard of fully opaque. Clearly the parent at the bottom is still translucent, as the blue bar behind it shows. Even without the blue bar, it appears as a washed out gray color, rather than the deep black we would usually expect of an unset fill. Furthermore, the clone is now forced to adopt the transparency of the parent, so there's no way that any clones of this object could be completely opaque, regardless of their own alpha value.

For most people this bug may never be a problem, but if you do want to set the opacity of your clones to be anything other than

100%, there is a “fix” for the issue. It will mean using Inkscape's XML Editor dialog, which is a topic I had hoped to avoid until later in this series, but as my hand has been forced, I've decided to introduce it now. But to understand the XML editor, you first need a little insight into the structure of an Inkscape file.

The SVG format that Inkscape natively uses is an XML file, meaning that it follows the rules, conventions and structure for such files as defined by the W3C – the standards body of the web. XML is a dubious abbreviation of “eXtensible Markup Language”. In short, it means that every Inkscape file is made up of a hierarchical collection of “tags” (also called “elements” or “nodes”), each of which can carry “attributes” to further define it. For example a simple rectangle might appear in an SVG document as a “rect” tag, with attributes for defining its size and location:

```
<rect height="300"
width="400" x="50" y="100" />
```



# HOWTO - INKSCAPE

What about the hierarchical aspect I mentioned? How about this more complex example:

```
<svg
xmlns="http://www.w3.org/2000
/svg">

  <g>

    <rect id="r1"
height="300" width="400"
x="50" y="100" fill="red" />

    <rect id="r2"
height="500" width="100"
x="200" y="50" fill="blue" />

  </g>

</svg>
```

As you can see, we've got two rectangles now, and they've gained a couple more attributes to set the fill color, and to give each of them an ID so we can identify them individually. Those are both inside a set of `<g>...</g>` tags, which defines a group in SVG terms. The group, in turn, is inside the outermost pair of `<svg>...</svg>` tags. You can think of these as a clue to an application that the content inside them should be rendered as SVG, rather than as HTML or plain text.

Because the "r1" rectangle is first in the file, it's drawn first on

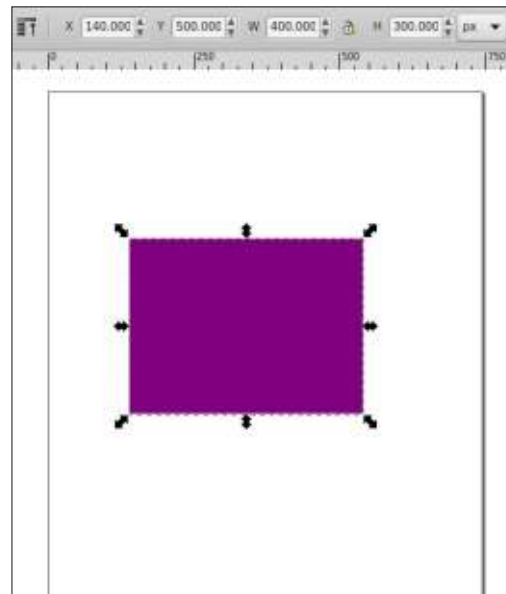
the canvas. The "r2" rectangle is drawn afterwards, so it overlaps the first one. The result is a simple SVG image with a blue rectangle on top of a red one, both inside a group. Try it for yourself: copy the code above into a text editor and save it with an "svg" extension, then load the file into a web browser or Inkscape.

What if we wanted another rectangle, outside the group? We could just include an additional `<rect>` element but place it after the opening `<svg>` tag but before the opening `<g>` tag. That would put it behind the group when the image is rendered. Place it after the closing `</g>` tag, and before the closing `</svg>` tag, and it will appear on top of the group. Give it a try for yourself, but remember to change the position, size and colour of the new rectangle so that it doesn't get obscured by the existing ones. While you're editing the file, how about adding "rx" and "ry" attributes to set the size of the corner radius. Or replace the `<rect>` with a `<circle>`, swapping the dimension and position attributes for "cx", "cy" and "r" to set the center coordinates and the radius.

By now you should be starting

to get a feel for the structure of an SVG document. Of course the ones that Inkscape produces are far more complex, generally including many more elements and attributes, but the basics remain the same. If you want to take a look at some more simple files in your text editor then I recommend the various flag images on Wikipedia, which tend to be pared down and minimised by hand, removing any unnecessary structure or metadata. Examining a few of these will quickly give you some insight into the structure of XML files.

Let's switch back to Inkscape now, and create a very basic drawing – just a single purple rectangle on the canvas.



With your new found knowledge of SVG you should know how to hand-code this in just three lines, yet, when I saved my copy from Inkscape, the resultant file had 62 lines in! Admittedly many of these were due to it putting every attribute onto its own line – an option that can be set in the SVG Output pane of the Inkscape Preferences dialog. Yet, even enabling the "Inline attributes" setting still resulted in 19 lines. What's going on?

Look at an Inkscape SVG file in a text editor and you'll quickly spot a lot of attributes that have a prefix to their names. So rather than `label="Layer 1"` you'll see `inkscape:label="Layer 1"`. This is a feature of XML called "namespaces", and it's basically a mechanism by which one XML file can safely include elements and attributes from other XML languages without having to worry about them clashing. In this case it indicates that the "label" attribute isn't part of the SVG spec, but is rather an attribute from the "inkscape" namespace. This allows Inkscape to include application-specific data in a file, whilst still remaining compatible with the SVG

# HOWTO - INKSCAPE

specification, and therefore with other applications that can read SVG files (though they'll usually ignore the Inkscape-specific additions).

In an Inkscape file, you'll typically see "inkscape" and "sodipodi" namespaces that are used to store application-specific data (Inkscape was created as a fork of an older SVG editor called Sodipodi – which was, itself, a fork of an even older vector graphics program). You'll also see "dc" which stands for Dublin Core, and represents the namespace for a set of defined terms used to contain metadata about the file. You can set these using the File > Document Metadata menu item in Inkscape, and it's recommended to fill out at least some of the fields if you plan to distribute your SVG file online. Because the metadata are stored in a standard form using a well known namespace, it increases the chance that your document could one day be indexed by online search engines.

One final thing to note in the file is that the rectangle itself, although it's pure SVG with no namespaced attributes, is a little different to the ones we created

earlier. Whereas we used the fill="red" syntax to provide a fill color, Inkscape uses a more general purpose "style" attribute to carry numerous details about the color and style of the rectangle. It also uses hexadecimal RGB numbers for the color, rather than a color name – you can force it to use color names where possible in the Inkscape Preferences, but it's usually not worth bothering with unless you have a specific reason to do so: most colors don't have corresponding names so will still be stored as hex numbers, and using names can cause problems with some Inkscape extensions.

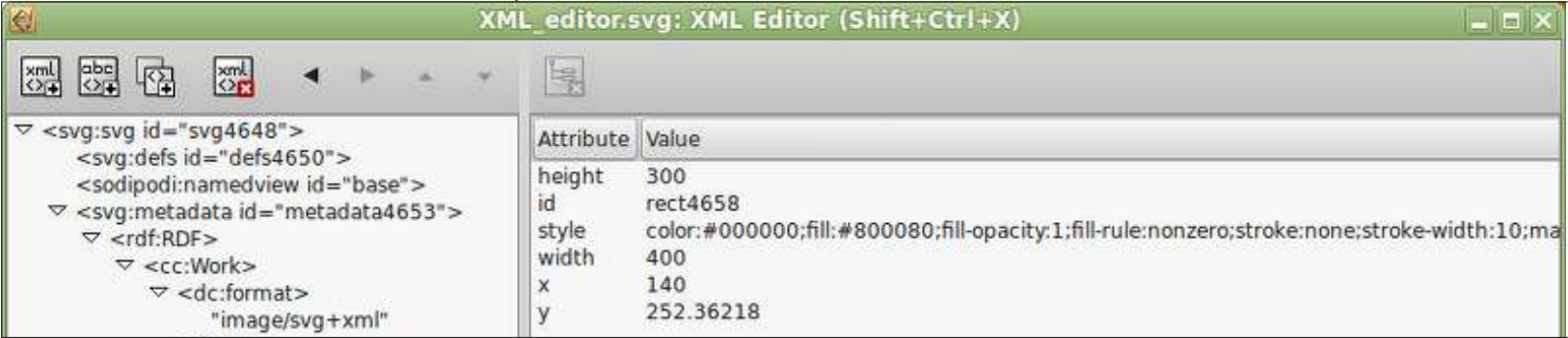
With all that background in place, it's finally time to look at the file in Inkscape's XML editor. You can open this by pressing CTRL-SHIFT-X or by selecting Edit > XML Editor from the menu bar. The dialog is made up primarily of a tree on the left which shows the

structure of the SVG file, and a pane on the right to list and edit the selected item's attributes. The little triangles in the tree can be toggled to show or hide that particular part, and indentation is used to show the hierarchy of the elements. In this screenshot I've expanded all the triangles so that the metadata elements are visible, with their Dublin Core namespace. Despite the closing tags not being explicitly shown, you can nevertheless see that the rect at the bottom is "inside" the group (g) just above it – actually an Inkscape layer, as you can tell from the Inkscape-namespaced "label" attribute. This layer is, in turn, inside the root svg element. One thing to note is that the XML Editor shows the SVG namespace on elements (so you can see svg:svg, svg:g, svg:rect...) even though the exported file just uses the base names (in XML terms the SVG namespace is set as the default for

the document, so it doesn't then need to be explicitly added to every element).

