3. Using the Conduit Effect System

Conduit Effect System is the rendering core of PixelConduit. It's a powerful hardware-accelerated rendering system that you can access using a node-based graphical interface. It's also available as plugins for other apps like Final Cut Pro and After Effects.

To create effects, you need a *Conduit Effect* node widget in the Project window. Usually there is one already created by PixelConduit when you start a new project.

Conduit Editor is the window that opens when the "Edit" button is clicked on the *Conduit Effect* node widget.

The Conduit Editor presents a very powerful interface for designing visual effects. An effect is built up from *nodes* which represent image operations like color manipulations, blurs, or simple math (such as Add or Multiply). Connections are made between nodes to determine the order of operations. Using these building blocks, pixels can be manipulated practically without restrictions.

Conduit's node environment has some unique features:

- Designed for realtime.
  Conduit makes full use of the powerful GPU (graphics processor) found in modern computers. Individual nodes are compiled together for maximum performance. Conduit is particularly fast for operations like keying and color corrections which can be compiled so they are rendered simultaneously.

- Pervasive floating point color.
  Unlike typical computer graphics systems which have a limited range of color values (for example 0-255 per channel), Conduit uses floating point color everywhere. For example, you could brighten a pixel value 10000 times, then darken it to down to 1/20000, and it will never get clipped or lose precision. This is a powerful feature on its own, but together with the extensive high-depth file format support in PixelConduit, it allows you to access your video data in ways that no other application does (see *Pro Pixels* in this guide for more information).

References and other documentation

To get the most updated information, see these online documents:

- **Conduit Cheat Sheet (Frequently Asked Questions)**
  http://lacquer.fi/conduitdoc/Conduit_Cheat_Sheet_(FAQ)

- **Conduit Node Reference**
  http://lacquer.fi/conduitdoc/Conduit_Node_Reference

The Node Reference is a long document and it's not reproduced here. To find out the specifics of how a particular node works, please see the online version at the above link.

There is a tutorial called *Drowned World* included in this book that shows concrete examples and techniques for compositing and creating "looks" in Conduit.

Don't forget that you can also use Conduit in Final Cut Pro and After Effects! Using the Conduit plugins, the Conduit Editor becomes available in these applications. Once you've created an effect, you can easily transfer it from PixelConduit to FCP or AE for use in editing and compositing, or vice versa.

The Conduit plugins are available on the PixelConduit web site, *http://pixelconduit.com*.

How values are transferred from PixelConduit to the effect

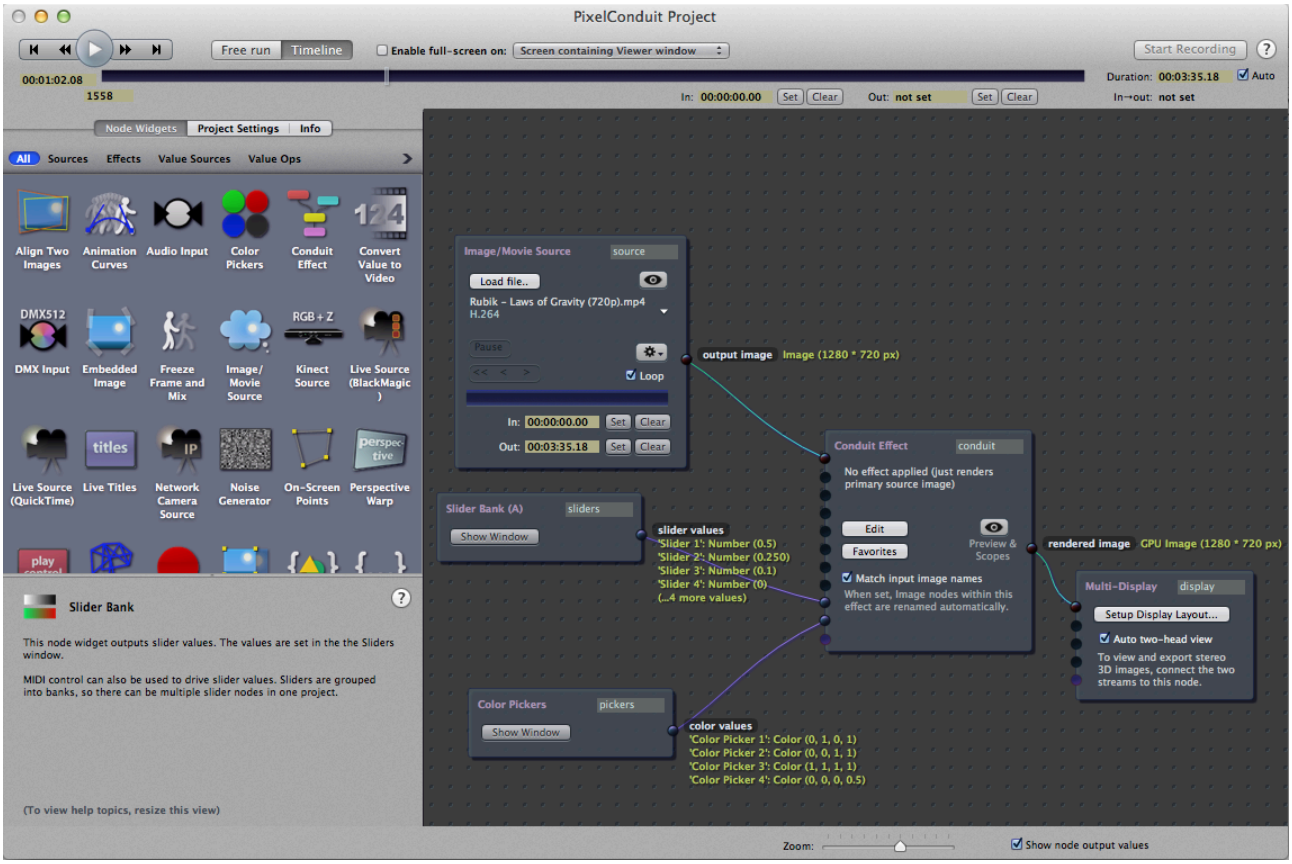
Because PixelConduit's project layout is so customizable, it's not always obvious how the *Conduit Effect* node widget seen in the Project window relates to the nodes that you see in the Conduit Editor.

