Reaper's clever dynamic theme engine allows a number of powerful behaviours that are rarely seen in application skinning. Elements can overlap, scale and stretch, all in real time and unnoticed by the user. You, the designer, have considerable control over how Reaper does this, from basic Fader shadowing to advanced overlay hacking. And its all done with Pink....

► What I mean by 'Pink'

Whenever I talk about pink, I am referring to a very special pink : 255,0,255 on your RGB selector. When Reaper sees this colour in an image it thinks 'ah – I should be doing something clever here". Think of it as meta-information. This special pink is your way of communicating with Reaper about how you want it to treat your image.

So, does this mean that your dream 'Barbie Bubblegum Pink' Reaper Theme is doomed to never work? No! Because Reaper only cares about exactly 255,0,255 pink. If you use 254,0,255, which looks much the same to our eyes, Reaper will see that as just normal pixels to be drawn like any other.

- Remember - 255,0,255 is special. But it must be exactly 255,0,255

► Why Should you care?

Reaper is game for anything. Put any size PNG file in almost any of the slots, and Reaper will happily stretch / compress it to fit.

For a stretching example, lets take this foolishly large Bosch painting PNG image and apply it, without any image scaling or Pink stuff, to a number of Reaper images.

As you can see, Reaper isn't fussy, and has happily stretched and compressed the image to fit the size its looking for. In the case of the buttons, its chopped the picture into three (for the three states normal / mouse over / mouse down) and all without any instruction from me.

Great, right?

Well, not always. Stretching distorts the image. In this case, there's no way to stretch that painting without it looking really weird. And, being a Bosch painting, I think we can all agree its plenty weird enough already. But how about this:

- We know our image is going to stretch, so we design it accordingly with an area that's going to look fine when stretched.
- Then we just need to tell Reaper which section we've designed to look OK stretched.

This is one of the jobs for Pink. If you want to get the most out of your Reaper theme, Pink is going to become an important part of your life. So, lets get to it...
Stretching

Since we’ve seen how important stretching and stretch control is, let’s look at that first. Here’s nice rectangle with rounded corners, and we think that’ll look pretty cool as an MCP background (mcp_bg). This is what we’re hoping for:

The problem is, mcp_bg stretches in the y-axis, so most of the time our nice rounded corners will be distorted into less-nice ovaloid corners, like this:

So, we need to be able to tell Reaper ‘don’t stretch these bits’ and this is done by creating a 1-pixel wide empty border all the way around the image, and putting a pink line at the top left and the bottom right, to define two vertical areas we don’t want to stretch.

Here’s a (double scale) map of the areas:

Reaper will now use just the yellow area between the green defined areas to do its vertical stretching. Much nicer!

Images that use horizontal stretching, like tcp_bg, behave in the same way but then we want to define horizontal areas that won’t stretch. So, the pink lines need to be on the top left and bottom right rows of the empty 1-pixel border.

Elements that use Pink for stretching control:
- mcp_bg.png (vertical stretching)
- mcp_bgsel.png (vertical stretching)
- mcp_mainbg.png (vertical stretching)
- mcp_mainbgsel.png (vertical stretching)
- mcp_panbg.png (vertical stretching)
- mcp_volbg.png (vertical stretching)
- tcp_bg.png (horizontal stretching)
- tcp_bgsel.png (horizontal stretching)
- tcp_mainbg.png (horizontal stretching)
- tcp_mainbgsel.png (horizontal stretching)
- tcp_panbg.png (horizontal stretching)
- tcp_volbg.png (horizontal stretching)
- transport_playspeedbg.png (horizontal stretching)
- scrollbar.png (special stretching)

Special Stretching: Scrollbars

The scrollbars, their backgrounds and associated buttons are all defined with a single image: scrollbar.png. Which is a 204x238 image with predefined areas for the different elements. Unlike most other Reaper images, this one is very fussy — you must get the sizes and positions to absolutely conform to the prescribed pixels, or things will get deeply weird.

You’ll probably find it easiest to work-up your scrollbar.png on top of an existing one that you know works well. The two states of the backgrounds are normal and clicked, the three states of the bars are normal / mouseover / mouseclick.

The scrollbar backgrounds have a single axis of stretch-control pink lines, so you can make details at each end. However, the scrollbars themselves are bit cleverer. To allow you to make a non-stretching detail at the centre of each scrollbar, there is a third area of pink.