When an entry in the tree is highlighted, its attributes are shown on the right. If a single element or group is selected on the canvas it will be automatically selected in the XML Editor, so you can simply leave the dialog open and click on various objects in your drawing to see their details. Equally, selecting an entry in the tree will also select the corresponding object on the canvas.

Here I have the rectangle selected, but there's something odd going on. If you look back at the image of the rectangle on the canvas you'll see that it has dimensions of 400x300 pixels, and is positioned at x=140, y=500. Now look at the XML Editor image: width, height and x are all correct,



but y claims to be 252.36218 – which is pretty far from 500!

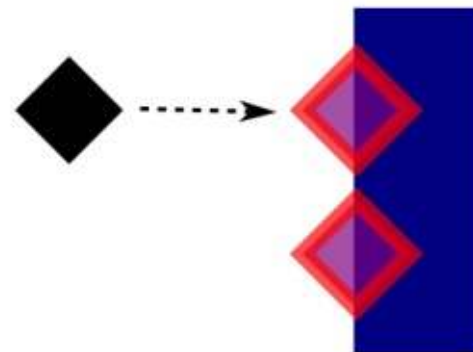
SVG places its origin point at the top left of the document. This sort of makes sense, given that it comes from the world of the web where the height and width of a document can change dramatically, but the top left is always the top left. The x-axis therefore runs from left to right, as you might expect, but the y-axis runs from top to bottom, with positive values moving further down the page. Inkscape, on the other hand, presents a more traditional drawing view, with the origin in the bottom left, and the y-axis running up the page from top to bottom. So the 500 value you see in the main Inkscape window represents the distance from the bottom of the page to the bottom of the rectangle, whereas the value in the XML Editor (and the value that appears in the SVG file) is the distance from the top of the page to the top of the rectangle. Usually this incongruity has little impact, but if you're trying to find specific coordinates in an SVG image you do need to be aware of the difference.

With the rectangle still

selected, let's click on the “style” attribute on the right. The attribute name and value is put into the fields at the bottom of the dialog. In the case of the style attribute, the value is actually a single long string which is, itself, made up of name:value pairs. If you're familiar with CSS from the web world, then you'll recognise the format – if not all of the property names (SVG uses a lot of the standard CSS properties you might know from writing HTML, but adds a few of its own). With the style attribute loaded for editing, we can now address that pesky issue with the fill-opacity and clones.

See the “fill-opacity:1;” section, right near the start? We need to remove that. This is just a multiline text field, so simply click to place the cursor in there, then move around with the arrow keys and edit the text as you would normally. Once your editing is done, you need to click on the “Set” button to make it take effect. Assuming the fill-opacity's value was 1, then you shouldn't notice any change, since 1 in here corresponds to 255 in the Fill and Stroke dialog, and is the default for SVG if it's not specified.

Now clone the rectangle, and try changing the clone's color. You can't, of course, since the parent rectangle's fill is still purple, not unset – but, once you give the clone a fill color, you gain access to the alpha slider in the Fill and Stroke dialog. Reduce that value and you'll see that you can affect the transparency of the fill, if not its color. Select the parent again (SHIFT-D if the clone is still selected) and then unset the fill. Now you can change the clone's fill color and opacity to your heart's content. It's as simple as that: to work around this Inkscape bug, and restore the ability to change a clone's fill opacity independently of its parent, you just have to remove the fill-opacity property from the parent's style attribute. Doing this on my original test image gives exactly the result you would expect.



You may have noticed that I haven't talked about the toolbars in the XML Editor, and that's with good reason. The buttons there give you the ability to significantly change the structure of your SVG file – potentially with disastrous effects if you're not sure what you're doing. By all means have a play around in the XML Editor. Move nodes, un-indent them, change their attributes or remove them altogether. It offers a fascinating insight into the structure of an Inkscape file, and gives you unprecedented power to tweak things that aren't always exposed in the Inkscape user interface. But if you do decide to experiment, please make sure you do it on a temporary file, or one you've got backed up elsewhere.



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





# HOW-TO

Written by Mark Crutch

## Inkscape - Part 32

**Erratum:** During magazine layout, it's common for images to be cropped or dropped entirely. Unfortunately, that happened to a couple of the images in last month's article which may have caused confusion – some of the cropped parts were directly referenced in the text. I've put the full images online at <http://www.peppertop.com/fc/>

Back in part 29, I showed one use for putting a single object into a group: as the start of a method for producing mirrored or kaleidoscopic drawings using clones. This time, I'm going to use a single object in a group, plus a bit of cloning, to perform some sneaky tricks with clipping and masking.



Let's start with a familiar image to which we'll apply a clipping path. Remember, all the parts inside the bright green clipping line will remain visible once the clip is applied, whereas those outside it will be hidden.

Having clipped our image down to just show her head, what if we then decide we also want the hands? We could release the clip and construct a more complex path that includes the hands as well, before re-clipping. But that still leaves us with a single object, with the head and hands at a fixed distance apart. If we want to move the hands independently of the head – or maybe scale or rotate them – we're out of options.

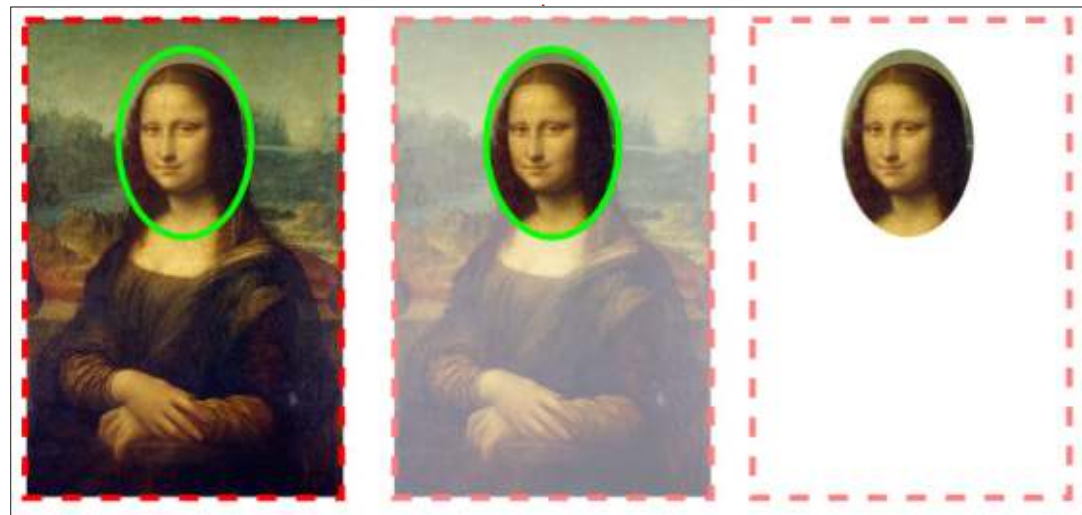
The next obvious approach would be to import the image a

second time and clip it to show the hands. Now we have two images, each clipped differently, resulting in two separate objects that can be modified independently. With a linked image, that might be a reasonable approach, but if our images are being embedded so that the resultant SVG can be shared more easily, we've now got two large bitmap images bloating our file. What we really want is a way to include the image just once, but create two completely separate clips from it.

From the introduction, you've probably already deduced that the answer is to group our image

before clipping it. Here I've denoted the group with a dotted box for illustrative purposes – it doesn't appear in a real drawing.

The result (below right) doesn't look altogether different from our first attempt, but that's because the effect is not a visual one, but rather a structural one. Previously, our clipping path was applied directly to the image. If you were to look in the XML Editor you would see that the image has "clip-path" attribute whose value is the ID of a path stored in the <defs> section of the XML file. With the image grouped and then clipped, however, the clip-path attribute is



now on the group itself and the image either has no clip-path attribute at all, or its value is set to "none".