An effective way to think about the Conduit Editor is that it shows "what's inside the box" for the *Conduit Effect* node widget.

To understand how things come together within the Conduit Effect node widget, we'll note that it has a total of 11 inputs. These are:

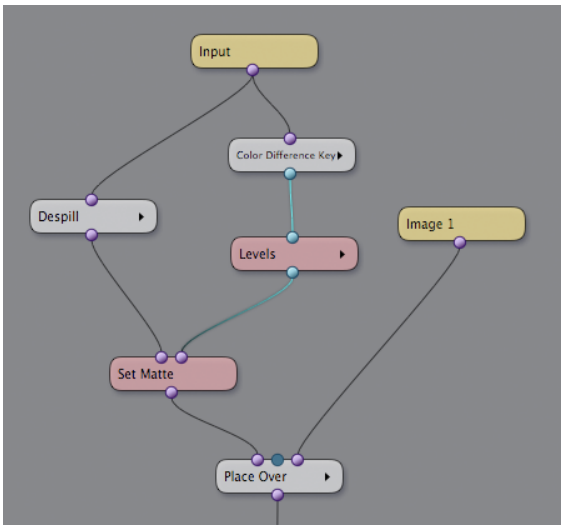
- Eight **Image** inputs
- One **Slider values** input
- One **Color picker values** input
- One **Custom values** input

In the next screenshot, two image inputs are connected, as well as the slider and color picker inputs:



Within the Conduit Editor, these connected input values become available to us using nodes. It's up to you to decide how the images and values are used.

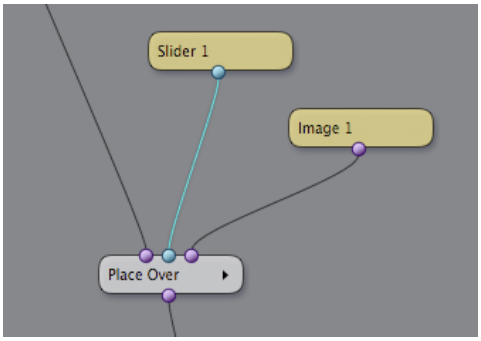
The next image is a screenshot from the Conduit Editor. It shows a color keying effect that uses two images:



"Input" here represents whatever image we connected to the first input of the Conduit Effect in the Project view. In the Project screenshot above, this was an *Image/Movie Source* node widget.

"Image 1" represents the other image that we connected to the Conduit Effect. In the above screenshot, this was a *Test Pattern* node widget.

Note that we're not currently using the sliders for anything within the effect. Something that is often necessary is to control the foreground image's opacity using a slider value. That's easy to do. Create a *Slider* node in the Conduit Editor, and connect it to the middle input of the *Place Over* node:



Now we can drag “Slider 1” which is shown in PixelConduit’s *Sliders and Controls* window, and the Conduit Effect will respond instantly by modifying the image's opacity.

This way of setting up layering is admittedly more involved than stacking up images Photoshop-style. Conduit's approach of using nodes really comes into its own when you want to do something more complicated. For example, what if you wanted to subtly modify a color correction effect together with the layer opacity? In most applications, it can't be done. In Conduit, it's a simple matter of connecting the Slider node to a parameter in the color correction node.

Remember that you can modify slider values within Conduit using all the nodes at your disposal. You can use this to build operations that would need to be done using programming or “expressions” in many other applications. For example, you might want to scale a slider value so that it goes up to 50 – perhaps to control a blur amount. This is easy to accomplish in the Conduit Editor with either a *Multiply* node or *Change Range* node placed after the Slider node.

