This opens the door to completely new design workflows. Consider the example of a TV commercial that needs a unique stylized color look as well as overlay graphics (e.g. logos). A graphic designer could use Conduit in Photoshop to design both the visual style and the overlay graphics. She embeds the overlay images inside the .conduit file, so the entire look can be transferred as a single file over to a PixelConduit setup where it's used to preview the graphics while shooting footage. Finally, the same file can be moved to a Final Cut Pro editing station where it is applied to the final edit.

2. Getting started

If you're familiar with video editing applications, you may find that PixelConduit works quite differently than what you're used to. This is because the fundamental metaphor is different.

Video editing software uses a *film-strip metaphor:* your video content is represented as a timeline with frames flowing from left to right. This metaphor originally comes from physical film edit desks. It's very suitable for editing, but doesn't work so well for representing live situations where the duration of your content is not known beforehand.

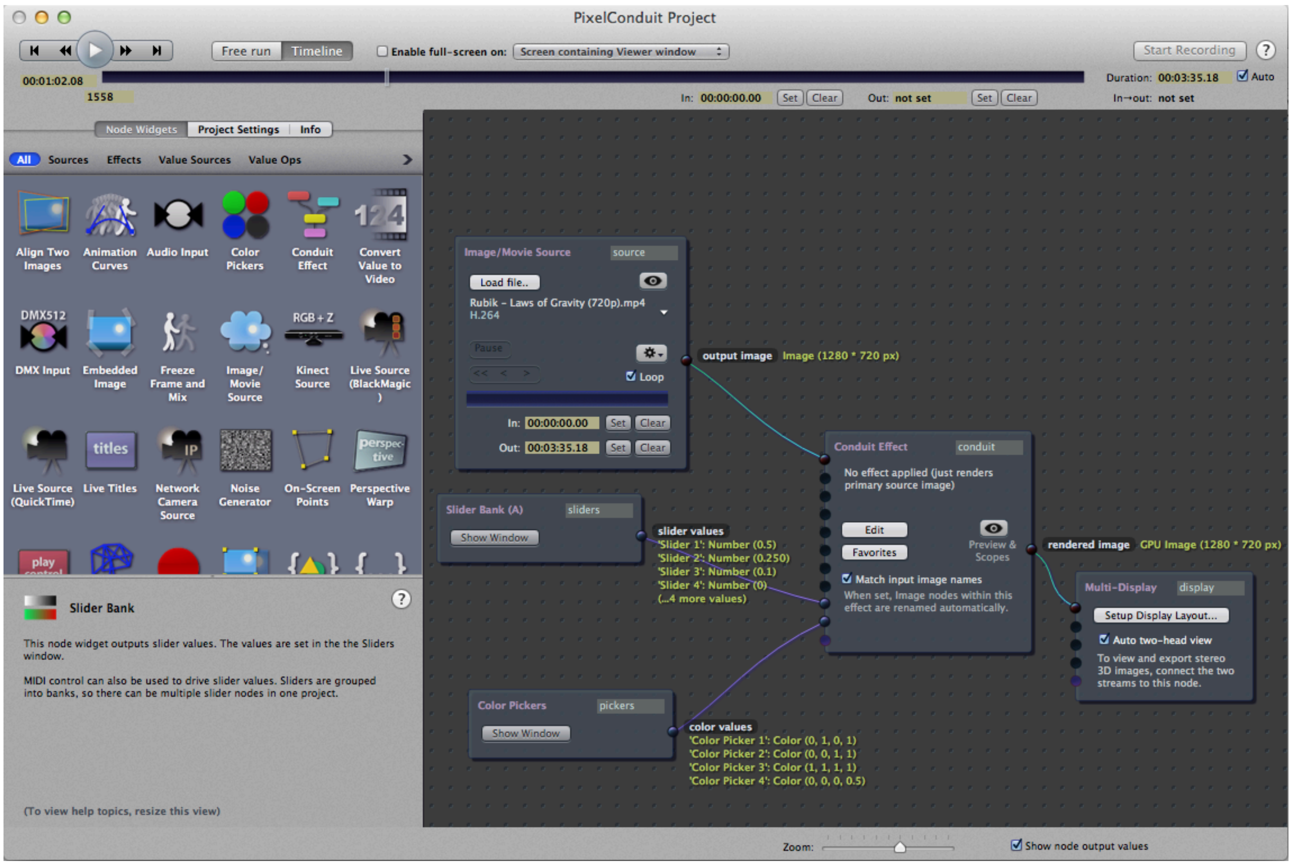
PixelConduit uses a *control room metaphor*. There is a room full of devices, and you decide how they are connected to each other. Some of the devices produce images – similar to a real-world camera or DVD player – and some produce control signals, similar to a lighting control desk. Some devices process video images, like you might use a video mixer in the real world. Finally, some devices output video onto a display or projector, or record them to disk.