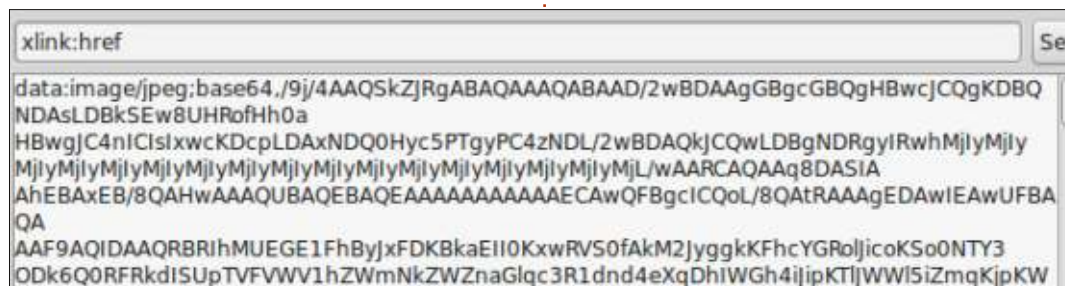
The difference is subtle, but useful. By clipping the group, we've effectively created a window through which we're viewing just a part of the image at any time, but the image itself is still the full size. We can demonstrate this by double-clicking to enter the group, then dragging the image around. We're moving the un-clipped image inside the clipped group, not moving the group itself, and the effect is quite different. You could also scale, rotate or skew the image if you want, all without the size or shape of the "window" being affected.

It's a useful trick in its own right – combined with performing an "object to path" on your clipping path before the clip is applied (see Part 13) the simple expedient of grouping your image before clipping means that you can not only change the clipping path itself without releasing it, but you can also move the focal point of the content within it. Clearly you could move the image like this to bring the hands into view through the

"window", but that still doesn't get us two separate clips. For that we need to get a little devious.

The steps we'll be taking aren't difficult, but they do need to be done in the right order. Once you've done it a few times, it will become second nature to you. To make things clearer, I suggest starting with a new file, and opening the XML Editor so that you can see exactly what's happening at each step in the process.

**Step 1:** Drag and drop your image into the document. I chose to embed the image to really prove the point, but linking works as well. In the XML Editor you should see an `<svg:image>` tag with an "xlink:href" attribute. If you've embedded the image, then the attribute will contain a Base64 encoded version of the image's binary content (if you linked the image it will contain the path to the original image).



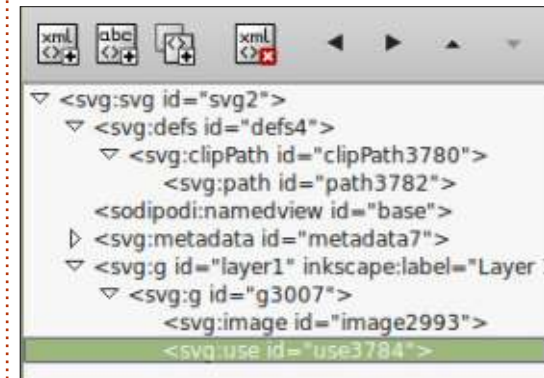
**Step 2:** Group the image. An `<svg:g>` element will appear in the XML Editor – expand it out to see that your image is still inside it.

**Step 3:** Draw your clipping path. You should see it appear in the XML Editor as a sibling of the group.

**Step 4:** Select both the path and the group, then apply the clipping path using either Object > Clip > Set, or by selecting "Set clip" from the right-click context menu. Note the effect in the XML Editor: your path is moved into the `<svg:defs>` section, and the group gains a "clip-path" attribute that references the path by its ID.

**Step 5:** Double-click to enter the group, and select the image. Keep an eye on the status bar to confirm what's happening, and the XML Editor should also highlight the image's entry.

**Step 6:** Clone the image using Edit > Clone > Create Clone, or by pressing ALT-D. Note in the XML Editor that an `<svg:use>` element is created with an "xlink:href" attribute that references the image by its ID. That's your clone! No matter how complex the parent object is, a clone is actually always just a simple little `<svg:use>` element that holds a reference to the original.



**Step 7:** Currently you have an image, and a clone of that image, both inside a single group. Let's put the clone somewhere more useful. With the clone selected, use Edit > Cut or CTRL-X to remove it from the document (watch it vanish from the XML Editor before your very eyes), and put it on the clipboard. Remember, what we've actually put on the clipboard is just a small `<svg:use>` element, not the heavyweight binary data of the



original image.

**Step 8:** Inkscape doesn't really care where we paste the clone, so long as the ID it references still exists in the document. So double-click on the background until the status bar tells you you're out of the group, then use Edit > Paste or CTRL-V to paste the clone into the document.

**Step 9:** If everything went well you should now be looking at another, un-clipped copy of your image. Remember, the clip was applied to the group, but we've taken our clone from a lower level, before the clip has been applied, giving us access to the original image again but without requiring a second copy of all that binary data.



**Step 10:** There's nothing special about this clone – you can treat it as you would any other. That means you can clip, mask, group,

rotate, skew or blur it, and much more besides. So let's complete our original task, and clip it to show just the hands.

There you have it – two different clips of the same image, with just a single copy of the binary data embedded into your document. Of course you don't have to stop at two copies, you can add as many clones as you like. Each one only adds a tiny amount to the document size, but gives you a complete copy of the original image to work with.

Although I've demonstrated this technique with clipping, it works equally well with masking, allowing you to use all the fine control over opacity that masking provides (see Part 14 for more details). You can even mask some clones whilst clipping others to produce something like this information sheet that uses only a single embedded image.

As I mentioned in step 8, Inkscape doesn't really care where you paste the clone. In the examples above I've simply pasted it outside of the original group, but you could also paste it into a different group entirely – even one

that is, itself, clipped or masked. And don't forget that layers are just groups with some extra metadata. There's nothing to stop you cutting a clone to the clipboard (even one that's not in a group), then switching to a different layer before pasting.

Don't think that this technique is limited to bitmaps either. As you know, any Inkscape object or group can be cloned, and equally any can



be put into a group. So you could draw a complex character or scene, group it (let's call that "Group 1"), then group it again ("Group 2"). Clip or mask Group 2 and you can still enter the group, clone Group 1, cut it to the clipboard, and use it elsewhere in your drawing.

I use this technique a lot when creating comic strips. Typically I add some movement to a comic by zooming or panning the scene between frames, but rather than copy or redraw the background and characters I usually use clones that are then scaled as necessary before being clipped to fit in the frame. With this approach, any changes to the originals are automatically propagated to the clones, so I don't need to update multiple panels each time there's a tweak to be made. This strip, for example, really consists of only one panel (the top one), with the background being cloned and clipped to create the second two panels, and the heads added in different poses on top to introduce a little more variety. Finally the text was added in a separate layer to produce the finished comic.







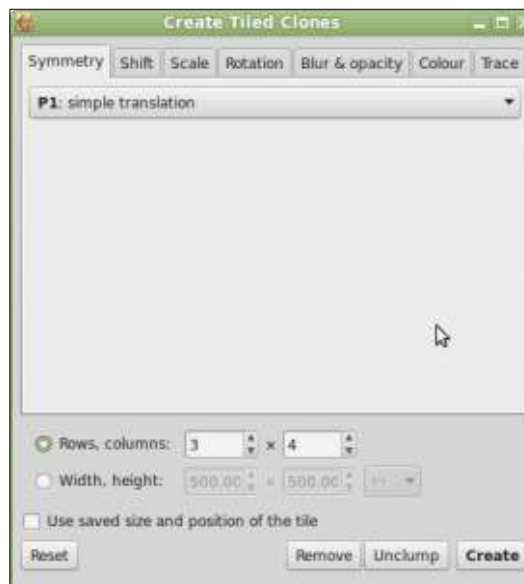
# HOW-TO

Written by Mark Crutch

To conclude our foray into the world of clones, I'm going to spend a few articles describing Inkscape's "Create Tiled Clones..." dialog (which I'll refer to as just the Tiled Clones dialog from here on). This is arguably one of the most powerful, and most confusing, dialogs in the whole application, so take a few moments to prepare yourself before we dive in.

You've already seen the easy way to create a clone: select the parent object and press ALT-D (or use Edit > Clone > Create Clone). If you want a second clone from the same parent, you can either repeat that process, or duplicate the first clone (using CTRL-D this time). Do you want a handful of clones? Drag the first one round the screen, stamping copies by hitting the spacebar from time to time. But what if you want a hundred clones? Or a thousand? And what if you want them precisely positioned? Or you want each clone to be rotated or scaled a little? The Tiled Clones dialog can do all this, plus a lot more.

We'll start by creating an object to use as the parent for our clones. To keep things simple for the time being, I'll use just a coloured square with rounded corners, but the parent can be almost any individual object or a group. 3D boxes, however, don't work with the Tiled Clones dialog – though you can convert them into a group of simple paths first to get the same effect, if you don't mind losing the ability to edit the parent as a 3D box. With a parent object created and selected, open the Tiled Clones dialog via the Edit > Clone > Create Tiled Clones... menu entry.