Effects essentials – the “Conduit Cheat Sheet”

Creating effects in the Conduit Editor is a large topic. Instead of trying to cover everything, this section is formatted as a Frequently Asked Questions (FAQ) style list of topics. It’s updated also on the web, so if your question isn’t answered here, have a look at the online version at:

`http://lacquer.fi/conduitdoc/Conduit_Cheat_Sheet_(FAQ)`

What's a conduit?

A "conduit" is a visual effect created in the Conduit Editor. It's like a flow chart that reads from top to bottom, explaining all the operations that are performed on the images to produce an output.

These operations are represented as "nodes", visual blocks that have inputs and outputs. By connecting nodes, you decide the order of operations that happens when the conduit is rendered.

The Conduit Editor is the core of Conduit. It works exactly the same in all the Conduit products, whether you're using the standalone PixelConduit application, or Conduit as a plugin inside an application like FCP or After Effects.

Conduits are stored in the ".conduit" file format. To save and open these files, click the "File" button in the top right-hand corner of the Conduit Editor.

What are sliders and color pickers?

They are an easy way to control your conduit from the outside.

When using Conduit as a plugin, these values are controlled from the host application's interface. In Final Cut Pro, you'll find the sliders and color pickers in the Motion tab. In After Effects, they're in the Effect Controls tab.

In PixelConduit, you can set up these connections in the Project window and use the Sliders and Controls window to modify the values. See the section *PixelConduit user interface* for more information.

By default, the sliders and color pickers don't mean anything. You must give them a meaning within your conduit by connecting them to some values. For example, Slider 1 could be the opacity of a compositing node, while Slider 2 could be the amount of blur applied.

This is done by using the Slider and Color Picker nodes within your conduit. They are found in the "Inputs" category in the Conduit Editor.

You can use all of Conduit's nodes to modify the slider and color picker values as desired. For example, a slider's default range is 0 - 1, but it's often useful to apply it with a larger range such as 0 - 50 (perhaps to control a blur). This is easily accomplished by multiplying the slider's value by 50. Simply place a Multiply node after the Slider node.

Where's the eyedropper tool?

(The eyedropper tool is used to pick colors directly from an image.)

In PixelConduit, you can find it in the Viewer window. When you pick a color using the eyedropper, the color value is applied to the first color picker. You can see it in the “Sliders and Controls” window. (To edit the color, drag the sliders or click on the color swatch.)

For the Conduit plugins, the location of this tool will depend on the host application. You can usually use the host app's eyedropper tool to pick a color and place it into one of the color picker inputs for the Conduit filter.

In After Effects, Photoshop and Aperture, Conduit displays a preview of the image alongside the Conduit Editor. The eyedropper tool is also available there.

Where's Brightness & Contrast?

We recommend you use Levels instead, it's more flexible. See the next question on how to use Levels!

If you miss Brightness and Contrast, you can create the same effect using Add and Multiply nodes. First use Add to control the level of contrast falloff; then use Multiply to apply contrast; then another Add to apply overall brightness as needed. (But don't forget that Levels does this same job in one step.)

How to use Levels?

There are three parts to Levels: the input, gamma, and the output.

(By the way, if you know how Levels works in Photoshop, then all you need to know is that it works the same way here.)

Input levels defines the contrast of the input image. Choose the black and white points by dragging the "Input black" and "Input white" sliders.

(You can use the histogram to get a clear idea of how the tones of the input image are distributed. Because calculating the histogram can be expensive, it's disabled by default, but can be enabled by toggling the checkbox in Levels' parameters.)

Gamma defines the tone curve applied to the image. You can use it to make shadows brighter, or highlights darker.

Output levels defines the tone range of the output image. To brighten the image overall, increase the "Output black" value. To increase the contrast of the output image, increase the "Output white" value. (Although the slider only goes to one, you can increase the value beyond one by clicking on the arrows around the number, or simply clicking and dragging the displayed number.)

Levels in Conduit allows you to create HDR values by simply toggling the checkbox that reads "Clip values to 0-1". That means you can use Levels to scale an image's brightness to a "superbright" range. In the Viewer, pixel values beyond one will appear white, but the data is still there.

What's alpha?

Alpha is the name commonly used to mean a transparency channel. It is a greyscale image that's associated with your actual color values (those RGB channels - red, green and blue). The alpha channel determines which pixels are visible, and how transparent they are.

When the alpha channel is all black, no pixels are visible. When it's all white, the entire image is visible.

This sounds fairly simple, but there's an extra complication because there are actually two flavors of alpha in common use: "premultiplied" and "unpremultiplied" (also known as "straight"). Read on...

What's premultiplied alpha?

Premultiplied alpha means that the RGB pixel values have been multiplied by the alpha value. That means, if the alpha is zero (completely transparent) at a particular pixel, the color value will be black.