The best way to familiarize with these concepts is to dive right in. In this chapter, we'll go over the PixelConduit user interface components first. Then we'll look into some essential settings of a PixelConduit project, and finally the various "devices" that you can use in the project.

PixelConduit user interface

Project window

The Project window is the centerpiece of PixelConduit. It shows an editable overview of all the components that contribute to your project's visual output. These can be on-disk video sources, live video sources, effects, displays, and so on.



This node widget outputs slider values. The values are set in the the Sliders window.

MIDI control can also be used to drive slider values. Sliders are grouped into banks, so there can be multiple slider nodes in one project.

(To view help topics, resize this view)

No effect applied (just renders primary source image)

When set, Image nodes within this effect are renamed automatically.

To view and export stereo 3D images, connect the two streams to this node.

'Slider 1': Number (0.5)
'Slider 2': Number (0.250)
'Slider 3': Number (0.1)
'Slider 4': Number (0)
(...4 more values)

'Color Picker 1': Color (0, 1, 0, 1)
'Color Picker 2': Color (0, 0, 1, 1)
'Color Picker 3': Color (1, 1, 1, 1)
'Color Picker 4': Color (0, 0, 0, 0.5)

These components are represented as **node widgets** in the Project window. They are visual blocks with two distinctive user interface features. Firstly, they enclose a set of controls, such as Play/Pause buttons for a video source.

Secondly, they have input and output ports. Using these ports, you can connect node widgets to each other in order to decide how they interact: for example, to display a video image on screen, you would connect the video source widget's output port into the display widget's input port. This is essentially like connecting a camera to a display using a cable. In PixelConduit, these 'cables' are created by dragging with the mouse from an output port.

The project can also contain components that don't directly modify the output, but which do something useful while the project is running and video is playing. For example, the Rendered Capture node widget can be used to record any video stream to the computer's hard disk.

Node widgets can represent physical devices or interfaces. The Live Capture node represents a video camera connected to the computer; the Sliders node represents the controls in the Sliders window or a MIDI controller; and so on.

This kind of device metaphor can be a useful way of thinking about the project, even for node widgets which don't have a direct physical equivalent in your hardware system. We could imagine a node widget which represents a real-world video mixer. It has eight video inputs and a single video output. To produce an image, the mixer "grabs" eight images from its inputs and renders a composite image which blends those images based on the state of the mixer's controls.

PixelConduit has a node widget that's exactly like this: it's called **Conduit Effect**. Unlike a physical video mixer, the *Conduit Effect* node widget is completely reconfigurable. To “rewire” the mixer’s internal processing, a click on the node widget's Edit button is enough. This opens the Conduit Editor, a rich node-based user interface for designing visual effects. It's exactly the same window that you can find in the Conduit plugins.

This illustrates the relation between PixelConduit and the rest of Conduit Suite. The *Conduit Effect* node widget is essentially the same thing as the Conduit filter applied to a video clip in Final Cut Pro or After Effects. It's the surrounding application environment that's quite different: a sequence in FCP (or a composition in After Effects) is based on layered video tracks, whereas the PixelConduit project is based on interconnected node widgets. The basic interfaces are so different that the applications are complementary; they have quite different strengths and use cases. PixelConduit is not the tool of choice for editing and trimming video clips, whereas FCP or AE are not suitable for dynamic live video applications – that is where PixelConduit shines.