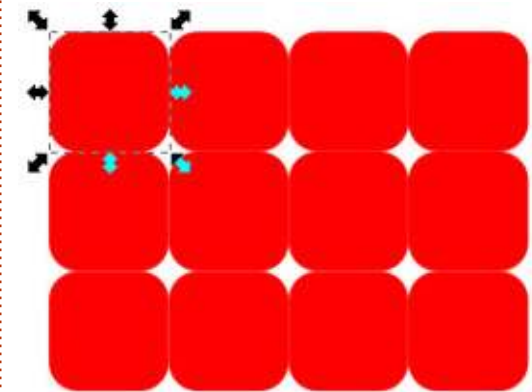
Initially the dialog doesn't look too complex, but if you step through the tabs, you'll quickly see that there are a lot of fields and controls hidden away. The first tab, Symmetry, has only a single pop-up menu, but even that contains 17 different options. For now, we'll just stick with the first one: "P1: simple translation". This will let us create simple rectangular arrays of clones, and is the easiest to understand when first getting to grips with this dialog.

Beneath the tab panels are some global options to define the number of clones you want to create, or the area you want them to cover. Note that I said "cover" rather than "fill". Think about tiling a bathroom wall: you need enough tiles to cover the wall, even if it means some overhang, and the need to cut some to size around the edges. Similarly, Inkscape will create enough tiles to cover the specified area (defined in terms of width and height), leaving you to optionally clip them yourself if they overhang. The "Use saved size and position of the tile" option should

## Inkscape - Part 32

be left unchecked for now – we'll take a look at that in a later article.

To create some tiles, first press the Reset button. This will put the values in all of the tabs back to sensible defaults, so is usually a good starting point. Now enter some values into the "Rows, columns" fields. I'm going to start with a 3x4 array of clones. Finally, with your parent object selected, click the Create button.



There are a few things to observe about the array of clones that has been created. First, notice that your parent object remains selected once the clones are created. This makes it easy to click the Remove button in the dialog to



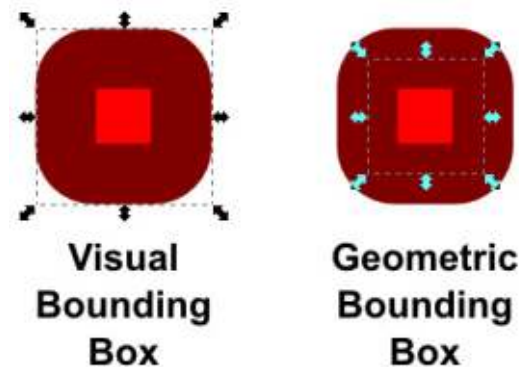
## HOWTO - INKSCAPE

delete all of the clones at once if you're not happy with the result. Be aware that the first clone is placed directly on top of the parent object. If you change the focus and then need to re-select the parent, clicking in the top left corner will actually select the clone. The easy answer is just to select any of the clones, then press SHIFT-D or use Edit > Clone > Select Original.

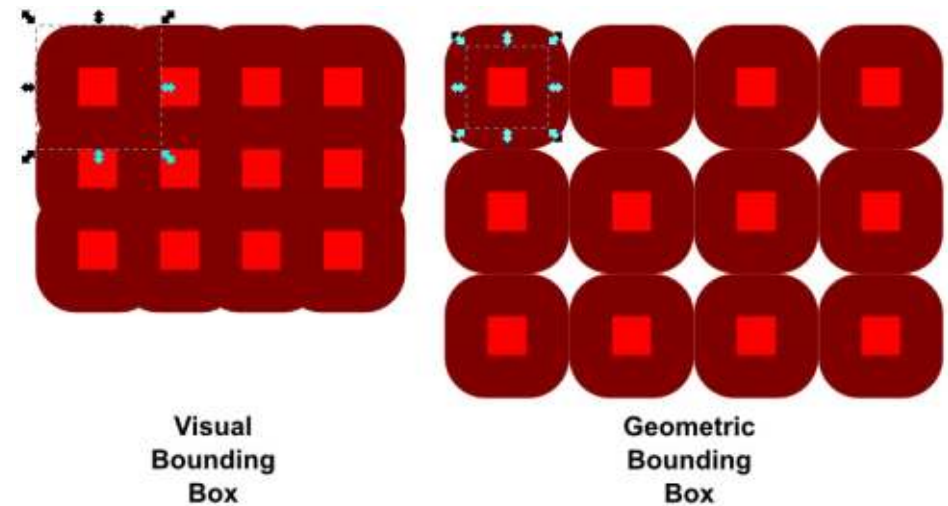
Take a look at the positions of the clones. After resetting the parameters in the dialog, the default behaviour is to create the clones in such a way that they all butt up against one another. When selecting objects in Inkscape, the dashed line that acts as a selection cue is referred to as the "Bounding Box". The dimensions of this box form the basis for positioning the clones. The first column is moved to the right by 100% of the bounding box width. The second column is moved by 200%, and so on. The rows follow identical rules based on the bounding box height instead. This may sound like a slightly abstract way to describe the positioning of the clones, but the Shift and Scale tabs in the dialog use "percentage of bounding box width/height" as

their units of measurement, so it's easiest to think in those terms.

To further confuse matters, Inkscape has two different types of bounding box: the visual bounding box, which includes any stroke that is applied to your object; and the geometric bounding box, which is based purely on the core object – regardless of its stroke. You can choose which one Inkscape uses for selections in the Tools pane of the File > Inkscape Preferences... dialog. The difference between them is clear when an object with a very thick stroke is selected.



According to the documentation, Inkscape always uses the geometric bounding box when creating tiled clones. This doesn't tally with my own experience of Inkscape 0.48 on Linux Mint 17. I've found that

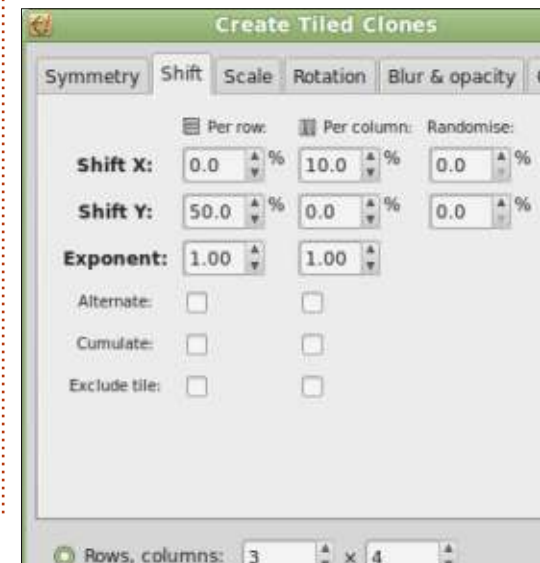


switching the preferences between visual and geometric bounding boxes does have an effect on the tiling. Even more confusingly, the behaviour seems to be the opposite of what you might expect. Look at the same thick-stroked rectangle above when it's tiled using each of the two preferences.

From now on, I'll mostly stick to using shapes with no stroke to demonstrate tiled clones, so this discrepancy won't be an issue. But do bear it in mind if you're trying to tile objects with strokes and the clones aren't appearing in the positions you expect.

The default arrangement of a tightly packed array may be fine if you just want to use this dialog as a

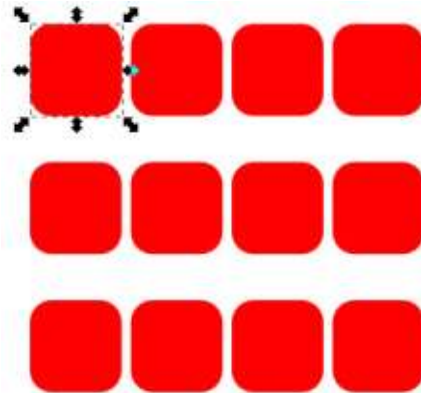
quick way to make a large number of clones. But the real power comes from the myriad ways in which those clones can be positioned and transformed. Let's start by loosening up the array of clones a little by using the Shift tab.



The key to understanding this tab is that the controls are arranged in three columns. The controls in the first column affect how much each row of clones will be shifted around. The controls in the second column affect how much each column of clones will be shifted around. The third column applies to every clone, and is used to add a random amount of shift in the x and y directions.

A simple example to begin with: maintaining the rectangular array whilst adding a little space between the clones. The first thing we want to do is add some space between each column, so we'll put a positive value in the top-middle control. This field alters the x-position (it's in the Shift X row of controls) for each column of clones (it's in the middle column of controls). A value of 10 in here will add 10% of the bounding box width before it's added to the position of each column, so rather than being placed at 100%, 200%, 300%... they'll now be placed at 110%, 220%, 330%... – each subsequent position is increased by 110%, rather than the standard 100%.