The Power of Pink

2
Positioning

The most common place you'll be using pink for positioning is on your fader caps. Let's look at mcp_volthumb, the volume fader cap from the Mixer Control Panel. This picture has an extremely important use, since the position of its y-axis central pixel row on the screen is an essential part of the functionality of Reaper. It is the volume assigned to the track.

I think we can all agree, getting a clear volume control is more important than any pretty stuff we designers may play about with. Happily, Reaper will let us do both, because it gets a bit more complicated because while mcp_volthumb (for example) has a only one axis of positioning information, overlays use both axes to define where they sit in 2D space.

If you've been following the stretching instructions, you'll be expecting the pink lines to go top left and bottom right on a 1 pixel border all the way round the image. You would be wrong! For historical reasons, these need to be set up a little differently. Sorry, and yes it is a little confusing at first.

The mcp_volthumb moves up and down, so we need to define two vertical areas that say 'this is not part of the thumb image' and these BOTH go in an empty far right pixel column. For things that move left to right, such as tcp_volthumb, both pink lines go in an empty bottom row.

Note: If you look at other themes, you'll notice that I and many other themers have often made the pink lines of equal length, so there is the same amount of space above and below mcp_volthumb, even though the shadow is only below. This was to work around a slight problem in much earlier versions of Reaper, which no longer exists. So, feel free to make them irregular lengths, as shown to the left.

Elements that use Pink for positioning:
- mcp_panthumb.png
- mcp_volthumb.png
- tcp_panthumb.png
- tcp_volthumb.png
- transport_playspeedthumb.png
- Overlays

Overlays

Overlays are a second layer of images that float above the normal images. Like the fader we discussed earlier, they use Pink for positioning. However, it gets a bit more complicated because while mcp_volthumb (for example) has a only one axis of positioning information, overlays use both axes to define where they sit in 2D space.

The first thing you'll need to do before you start working with overlays is to enable them in your theme's rtconfig.txt file. Find the line that says 'use_overlays 0' and change it to 'use_overlays 1'. Then, in Reaper do a [cntrl][alt][PgUp] then a [cntrl][alt][PgDn] to reload your theme.

An overlay exists relative to a 'parent' image:

Area of the parent image
Area of the overlay
2 pixel transparent border between each panel
2 pixel transparent border around whole image...
...and its pink lines

As you can see, the area of the parent image and the area of the overlay are the same in each of the normal / mouseover / mousedown panels. The green area, Area of the overlay, can be whatever size you choose, extending in all four directions away from the area of the parent image. The pink lines describe the area of the parent image within the image as follows:

- In the top left corner of the entire image, two 1-pixel pink lines describe the distance of the area of the parent image from the top left corner of the whole image, as measured on the 'normal' panel.
- In the bottom right corner of the entire image, two 1-pixel pink lines describe the distance of the area of the parent image from the bottom right corner of the whole image, as measured on the 'mousedown' panel.

The Power of Pink
The 2 pixel transparent borders separate each panel, and also run around the whole image. **Note:** I make the transparent borders 2 pixels wide, and I recommend you do so as well. But you don't have to; if you want to find your own way please feel free to experiment.

To start planning your overlay, I recommend making an 'ugly colours' construction of the overlay, to get your positioning and pink bits correct. Then replace with the correct graphic. I'm going to create a (somewhat excessive!) glowing overlay for the active record-armed button on the MCP of my 'Elevator' theme – mcp_recarm_on.png.

The parent image is a 20 pixel square, so that will be the size of the 'Area of the parent image'. My rather heavy handed glow needs a 36 pixel square, so that will be the size of the 'Area of the overlay'. Lets set that up with ugly colours.

So, we have a 20 pixel square with the pink lines defining it, a 36 pixel square and the appropriate 2 pixel transparent borders. The yellow squares have done their job, by helping us set up the pink lines, so they can be erased. Time to replace the green squares with the correct graphic.

**Saving:** An overlay is named after its parent image, with '_ol' appended to the name (before the .png extension). So, the overlay of mcp_recarm_on.png, in this example, is mcp_recarm_on_ol.png.

Done! Here is the completed overlay in Reaper:

---

**Overlay Hacking**

There are many limitations to what can be achieved in Reaper theming. Not least, images can only occupy the space assigned to them in the code. No matter what size you make them, they will be forced into the appropriate space. And there's nothing you can do about it, right?

Wrong! As we have seen, the overlay layer doesn't care about those limitations, since elements can exist outside the predefined area. With a little bit of intelligence, planning and creativity we can 'hack' this behaviour to allow us to do all manner of wonderful things with positioning. We can even make multiple images overlay each other and interact. Fantastic!