For more information on node widgets, how connections between them work, and what the colors on the connector lines mean, see *Node Widgets in the Project window* in this book.

Project window tabs

The Project window has three tabs: Library, Settings and Info.

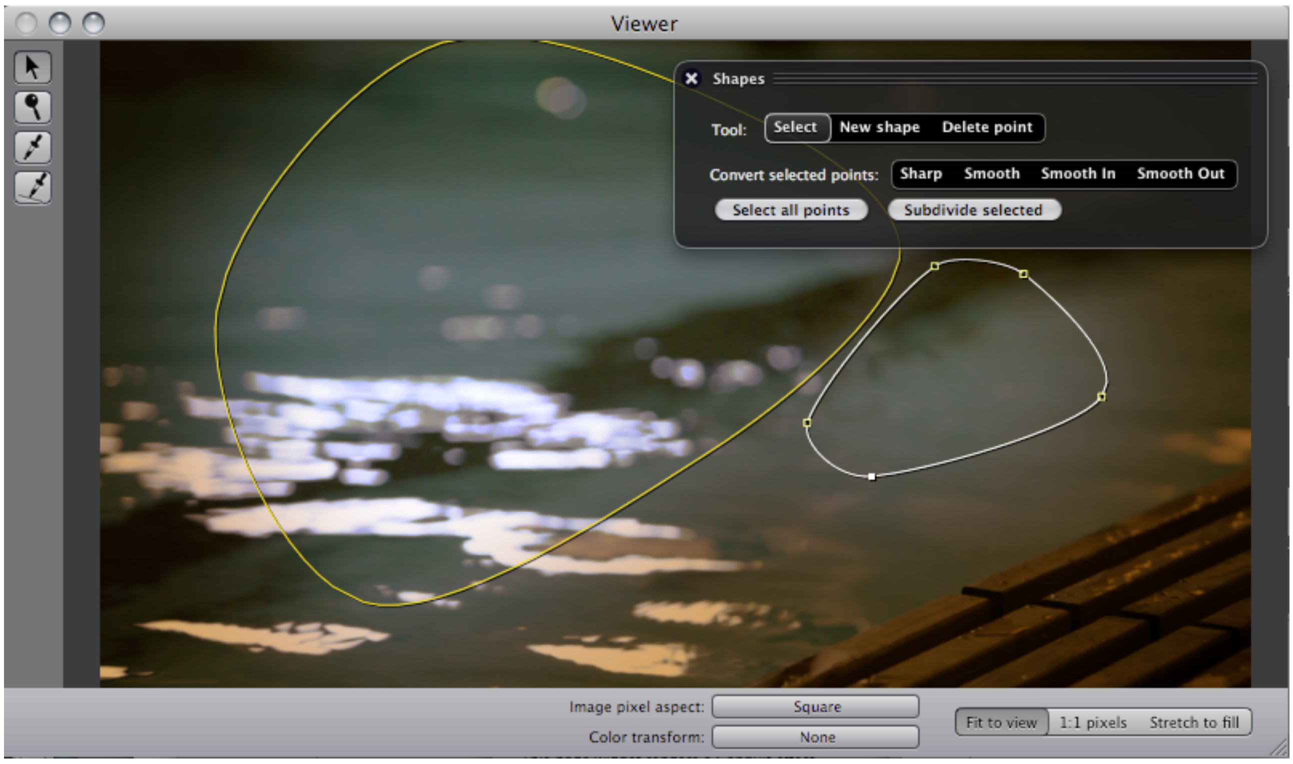
Library contains a list of node widgets available in the system. To view a description of a node widget, click on it. To create a node widget in the project, drag from the Library to the Project node area.

Settings has important parameters that affect the output of the project. For more information, see *Setting up the project*.

Info shows the application version number and information about installed plugins.

Viewer window

The Viewer window shows the project’s final video output. This is determined by whatever is connected into the Display node widget in the Project window’s node area.