What's the point? Wouldn't it be better to avoid premultiplication and keep the color data intact? Yes, it would be nice... But due to a number of factors, it's often much faster for computers to render images with premultiplied alpha. Hence you'll find that a lot of image files contain this kind of alpha.

If you want to import unpremultiplied alpha into Conduit, the TIFF and OpenEXR formats should do the trick.

PixelConduit also expects your images to be premultiplied when displayed. (Again, this is for performance reasons.) If you apply an alpha channel to an image but don't premultiply it, the transparency will not display correctly in PixelConduit's Viewer window.

If you're using the Over node to composite, it has a "premultiply" checkbox that can be used to do this. Otherwise, you can always place a Premultiply node at the end of your conduit.

What's a matte?

"Matte" is simply another name for an alpha channel. The word "matte" comes from visual effects tradition. It means a mask that reveals some parts of the image.

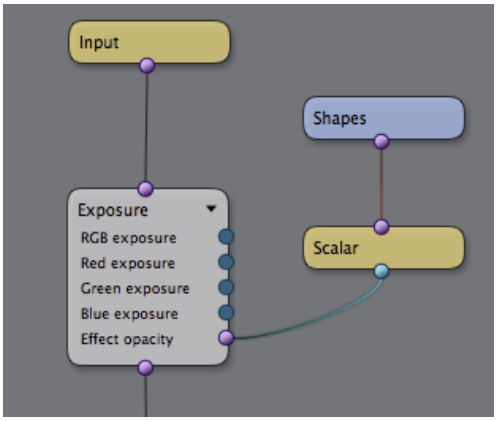
Conduit has a node called Set Matte, which sets the alpha channel of an image.

How do I mask an effect (i.e. limit an effect so that it renders only within a specified matte)?

It's fairly common that you want to apply an effect only within a specific part of the image, rather than the image as a whole.

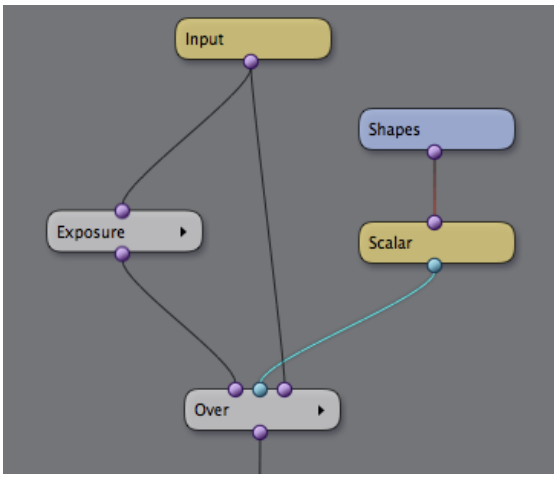
There are two easy ways to mask an effect in Conduit.

Many nodes have a parameter called "Effect opacity". This makes it really simple to mask the effect. Simply connect the matte image to this parameter's input. In the following screenshot, an Exposure node is masked using a Shapes node:



Note that the Shapes node's output needs to be converted to "scalar" (i.e. grayscale) so that it can be used as a matte. The purple connector on Shapes indicates that its output is a color image. The Scalar node simply converts this to a grayscale image by taking the first channel.

The other way to mask an effect is to use an Over node. This way, we simply composite the effect over the original image using the matte:



This method works for masking anything.

Problem: transparent areas are displaying as bright.

This is happening because alpha has not been premultiplied. See *What's premultiplied alpha* above.

Problem: composited object has a black outline.

This is happening because the object's alpha channel has been premultiplied, but you're using a compositing operation that expects straight alpha. See *What's alpha*.

If you're using the Over node, just toggle the checkbox that is labelled "Foreground is premultiplied".

What's gamma?

It's a color correction that's typically applied to digital images. See the next answer to find out why…

What's linear light?

The concept behind linear light compositing is fairly simple. It's all about making our pixel values behave like real light values. This is done by removing "gamma correction" from the source images.

Gamma correction is a process usually applied by the camera. When the image sensor behind the camera's lens captures an image, the picture is in a linear light format – each pixel corresponds to an actual light value. But digital images are not stored like this. The fundamental reason is that human vision does not perceive lightness values in a linear fashion: to our brains, darker areas appear lighter than the actually are. The camera applies a gamma curve to the image data in order to make the pixel values correspond more closely to how the viewer will perceive them.

This is fine for viewing images, but in compositing, we want to work with something closer to actual light values. Consider a situation where we would like to add a semi-transparent screen into an image.

The screen would block 50% of the light. If we're working in linear light, we can simply use a black layer at 50% opacity, and the resulting effect will look "right" in a way that's difficult to approximate when working with gamma-corrected images.

Conduit has excellent support for linear light compositing. To convert your images to linear, all you need to do is apply the Convert Video to Linear node.

There's no need to worry about loss of precision, because Conduit always works at floating point precision. See the next answer...