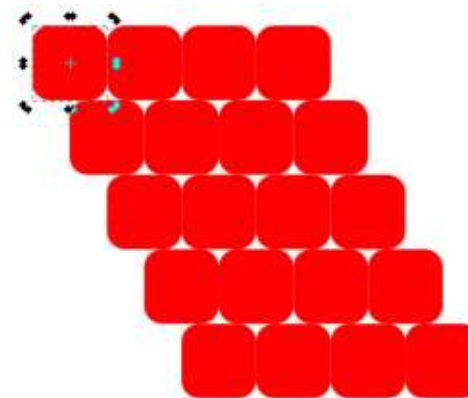
I've also put a value of 50 into the first control on the second line. This will add 50% of the bounding box height to the y-position (as it's in the Shift Y row of controls) to every row of clones (it's in the first column of controls). The rows will therefore be placed at 150%, 300%, 450%... The result is that we've loosened up the array, with more vertical space than horizontal.



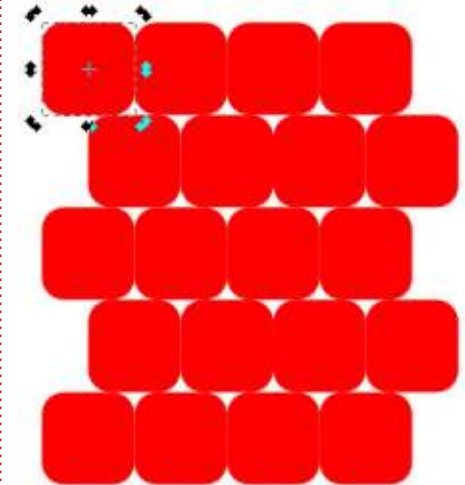
If you don't want your clones quite so rigorously positioned, simply put a positive number into one (or both) of the Randomise fields in the third column. The value you place in here will be used as an upper limit, so entering 20 into the Shift X control will allow the horizontal position of each clone to shift by up to 20% of the bounding box width. This is applied in addition to any other offsets, so

you can still loosen up the whole array as before, then add a little randomness on top. Alternatively you could use the Tweak tool (see part 22), or click on the Unclump button at the bottom of the Tiled Clones dialog, which jostles the X and Y coordinates of each clone a little. The latter can sometimes be useful if you're trying to create a more “natural” look, by cloning a drawing of a leaf or snowflake.

As well as a simple rectangular array, the shift tab can produce more interesting results. Do you need to draw a simple brick wall? Start with one brick, but put a 50% offset into the very first field. This will add 50% of the bounding box width to the x-position for each row of clones. This has the effect of causing each row to move to the right by half of the parent's width.



You can make this look even more wall-like by checking the Alternate control for the Per Row column. This causes the offsets in that column of controls to be applied first as a positive value, and then as a negative. So the rows first shift to the right by 50%, then back to the left by the same amount, then to the right again, and so on.

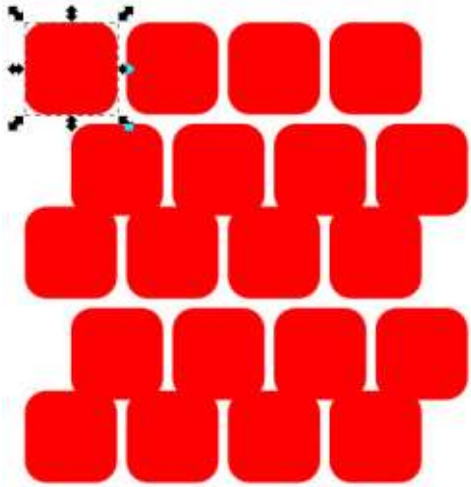


Our wall is looking good, but wouldn't a little mortar help? You might be tempted to space the tiles out by putting 10% into some of the other fields. Whilst this will successfully spread them horizontally, any efforts to spread them vertically will fail because that Alternate setting will also



## HOWTO - INKSCAPE

alter the spacing you enter in the first column of the Shift Y row. Rather than adding a constant 10% spacing between rows, you'll actually be adding 10% then removing 10%, then adding it again, and so on.



There are a few ways to solve this problem, all of which rely on simply adding to the size of the parent's bounding box so that there's no need to add extra padding when creating the clones. You could add a thick stroke and ensure that the geometric bounding box is in use. Create the clones with just the 50% value in the first box and the Alternate control checked and you should get some additional spacing based on the thickness of the stroke. Then just remove the stroke from the parent object, and all the clones will immediately be changed as well.

A variation on this theme is to add some blur to your parent

object. This affects the size of the visual bounding box, so, provided you have Inkscape's preferences set to use the geometric bounding box (remember, they work the wrong way round in this dialog), your clones will get some extra space around them. Then remove the blur from the parent.

The final approach is to put your parent object in a group with another, larger object. The second object is there purely to set the size of the group's bounding box. Create your clones, then enter the parent group and remove the temporary object. This approach results in clones of a group, rather than the object itself, but avoids the need to mess with Inkscape's preferences.

The Cumulate checkboxes determine how any offsets are added to the base position for each clone. Usually, the offset is added to the bounding box width or height once, and that single value is used as the basis of every row or column of clones. Checking this box means that the offset is added again for each row or column, resulting in values that get progressively larger.



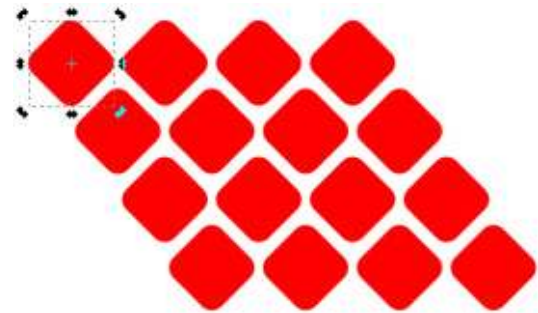
A similar effect can be achieved by setting the Exponent value to something greater than 1. The opposite effect – reducing the difference between each offset – can be achieved by setting the Exponent value to less than 1.

The last pair of controls in this tab, the Exclude Tile checkboxes, can be used to remove the bounding box dimensions from the clones' calculated positions. Settings that would previously have resulted in positions of 110%, 220%, 330%... become 10%, 20%, 30%... when this checkbox is enabled. This allows you to create clones with only a small offset from the parent – usually resulting in them overlapping it to some extent when creating a simple

	☰ Per row:	☷ Per column:	Randomise:
<b>Shift X:</b>	<input type="text" value="50.0"/> %	<input type="text" value="10.0"/> %	<input type="text" value="0.0"/> %
<b>Shift Y:</b>	<input type="text" value="10.0"/> %	<input type="text" value="0.0"/> %	<input type="text" value="0.0"/> %
<b>Exponent:</b>	<input type="text" value="1.00"/>	<input type="text" value="1.00"/>	
<b>Alternate:</b>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<b>Cumulate:</b>	<input type="checkbox"/>	<input type="checkbox"/>	
<b>Exclude tile:</b>	<input type="checkbox"/>	<input type="checkbox"/>	

rectangular array.



One final thing to note is that it's possible to enter a negative shift for the x and y positions. This results in a shift to the left (for x) or upwards (for y), and converts the standard 100%, 200%, 300%... positions to 90%, 180%, 270%... if the offset is set at 10%. This is another way to create an overlapping arrangement of clones, but, depending on the shape of the parent object, it may be just what you need to make everything fit together neatly.



Try playing with a few combinations of values and settings in the Shift tab. You'll quickly find that it's easy to create wild and unexpected clone placements – thank goodness for that Reset button! Try to understand how each individual control contributed to the clones' placement, and how the three-column layout relates to the rows and columns of clones, because, next time, we'll be building on this knowledge to explore some of the other tabs in the Tiled Clones dialog.



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

	 Per row:	 Per column:	Randomise:
<b>Shift X:</b>	<input type="text" value="55.0"/> %	<input type="text" value="10.0"/> %	<input type="text" value="0.0"/> %
<b>Shift Y:</b>	<input type="text" value="-20.0"/> %	<input type="text" value="0.0"/> %	<input type="text" value="0.0"/> %
<b>Exponent:</b>	<input type="text" value="1.00"/>	<input type="text" value="1.00"/>	



# HOW-TO

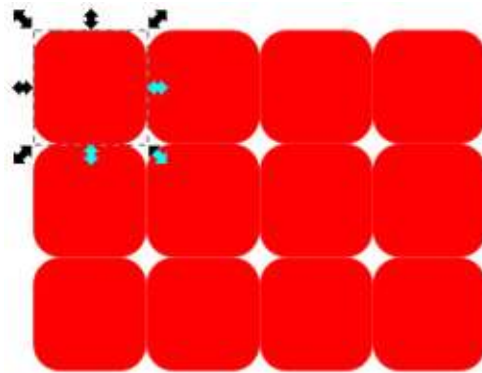
Written by Mark Crutch

## Inkscape - Part 32

**Breaking News:** Since the previous instalment of this series, the long awaited 0.91 version of Inkscape has finally been released. Whilst it has some exciting new features, there's nothing that radically affects any of the subjects I've covered so far, so all the previous articles still apply. I'll delve into some of the 0.91 additions in future tutorials, but, for now, let's carry on with the Tiled Clones dialog, which hasn't really changed with the new release...