By default, PixelConduit provides a **Multi-Display** node in any newly created project. The Multi-Display has multiple inputs and can be configured to display those inputs on screen in various layouts. It also does basic compositing, so you can use it to place one video stream over another; e.g. titles over a background. For more information, see the section *Using the Multi-Display node widget* in this manual.

The Viewer window has the following settings:

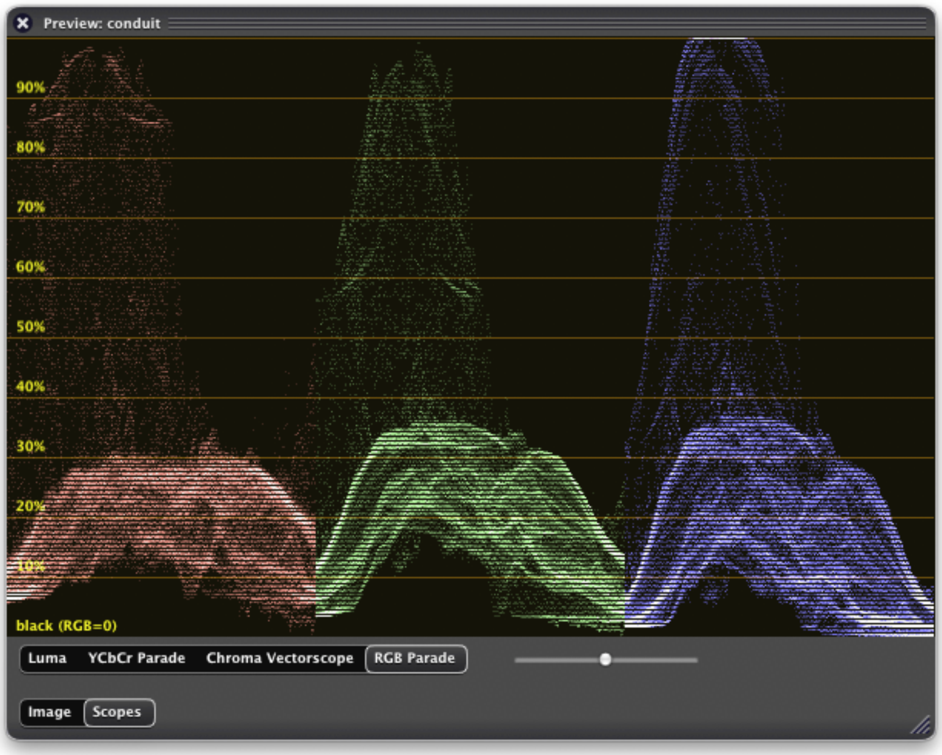
- Image pixel aspect ratio
- Color transform
- Scaling mode (Fit to view / Center 1:1 / Stretch to fill)

Depending on the active node widget, PixelConduit will also display on-screen controls in the Viewer window. For example, a *Conduit Effect* node widget will display shape drawing controls when there's a Shapes node selected in the Conduit Editor window.

Previews and scopes

Node widgets that output a video stream typically also offer a preview window of their own, so you can view their output directly. These include *Movie Source*, the various *Live Source* nodes and *Conduit Effect*. To open the preview, click on the button with the "eye" icon.

Previews are shown in dark floater windows. There's no limit to the number of previews that can be simultaneously open.



The preview window has two display modes: Image and Scopes. Scopes provides a complete set of video scope displays, helpful in analyzing the image's luminance and color distribution. The following video scope displays are available:

* Luma

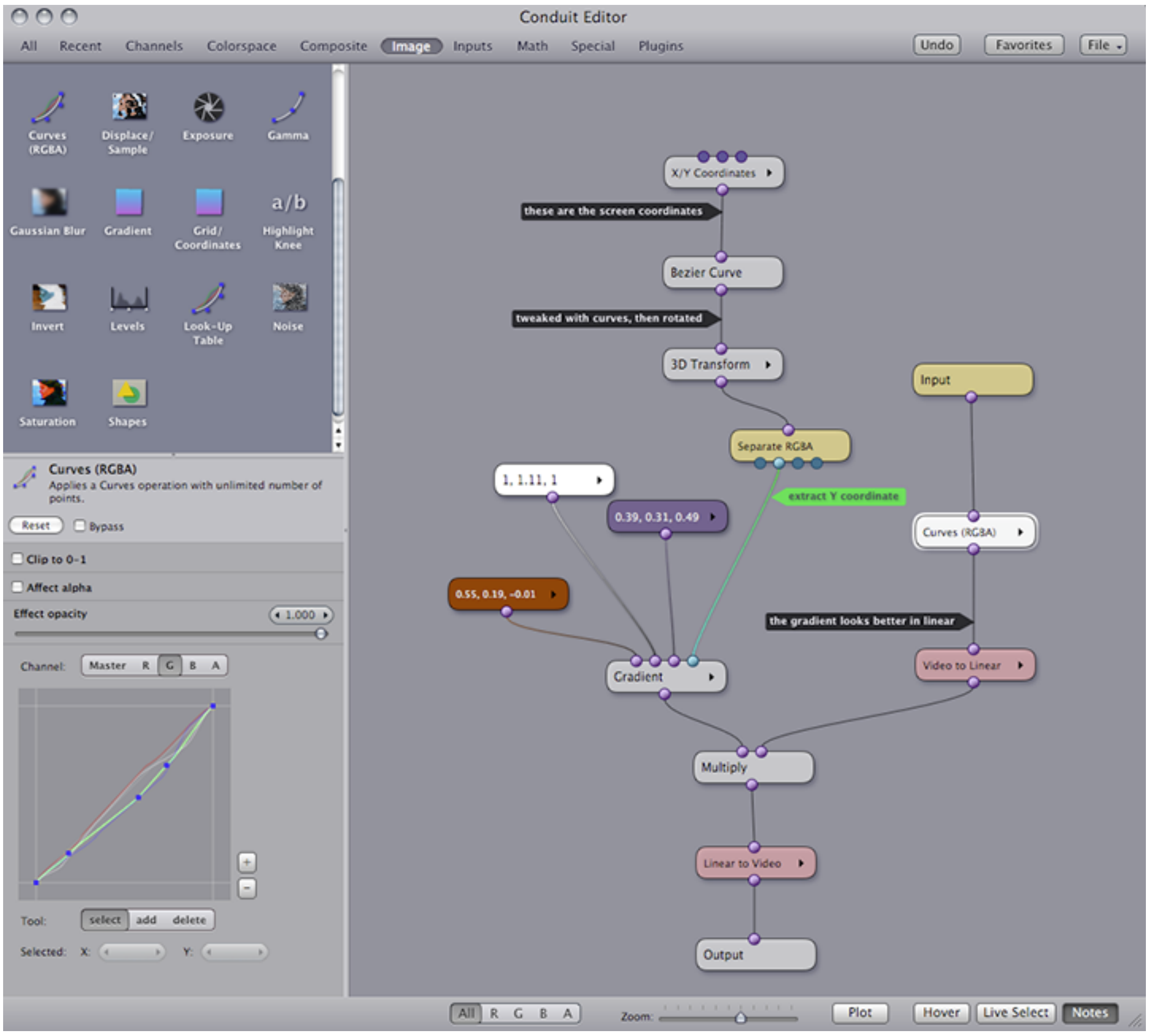
* Y'CbCr Parade

* Chroma Vectorscope

* RGB Parade

Conduit Editor window

The Conduit Editor is a graphical interface for creating realtime visual effects. Nodes represent image processing operations. The editor window opens by clicking on the "Edit" button in a Conduit Effect node widget.