What's floating point color?

Floating point color precision means that pixel values are not constrained to any fixed range. Typically pixels in digital images are limited to a very specific range, for example 0-255 for so-called "true color" or 24-bit images. These traditional images are better described as having 8 bits per channel, in contrast to floating point images which can have 32 bits per channel.

The major advantage of using floating point pixels instead of a fixed range of values is that you're suddenly free to use the whole range of numbers to represent your images. When working in floating point, a single image can contain brightness values of 1/1000 units in the shadows, and 1000 units in the highlights. That's an enormous amount of dynamic range. (Note that the "unit" doesn't have a fixed meaning. It's up to you to determine what a pixel with a value of 1.0 means.)

Floating point pixels can even contain negative values. This can be used for interesting effects, e.g. compositing with negative images that "eat" parts of the background image. (Negative values are also of interest for rendering effects like displacement and normal mapping.)

Conduit always works in floating point mode. Pixel values are never clipped, unless you explicitly ask a node to perform clipping.

What's HDR?

HDR is short for High Dynamic Range. It's an umbrella term that essentially means "more color precision, and no limits on pixel values". This is made possible by using floating point values for pixels. (See the previous answer.)

In photography, HDR images are usually created by combining multiple exposures.

Conduit supports HDR by default with no special settings needed because it always renders in floating point mode.

When to use the Bezier/Cubic Curve nodes instead of Curves (RGBA)?

The Curves (RGBA) node is powerful, but it's a bit slower to render than most other color correction tools in Conduit.

If you don't need all the features in Curves (RGBA), consider using the Bezier Curve node. It offers a 4-point curve that is adequate for a lot of color corrections, and it renders very fast.

The Cubic Curve node is an even simpler type of curve. (On modern graphics hardware, it doesn't have a significant performance advantage over Bezier Curve, but it can still be useful in its own right.)

How to rotate an image?

Use the Place Over node. See also the next answer...

What's the difference between Place Over and 2D Transform?

Place Over is a new node in Conduit 2.0. It allows for an image to be rotated, scaled, translated, and then placed on top of a background image. (You can also use Place Over to simply perform the transformation, without compositing on a background.)

2D Transform is simpler. It only does scaling and translating.

Another difference is that 2D Transform will essentially treat the input image as being of project size, whereas Place Over will composite the foreground image in its original size. This matters when you're combining elements that have a different resolution.

For example: assume your project is rendered at 1280*720, and you want to import a small graphic element which has a resolution of 400*300. Using Place Over, you can composite this graphic so that its original size is retained.

Place Over also has special support for more easily animating elements in PixelConduit. Any Place Over node can be controlled by a source in the PixelConduit project view. Using this feature, a layer's position and

transformation can be animated by a tracking or scripting node widget in PixelConduit. (To use this feature, connect a node widget to a Conduit Effect's "custom values" input. These values will become available in the Place Over node's interface.)

How to create a macro (or capsule)?

Use the Supernode. See also the next answer...

How to package multiple nodes into one?

The Supernode is one of the most powerful features in Conduit. Not only does it allow you to package multiple nodes into a single unit, but you can also save that node as a plugin that works just like Conduit's built-in nodes. Supernode also has an expansive scripting interface, so you can create completely custom visual effects using JavaScript.

For more information, check out the Supernode tutorial in Chapter 5 of this book, *Custom rendering*.

Drawing custom shapes

Custom shapes are often needed in compositing. In keying, a custom shape can be used to complement a mask that was created by keying to mask out unwanted objects that were not picked up by the keyer because of their color (this is called a garbage matte).

In color correction, custom shapes are very useful to control the area within the image where a particular correction should be applied (this is sometimes called a “power window”, as some color correction systems refer to masks as windows).

If you’re creating visual looks, shapes are similarly useful to control the part of the image where a particular visual manipulation should be strongest. For example, a “vignette”-style effect would require the edges of the image to be darkened or blurred while the center of the image stays in focus. This can be easily accomplished in Conduit with a custom shape that is blurred, then used as the mask for the darkening effect.

Creating shapes happens using the *Shapes* node in the Conduit Editor (i.e. within a *Conduit Effect* node widget).

A single Shapes node can contain any number of shapes, and a shape can contain any number of control points. The node has direct on-screen controls for drawing shapes and editing the control points that make up the shape.