Last time, we quickly skipped over the first tab of the Tiled Clones dialog, leaving the pop-up menu on the "P1" setting, then spent the rest of the article looking at the Shift tab. The key thing is understanding how each column of controls applies to the rows and columns of clones that you define at the bottom of the dialog. If you're not entirely clear about that, now's the time to go back and revise because the next four tabs are all based on the same type of arrangement.

Before we move on to the Scale tab, once again you'll need an object or group to clone, and once again I'll be using a simple rounded rectangle. You should also click on the Reset button in the dialog to ensure that you haven't got any odd values hanging around in the Shift tab that will confuse the results. Click the Create button at this point and you should see the same simple array of objects that we started with last time, which will confirm that all the controls are set to sensible base values.



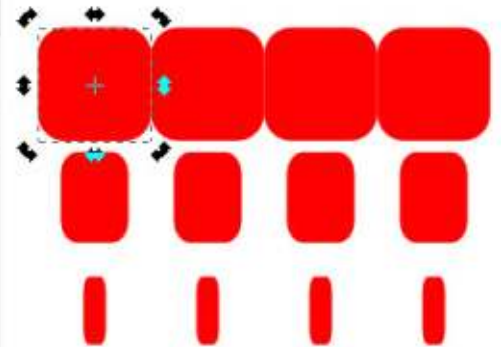
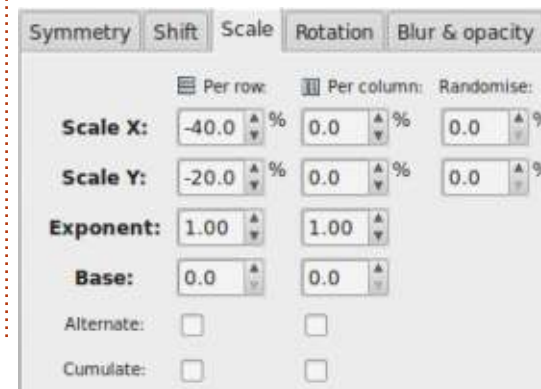
Now let's take a first foray into the Scale tab (shown right). The layout is almost identical to the Shift tab, so you should be able to work out what most of the fields are for. The Shift X and Y rows

have been replaced with Scale X and Scale Y, allowing you to set the amount by which the width and height of your clones are changed for each row and column – plus a random amount if you choose. Clones that have been scaled in this way are exactly the same as if you had manually scaled them using the normal resize handles. As usual, the values are percentages that are relative to the parent's bounding box dimensions. In this example, I've set the values to reduce the width of the rectangle by 40% and the height by 20% for each row.

The Exponent field lets you determine whether the amount of X and Y scale should be the same for each row or column, or

whether it should increase or decrease exponentially. The Base fields are used in conjunction with the Rotation tab to create logarithmic spirals, but I've never really had much luck with the technique. Finally, the Alternate and Cumulate checkboxes work the same way as for the Shift tab. The former allows the Scale factor to be applied as alternating positive and negative values for each row or column, whereas the latter causes the scale factor to be repeatedly added for each row or column, rather than just using the same value for every one.

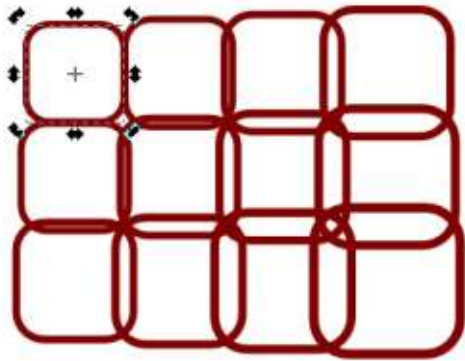
You can, of course, scale up as well as down using this dialog simply by setting positive values for the Scale X and Scale Y fields. If



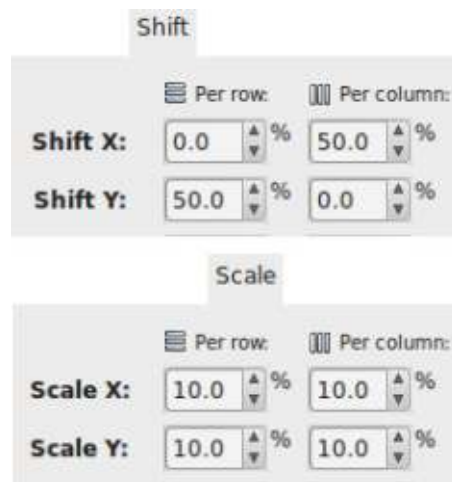


## HOWTO - INKSCAPE

you do this you'll see that the clones immediately start to overlap each other. Here I've set both the X and Y scale factors to +10% for both the rows and columns (in other words, I've put 10 into the four boxes at the top left of the dialog). I've used a shape with stroke and no fill to make it a little clearer what's happening.

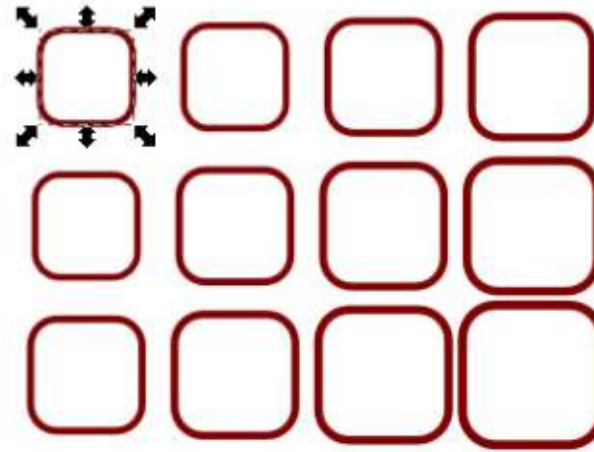
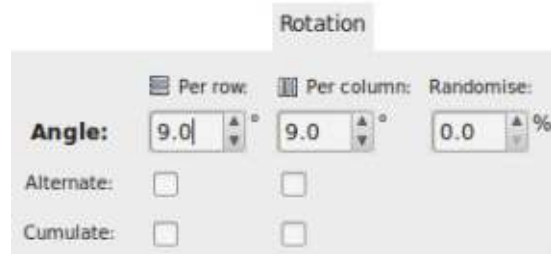


If you don't want your scaled clones to overlap like this, you simply have to give them a little more breathing room using the Shift tab. This is a key point of the Tiled Clones dialog: you can combine options from multiple tabs in order to create the arrangement you want – although it's also easy to create arrangements that quite literally spiral out of control! If your experiments take you too far off



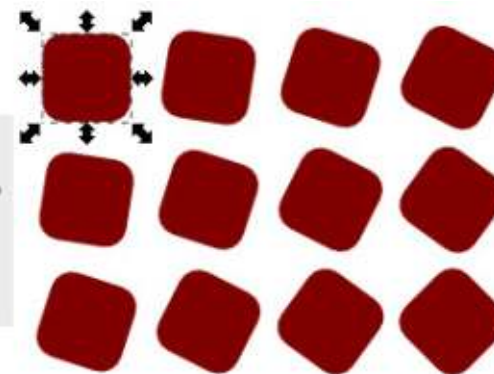
the beaten track, don't forget the Reset button.

Moving onto the Rotation tab, I'm not even going to describe each field because, by now, you should be seeing a common theme across the dialog. Instead I'll just present the following screenshot, and ask you to think about how those values of 9° for each row and column have accumulated into a

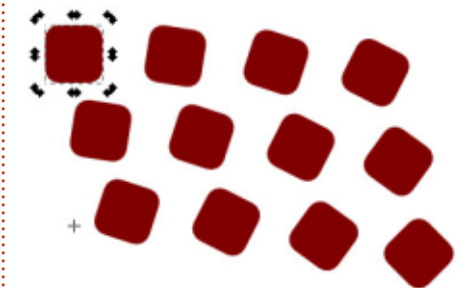


45° rotation of the bottom-right rectangle.

At first the rotate tab seems fairly plain and innocuous. It does what it suggests, rotating each clone according to its row and column position, and that's about it. But there's one vital parameter required for rotating that doesn't even get a mention in that dialog: the center of rotation.



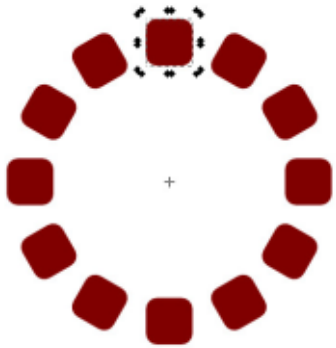
In the previous example I used the parent's default center of rotation, at the middle of the bounding box. But you can move it, as described way back in part 1 of this series: just select an object then click it a second time to bring up the rotate and skew handles, then drag the small cross that marks the center of rotation to some other position. If you want to return it to the default position, just SHIFT-click on it. With the center of rotation moved outside our parent object, the previous rotations become a little more interesting.