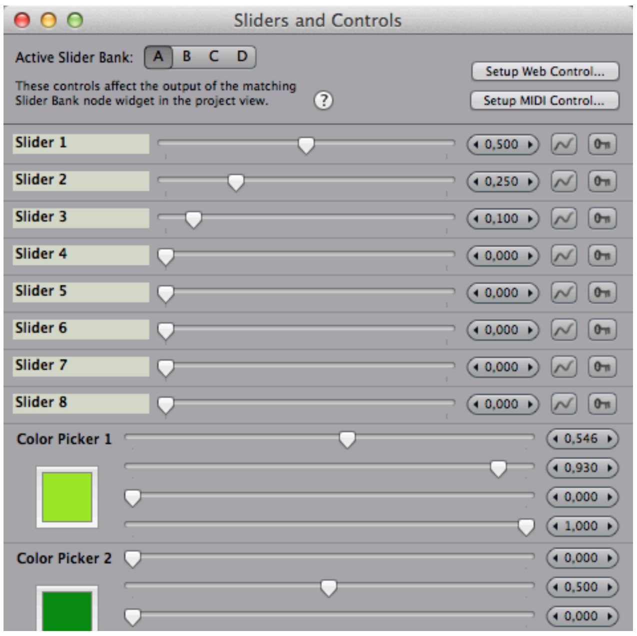


The Conduit Editor is also available in the Conduit plugins for Final Cut Studio, After Effects and Photoshop. Effects can be transferred between all Conduit versions using the ".conduit" file format.

For more information on Conduit Editor, see *Designing effects: the Conduit Editor*.

Sliders and Controls window

Sliders and color pickers provide a simple interface for controlling what your project is doing. They are available in a window named Sliders and Controls:


Sliders and Controls

Active Slider Bank: A B C D

These controls affect the output of the matching Slider Bank node widget in the project view.

Setup Web Control...

Setup MIDI Control...

Slider 1 0,500
Slider 2 0,250
Slider 3 0,100
Slider 4 0,000
Slider 5 0,000
Slider 6 0,000
Slider 7 0,000
Slider 8 0,000

Color Picker 1 0,546 0,930 0,000 1,000

Color Picker 2 0,000 0,500 0,000

Sliders can also be controlled by an external device or remotely. These settings are available in the top-right corner of the Sliders and Controls window.

There are two control methods:

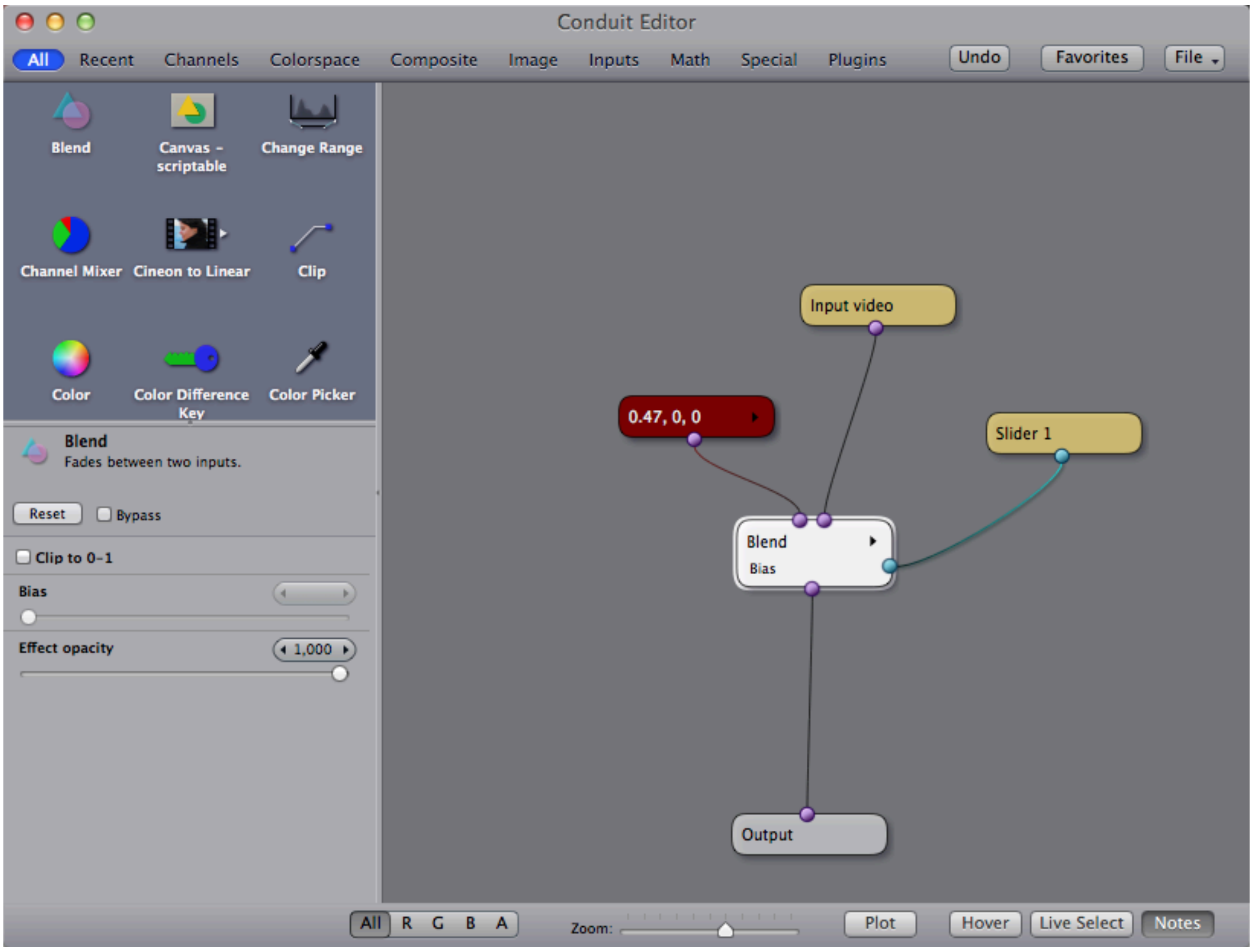
- MIDI control for hardware sliders that use the MIDI standard for audio data.
- Web control for access over a local network. This makes PixelConduit act like a web server into which you can connect from e.g. a smartphone or tablet. The web control interface provides sliders and other controls that can be used to remotely drive your project.