The editing controls for the Shapes node can be active while viewing the output of another node. This is a convenient way to manipulate a shape that’s generated in one part of the effect and then processed, e.g. used as the

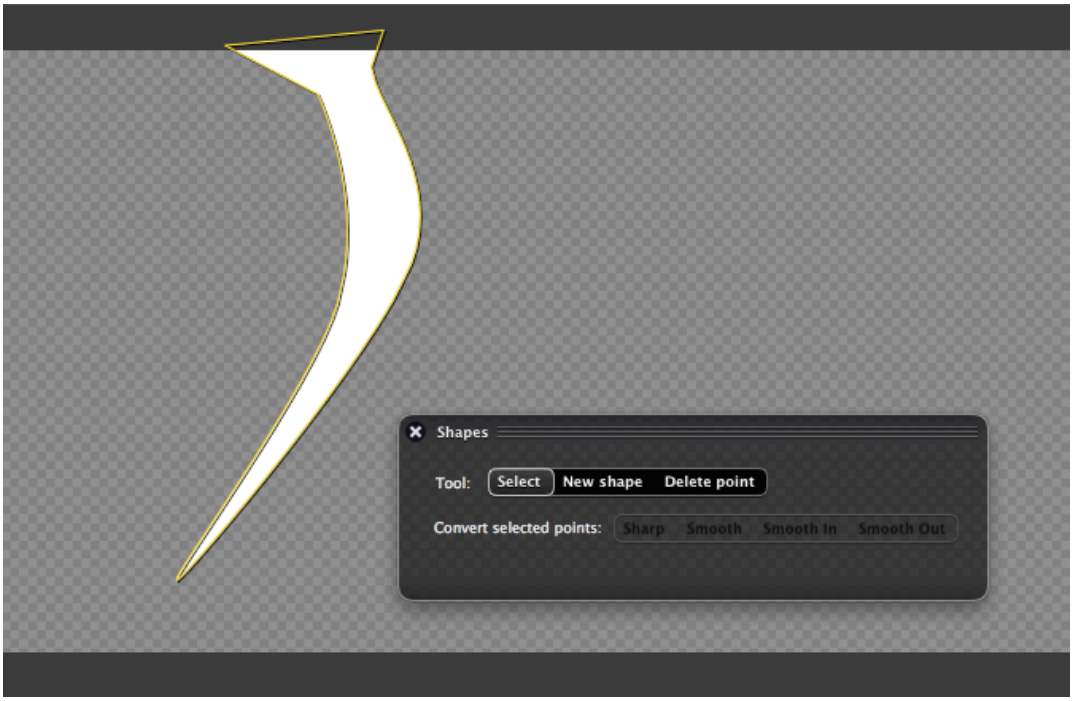
mask for a color effect. You can view the output of the color effect while dragging the shape points.

This tutorial offers a quick overview of how to create shapes; edit shape points; work with smooth curves; view shape editing controls in another context; and export shapes to a text file.

Creating shapes

The Shapes node resides in the *Image* category. To create a new node, drag&drop it from the node box in the Conduit Editor's top-left corner, or right-click on the composition area to bring up the list of nodes.

Once you've created a Shapes node, select it by clicking. This will pop up the Controls window over the Viewer. It's a translucent floating window that shows the tools pertinent to the selected node:



To draw a shape, select the "New Shape" tool in the Controls window, and start clicking in the Viewer. The new shape is displayed as a yellow outline.

To finish the shape, either double-click in the Viewer, or click on the "Finish shape" button which appears in the Controls window while you're drawing a new shape.

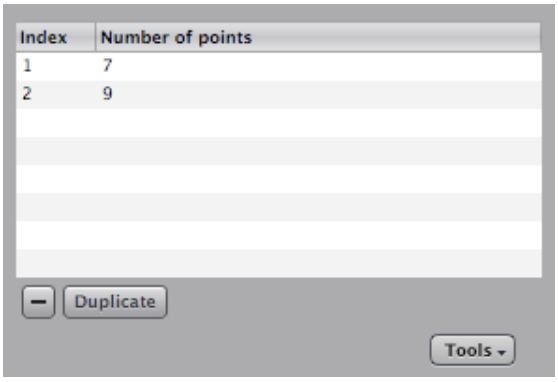
You'll notice that the shape's yellow outline is being displayed over whatever image your current Conduit effect is outputting. To view the actual output of the Shapes node, double-click on it. It should now display the shape in white, with the yellow outline on top of it.

Editing shape points

Now that we have a rudimentary shape, let's modify its points.

To select the shape, switch to the "Select" tool in the Controls window, and click on the shape's yellow outline in the Viewer. The shape will turn white and draggable handles are displayed for each control point.

Alternatively, you can select the shape in the list of shapes that's shown in the node's info area:



To modify the shape, drag the control points in the Viewer.

To delete individual points from the shape, switch to the "Delete point" tool in the Controls window, and click on points in the Viewer.

To delete the entire shape, click on the button labeled with a "minus" sign (shown in the above screenshot).

To add control points to the shape, first select a control point, then click on the "Subdivide selected" button in the Controls window. This will create new control points on both sides of the selected point (in other words, the surrounding edges are split in half). A useful way to think of the Subdivide tool is that it adds detail locally. This is particularly relevant when working with smoothed curves, which will be described next.