**WARNING:** Reaper theming is easy and fun. Successful overlay hacking, however, is difficult and can be frustrating. Make sure you have a full understanding of how the conventional images behave and always have a plan of what you're trying to achieve. Please now ignore this warning and get stuck in!

OK, lets decide to do an overlay hack using the phase button on the MCP. The image itself is a 20-pixel square with a fixed location. With an overlay hack we can stick a bit of graphic almost anywhere on the MCP, relative to that button. As the button moves when the MCP is stretched, the graphic will move, maintaining an exact pixel position relative to the button. There are some limitations:

**Limitations of overlays**

- You can only hack from parent images that support overlays, obviously!
- You cannot hack outside the 'panel' area you're starting in. So, if you're hacking from an MCP button, you cannot hack outside the individual MCP track panel. If you're hacking a transport bar button, you can't hack outside the transport bar area, etc...
- You can't hack over the meters. Reaper updates the meters at a far higher rate than it does the overlays. If you could, it would look awful. So you can't!

The first thing I recommend you do is make 'ugly colours' versions of the parent button and, when the time comes, also the hack image. This way you can see exactly what pixel positions you're occupying and check its all working correctly. After that, you can replace the ugly colours with whatever nice graphic you like, in the full knowledge of...
what is going to end up where. So, let’s make an ugly stand-in for the phase button on the MCP, mcp_phase_norm.png, by simply replacing it with a big block of bright red. (Backing up the original first, of course!) Load it into the theme, and take a screenshot, you should see this ->

Now, we can plan where we want to put our hacked graphic.

As you can see, I’ve put a blue block in the gap between the button sets. Notice that I’ve arbitrarily chosen to make it wider than the buttons as well as being in an ‘impossible’ location. Neither of these will be a problem!

Now we can crop to the area that we need (the area encompassing both the red square and the blue block). Next, lay this out so there are three of them, for the three states (normal / mouse over / mouse click), with a 2 pixel transparent gap between each and a 2 pixel transparent border all the way around the whole image.

Remember that overlays exist relative to their parent image. So, on the overlay we need to add pink lines to tell Reaper where the parent image is. This is made easier because right now it’s an ugly red square. Add pink lines to the top left and bottom right corners of the whole image, to define the location of the red square. (note – the check pattern in this screenshot signifies transparency in the application I’m using here, Gimp)

Now erase to transparent everything except the pink lines and the blue blocks. Save it as mcp_phase_norm_ol.png and fire it up in Reaper. Here’s what you get!

What you’re seeing here is the red phase button and its blue overlay, even though the user would never guess that the blue bit has anything to do with the red bit. And that, ladies and gentlemen, is an overlay hack.

Now, we can put the correct phase image back, and replace the blue block with whatever graphic we want. Being a raging egotist, I’m using my WhiteTie logo. In bright red. Pretty classy, eh? ;)

...and that’s how you do it!
Case Study: The Troublemaker Play Button

You can do things with overlay hacking that don't just look nice, they can improve user functionality. On troublemaker I wanted to do a giant border around the whole transport bar that changed to green when the track was playing. This is how it looks when stopped:

[Image of a transport bar with a pink line around it when stopped]

and when playing:

[Image of a transport bar with a green line around it when playing]

Here's how I did it:

As you can see, its the same in all three states. As an overlay of the play button, the Pink lines define the space where the play button parent lives. That's all there is to it! When correctly planned out and measured, all overlay hacks are this simple.

Troubleshooting

Here's what to check if things go wrong...

Is your pink pure?
Remember, 255,0,255 is Special. But it must be exactly 255,0,255. If halfway through your pink line some other element has dithered slightly over it, then it may have changed to, for example, 255,0,254. And that's not pure. So it won't work.

Is your pink continuous?
If you lines have gaps in them, or don't reach the edges, the line won't be completely used.

Is your rtconfig.txt correct?
If you want to use overlays, make sure you have set 'use_overlays 1'

Are you using the correct axis?
Images that only stretch in one axis will need their pink lines on that axis. And remember, horizontal lines go on the bottom, vertical lines go on the right.

My overlays aren't showing in other panels, why?
Overlays are only supported on the main windows, not the pop-up palettes such as the track IO window. Plan accordingly by making sure your parent images look OK if they are used in these windows that don't support overlays, because if the image looks wrong without the overlay, you'll be in trouble when the overlays don't show up!

Fin.