You can rename sliders by typing in the pale yellow box which displays “Slider N” by default.

It’s important to understand that the purpose of the sliders is entirely programmable. "Slider 1" doesn't represent anything in particular until it's given a meaning within your project.

In practice, this happens when a node widget uses the slider value in some way. This is done within the Project window by connecting node widgets. Just like video inputs are represented by Movie or Live Sources, there is a type of node widget that represents the sliders and color pickers. This node widget is called **Slider Bank**.

By default, there is a Slider Bank node in the project and it is connected to a Conduit Effect node widget. Within the Conduit Effect, the slider values can be accessed as Slider nodes in the Conduit Editor:



In the above screenshot, the slider value is used to mix between a red color and a video image.

There's no limit to what you can do with sliders in the Conduit Editor. For example, slider values can be attached to image processing values like saturation; used in math operations; or combined with pixel values so that the act like color components.

Notes on using sliders in the project

Although the Conduit Effect is the default "destination" of slider/picker values, you can also direct slider data to other node widgets. Typically this might a Script Widget that processes the slider values somehow, or combines the slider values with some other data.

The sliders are primarily a live control tool, but it's also possible to control the slider values using keyframed animation. Next to each sliders is a button with a 'curve' icon; click on this button to edit keyframes for the slider.

Note that there are many ways to create animation in PixelConduit, and keyframing sliders is just one possibility. Depending on your usage case, other methods may be more suitable. See section *Creating animation in PixelConduit* for more information on animation.

Project Script Editor window

Using the script editor window you can write JavaScript programs to control node widgets.

For most types of node widgets, scripting is entirely optional and can be used to provide advanced behavior. (For example, you could write a script to make a Movie Source node jump to a new random position every three seconds.)

A few node widgets are completely scripting-oriented and require some programming to be useful, such as *Script Widget*. These widgets can be powerful tools even when you don't intend to write programs yourself: useful scripts can be loaded from disk or copied from a web page.

Other windows

Application add-ons can create their own windows within the PixelConduit interface. You'll find these in the Tools menu. If the Stage Tools add-on is installed, it will add several items to the Tools menu.