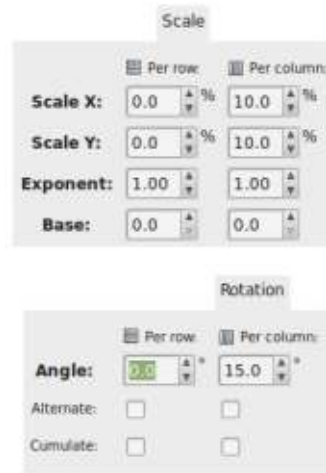
Notice how the arrangement as a whole is starting to curve? We can take advantage of this to create circles and arcs, even though the first tab still claims we're performing a "simple translation". By changing the parameters at the bottom of the dialog to just produce a single row of clones, with a center of rotation outside the parent object, you can

# HOWTO - INKSCAPE

create a circular array. Let's give it a try: set the "Rows, columns" fields to 1x12; adjust the center of rotation to drag it down below your object; set the rotation per column to 30°; finally either check the Per column "Exclude tile" box in the Shift tab, or set the Per column Shift X amount to -100%, in order to counteract the default behaviour of placing each column further along the X axis. Click the Create button and you should have a circular arrangement of clones.

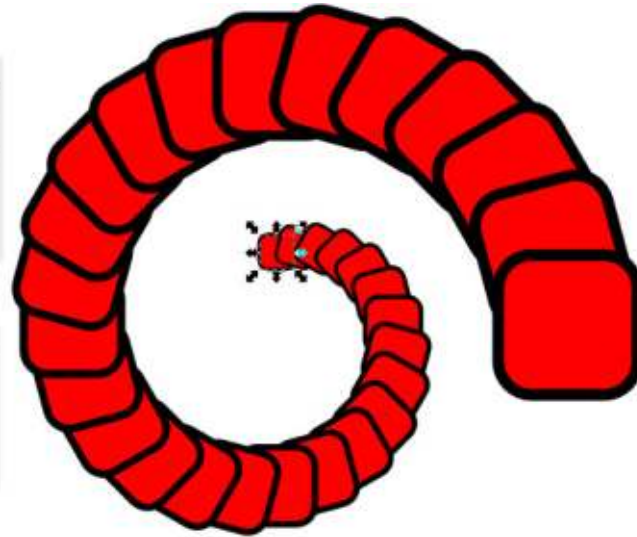


By also putting values into the Scale X and Scale Y fields, it's possible to create spirals in this way. Unfortunately the use of these fields will, of course, alter the size of the clones – I've yet to find a method for creating spirals of identically sized objects using this dialog. This is where the Base fields should allow you to create logarithmic spirals that grow (or



shrink) exponentially, but all they seem to do for me is to distort the clones as they progress around the spiral, so I tend to leave them as 0. Feel free to experiment on your own, though, to see if you can make them perform their magic.

Finally for this instalment, the Blur and Opacity tab should be fairly easy to understand. Tweaking the values in here is the equivalent of setting the Blur and Opacity sliders in the Fill and Stroke dialog for each clone. It's worth noting that any transparency in an object can cause Inkscape and other SVG renderers to slow down a little, as they have to calculate the effect that the pixels behind the object will have on the overall image. Blur has an even more significant effect

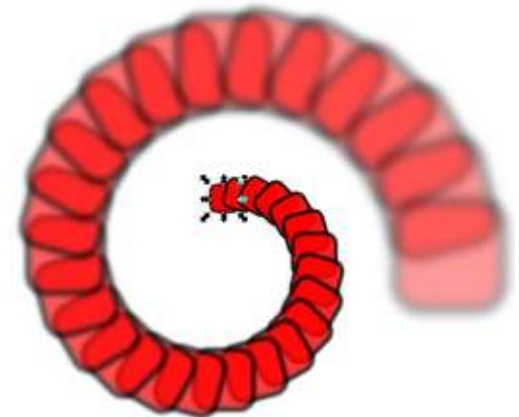


on rendering speed, with larger values requiring ever more intense calculations. It's easy to add too much blur via this dialog, especially when creating a lot of clones, so you should probably start with very small values and work your way up, rather than just going straight for multi-digit numbers.

Be aware that adding blur to clones in this way will actually create a new Gaussian Blur filter for each clone. Filters are a subject for another article, but suffice to say that it's easy to bloat your file with numerous redundant filters, especially when you're experimenting with several different values in this dialog. Using File > Vacuum Defs (renamed as File > Clean Up Document in

0.91) can often remove any obsolete filters, but it's not always 100% successful.

There are no "Cumulative" checkboxes on this tab because these values always add up: if you put 5.0 into the Per row Fade out field, the first row will be completely opaque, the second row will have 5% transparency applied, the third will have 10%, and so on. Applying a little blur and fade to our previous spiral gives this result.



Next time we'll continue our investigation of the Tiled Clones dialog by looking at the last two tabs: Colour and Trace.





# HOW-TO

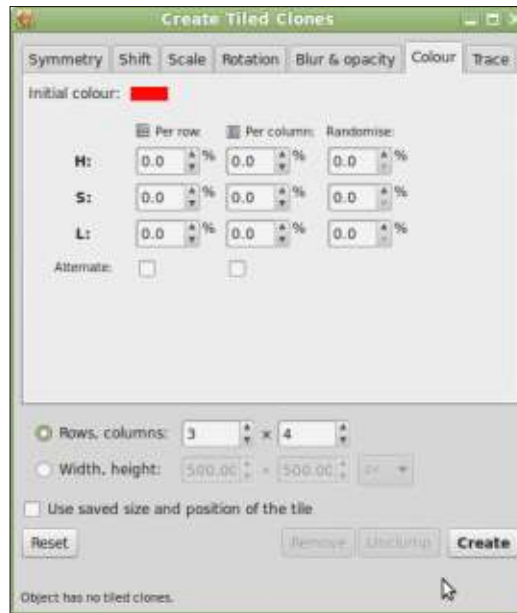
Written by Mark Crutch

## Inkscape - Part 35

Cast your mind back to part 30 of this series where I introduced the notion of an “unset” fill, which allows each clone to have its own color that is independent of the parent object. It's a handy trick for creating a collection of similar-but-not-identical objects, such as a crowd of characters with differently colored hair or clothes. You can use this same mechanism with the Tiled Clones dialog to produce arrays of clones whose colors differ from the parent object either in subtle drifts of shade and hue, or in big, bold steps.

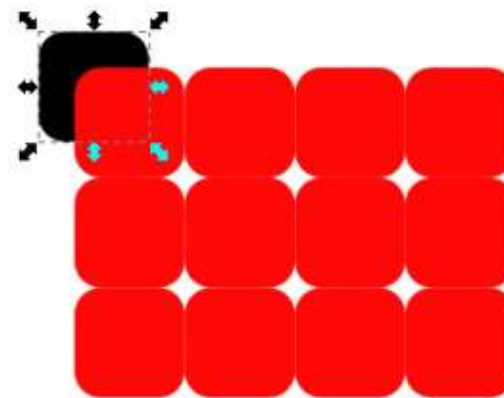
As usual, we'll start by drawing a simple parent shape – our familiar round-cornered square. But, rather than fill it with color, we'll unset it either by using the “?” button in the Fill tab of the Fill & Stroke dialog, or by right-clicking on the color swatch in the bottom-left of Inkscape's status bar and selecting “Unset fill”. We'll also use the Reset button at the bottom of the Tiled Clones dialog to get back to a sensible set of defaults, regardless of your

experimentations as a result of the previous two articles. With all the preparations in place, let's take a look at the “Colour” tab (in my British English installation) of the Tiled Clones dialog.



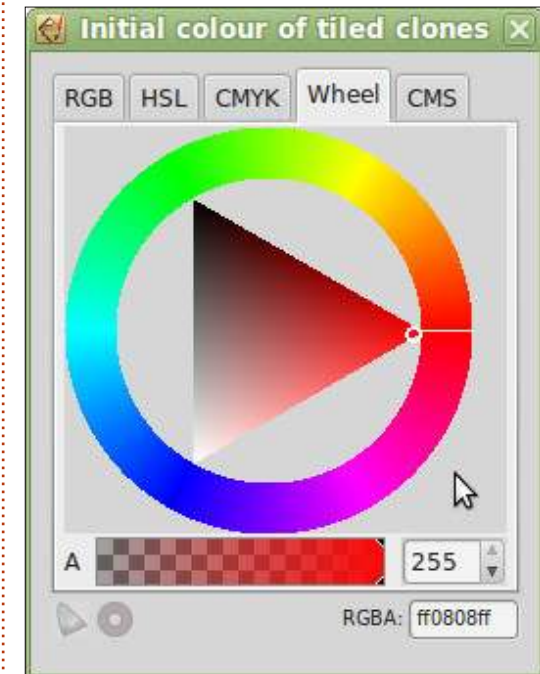
The general layout of this should be familiar by now, but the details differ a little when compared to the tabs we've looked at previously. The first change is the addition of the “Initial colour” field at the top. Clicking the swatch there opens a color picker from which to choose the initial

color that your clones will take. It's “initial” because the rest of the fields can subsequently change the color quite drastically. With everything else in this tab set to zero, clicking the Create button will produce an array of clones, all taking on that initial color. The visual effect will be no different to cloning a solid-colored parent object, so, in this case, we end up with an array of red squares. I've moved the parent out from under the first clone a little, so you can see that its own color remains unset.