Working with smoothed curves

Most vector graphics applications use Bézier curves to represent curved shapes. With Bézier curves, each control point in the shape has two additional control points, "Bézier handles", that determine the curvature of the line. The problem with Béziers is that the relationship between the handles and the resulting curve is not entirely intuitive: the handles do not lie directly on the shape's outline, but instead they behave like attractors that pull the curve towards them. It can be difficult to get Béziers to behave as one intends due to the complex interaction between the actual control points and the handles.

Conduit uses a different approach to curvy shapes. Every control point lies directly on the shape, and there are no handles that would distort the shape from outside. Instead, the curvature of the shape is affected by the surrounding points. This kind of curves are common in modern 3D

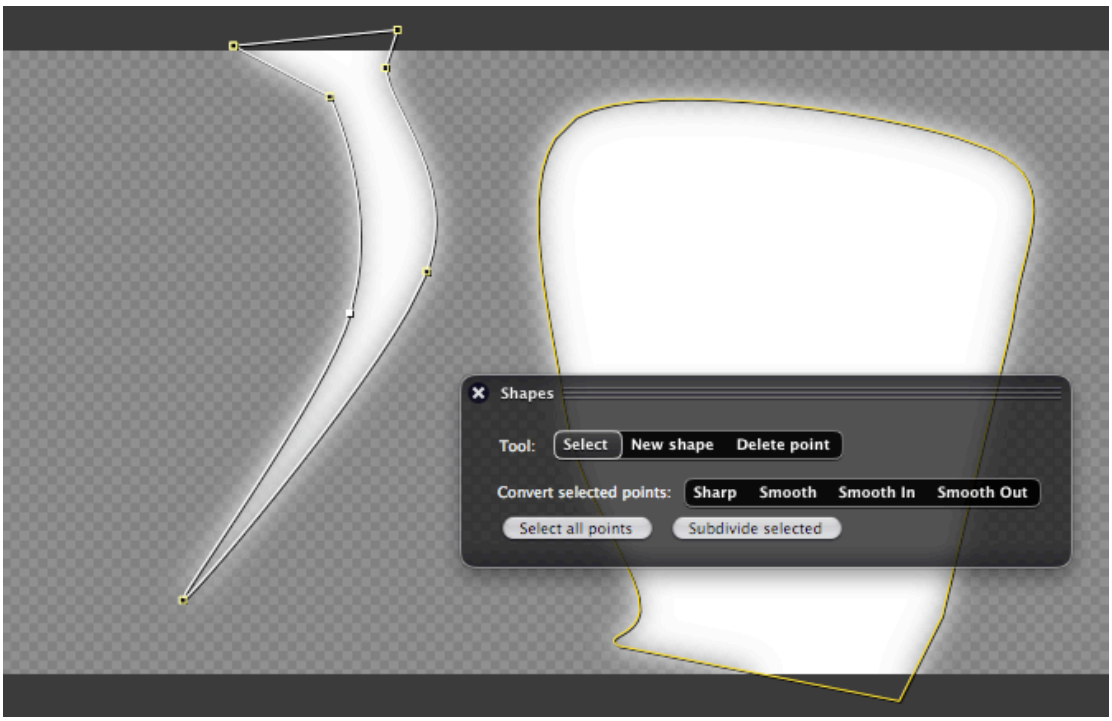
modelling applications: if you've ever worked with subdivision surfaces, you're already familiar with how smooth shapes behave in Conduit.

To smoothen a point, select it in the Viewer, then click on "Smooth" in the Controls window.

The other options available are "Sharp" (which converts the point back to a sharp corner), "Smooth In" (which smoothens the curve before the point, but not after it), and "Smooth Out" (which smoothens the curve after the point, but not before).

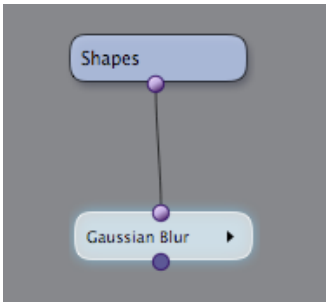
As you create a few smoothed points and drag them around, you'll notice that the curvature changes over a larger area than just the immediate surrounding of the point. This is an intentional feature of smoothed curves. To limit the area that's affected by smoothing, you can simply add control points. The Subdivide tool (described in the previous section) does precisely this.

If you're used to Béziers, smoothed curves may seem to behave non-intuitively at first. One useful approach is to try thinking of shapes in terms of "sculpting" instead of "drawing". Start with the roughest possible outline using a minimum of control points; smooth where necessary; then add detail using the Subdivide tool, and keep adding detail until the shape is fully sculpted.