The remaining fields in the tab allow us to change H (Hue), S (Saturation) and L (Lightness) for each row and column, with the usual options for randomise and

alternate. If you're not very familiar with the HSL color model, it's perhaps best explained by looking at the “Wheel” tab on any of Inkscape's color pickers. Yes, there's also a dedicated HSL tab, but although I find it to be the more useful for day-to-day use, the wheel view is a better explanatory tool.



Hue, the first of our three values, represents a position on the outside circle. You might expect a value from 0° to 360° – or



the equivalent in radians if you're more mathematically inclined – but that would be too sensible. Instead the range of values available varies in different parts of the Inkscape interface. Within the HSL tab, for example, the numbers run from 0 to 255. Within the Tiled Clones dialog, however, they run from 0% to 100%. In either case, 0 represents pure red with increasing numbers progressing anti-clockwise through yellow, green, blue and purple before the end of the scale brings you back around to red.

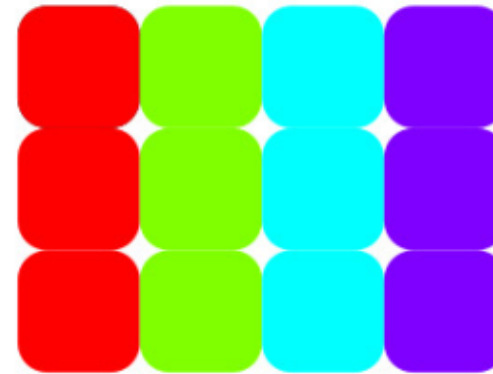
Having picked a base Hue, the triangle in the center is used to select a combination of the Saturation and Lightness values. With the hue set at 0 (pure red), the triangle is oriented as shown in the screenshot. Now imagine a pair of axes, one running from the pure colored corner of the triangle to its opposite edge (a horizontal line in this case) and another running along this edge between the two other corners (a vertical line). Saturation is the position along the first line, and defines the amount of the pure color that's present in the final swatch – how “washed out” it appears. Lightness is the position on the second line,

representing how dark or light the color is. When Saturation is zero there is none of the pure color present, so the result is a shade of gray that can run from pure black (when Lightness is zero) to pure white (when Lightness is at its maximum). The ranges for Saturation and Lightness also run from 0 to 255 on the HSL tab, or 0% to 100% in the Tiled Clones dialog.

The important thing to realize is that the Hue can wrap round – a value of 50% gives you exactly the same pure cyan as 150% or 250%. Saturation and Lightness don't wrap: values above 100% won't suddenly wrap round to lower values, but neither will they result in extra saturation or extra lightness. Values less than 0% behave similarly.

With all that in mind, let's put a value of 25% in the Per Column “H” field. We've got four columns, so the colors will be picked from our color wheel at positions of 0, 25%, 50% and 75%, working anti-clockwise from your selected Initial Color – pure red in this example. It should be easy to see that this gives us red, green, cyan and purple for the columns of our

clones.

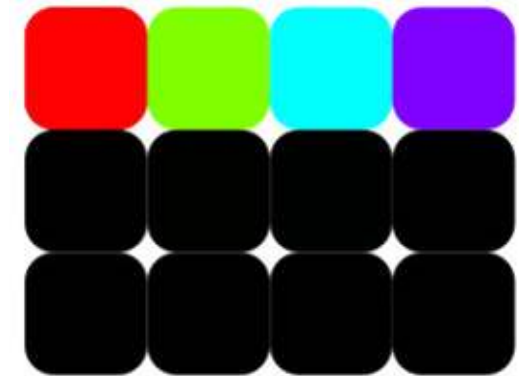


Can you work out what will happen if we change the number of columns to 8? Remember that the Hue value can wrap round. How about if we use a value of 33.3%, 50% or something else entirely?

Now try putting a value of -50% into the Per Row “S” field. With each row you'll get less and less of the pure color included. Given that our starting color is already pure red at 100% saturation, this gives us values for our three rows of 100%, 50% and 0%, resulting in rows that are pure colors, washed out colors, and completely gray. Given that the Saturation value doesn't wrap, can you guess what the result would be for more than three rows? Also try picking an initial color with low saturation and then putting a positive value in the

field instead.

Finally, let's reset our initial color to pure red, and play with the Per Row “L” field. You might expect that putting -50% in here would have a similar effect to the Saturation, giving values of 100%, 50% and 0% for rows that are bright, dark, then black. Instead you get this:



The issue is that the Lightness scale runs from 0% (black) to 100% (white) – pure red, of course, has neither too much white nor too much black, so its value is actually 50%. Thinking of Lightness as running along a vertical line in the earlier color wheel image, it's easy to see that the red corner of the triangle lies 50% of the way up. Checking the HSL tab will also show that your pure red color has a Lightness of 128 (out of 255). Now you should be able to see that a

## HOWTO - INKSCAPE

value of -50% in the field leads to rows of 50%, 0%, 0% (Lightness doesn't wrap either). -25% will give us the expected outcome.



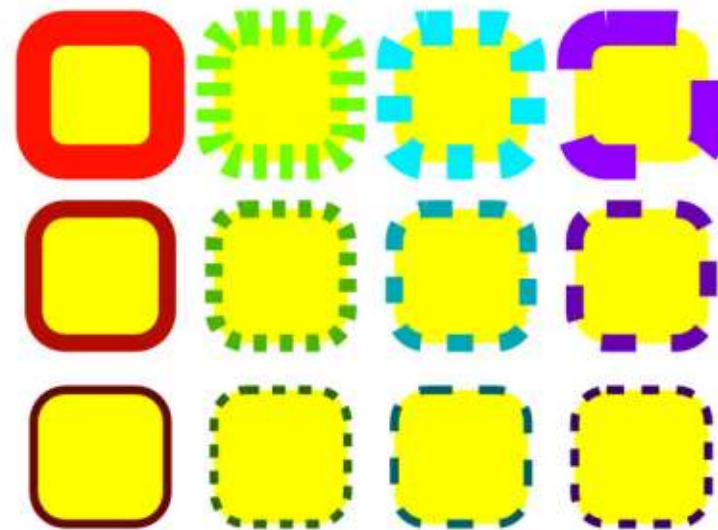
Try creating a larger array of clones with small values in the fields to get gently sweeping changes in color or tone. Or use bigger values – especially in the “H” field – to get bold differences between the clones. Finally, try drawing a simple leaf with veins but an unset fill. Group the parts, then use the tiled clones dialog to create an array of them. With a little use of the Random fields in each of the tabs we've discussed so far – plus some negative offsets in the Shift tab to pull everything together a little – you can quickly and easily create an autumnal forest floor background.

You may recall that it's possible to unset the stroke of a parent



object as well as its fill. This also works with the Tiled Clones dialog, but, as there is only one Colour tab, there's no way to use different generated colors for the fill and stroke: you can unset the fill color, the stroke color, or both, but anything that's unset will be given the same generated color. There's also no way to set any of the other stroke parameters through this dialog – although you can manually set them on each clone afterwards. This means that the Tiled Clones dialog isn't a great help if you want to create hundreds of clones whose stroke width or line style

varies. In this final example, I've cloned a yellow rounded square with an unset stroke, but the



different widths and dash styles of the strokes all had to be manually set afterwards via the Fill and Stroke dialog.

I had promised to cover the Trace tab in this instalment, but the Color tab ended up being a more nuanced topic than I had previously expected, so the Trace tab has been postponed until next time.



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

# 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](mailto: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](mailto:letters@fullcirclemagazine.org)  
Hardware/software reviews should be sent to: [reviews@fullcirclemagazine.org](mailto:reviews@fullcirclemagazine.org)  
Questions for Q&A should go to: [questions@fullcirclemagazine.org](mailto:questions@fullcirclemagazine.org)  
Desktop screens should be emailed to: [misc@fullcirclemagazine.org](mailto:misc@fullcirclemagazine.org)  
... or you can visit our site via: [fullcirclemagazine.org](http://fullcirclemagazine.org)

## FCM# 116

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

## Full Circle Team

Editor - Ronnie Tucker  
[ronnie@fullcirclemagazine.org](mailto:ronnie@fullcirclemagazine.org)

Webmaster - Lucas Westermann  
[admin@fullcirclemagazine.org](mailto: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](mailto: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.