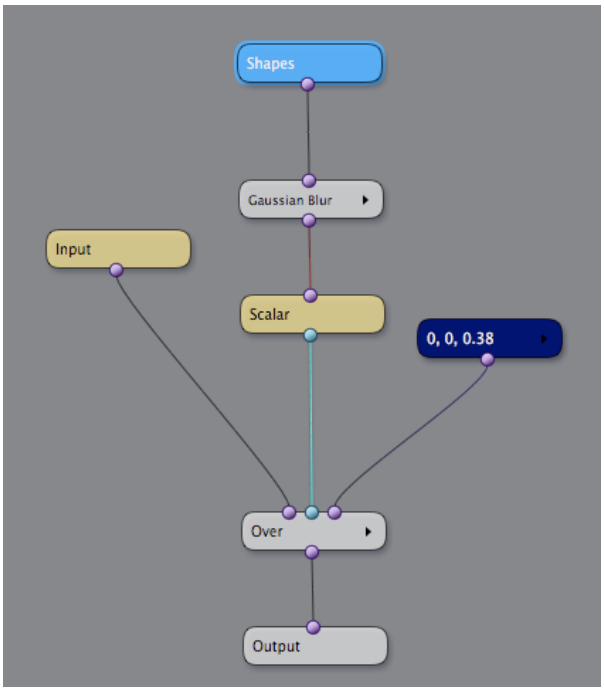
Viewing shape controls in another node's context

Notice how in the above screenshot, the rendered shapes (in white) are blurred. This is achieved simply by adding a Gaussian Blur node below Shapes:



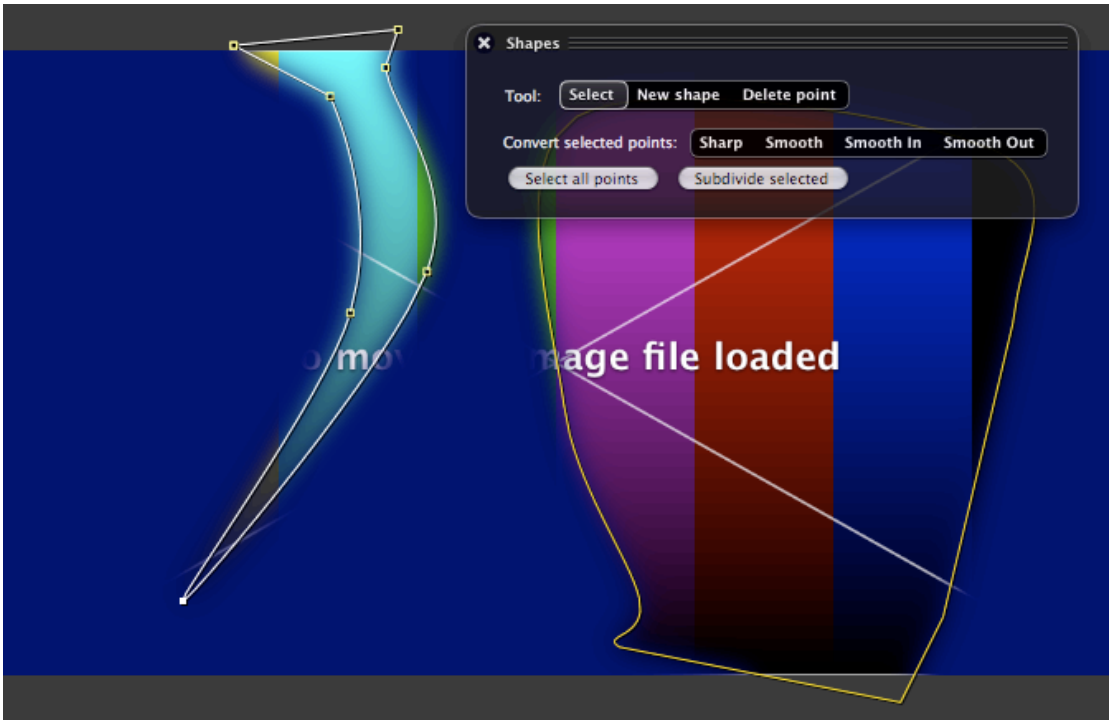
To view the Gaussian Blur's output, double-click it. Then, to view the Shapes node's controls on top of the blurred image, single-click on the Shapes node.

For something a bit more complex, but also more useful, let's apply the blurred shapes as a mask when compositing an image on top of another. Here are the nodes to accomplish that:



The Scalar node is necessary here because the Over node's mask input requires a single-channel image. What "Scalar" does is simply to take the first channel from the given color image, i.e. the red channel. This works just fine for our purposes because the image output by Shapes is monochrome.

Below we can see what the composited result looks like. The Shapes node is selected, so its controls are shown over the result image. (In this example, the "input" image is the default color bars placeholder from PixelConduit.)



Exporting and importing shapes

Shapes created in Conduit can be exported to a text file, and then later imported into another node. This allows shape lists to be stored and edited outside of Conduit.

To access the import / export options, click on the “Tools” menu button in the Shapes node’s info area (in the Conduit Editor).

Conduit uses a JSON-based file format for exported shape lists. JSON is similar to XML in many ways, but much more compact and easier to read and edit. The use of JSON also makes it very easy to process Conduit shape lists in scripting languages like JavaScript and Python.