Setting up the project

The project in PixelConduit is built from node widgets that determine what gets rendered. This offers a great deal of flexibility because node widgets can be added, deleted, reconfigured and interconnected. Very little in PixelConduit is fixed in such a way that it couldn't be customized to suit the user's needs.

There are a handful of fundamental settings and modes that affect the entire project, rather than individual nodes. This section of the User's Guide explains these important concepts.

Project settings

These settings are located in the Project window under the Project settings tab.

Default render resolution determines the resolution at which effects are rendered. You can freely mix video clips and live sources regardless of size and depth in PixelConduit. Only when compositing the application needs to make a decision about the output resolution. If you don’t have any effects, the video streams will be processed at their original resolution -- hence this setting currently only affects Conduit Effect node widgets.

Render quality contains two settings that affect the quality vs. speed of accelerated rendering.

By default, Conduit always renders at floating-point quality, so you never need to worry about artifacts introduced by insufficient color depth (such as clipping and banding). Sometimes the effects being applied are so simple that you don't need full precision, and it's acceptable to render using regular

8 bits per channel precision. This would be the case if you're only doing crossfades and simple color corrections, for example. If you know that you don't need full precision, enable Prefer speed over color precision.

PixelConduit will synchronize its output to the display to avoid artifacts like tearing. However, this synchronization may cause delays, as the system must sometimes wait until a display cycle is complete. If you want always the fastest rendering and lowest latency without care about artifacts, enable Prefer speed over display integrity.

Interface timebase determines the frame rate used by the PixelConduit interface. PixelConduit allows mixing video clips with different frame rates in the same project. This setting is thus necessary to control how time codes are presented in the user interface. It also affects other user interface functionality that requuch as “Frame forward” and “Frame backward” buttons.

There is an “Auto” mode: when it is enabled, the interface timebase will be set automatically when a movie is loaded using the Movie Source node widget. (E.g. if the loaded movie has a frame rate of 24 fps, the interface timebase will be set accordingly.)

Auto mode is on by default. Unless you work with mixed frame rates, you’ll generally want to leave it this way.

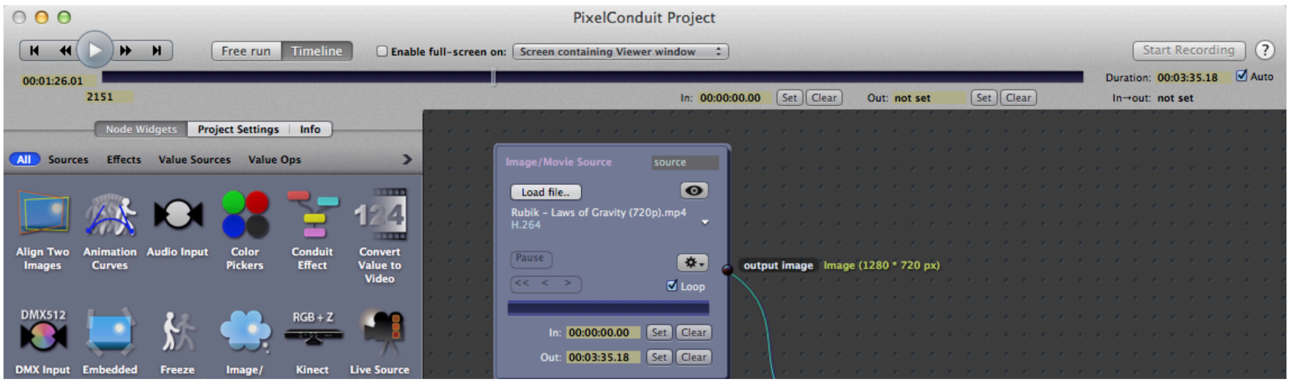
The Project settings tab also contains Capture session settings. These determine the folder where PixelConduit will store video recordings. For more information, see *Recording video to disk*.

Free Run and Timeline modes

PixelConduit has two clock modes, Free Run and Timeline. They are quite different, and the choice between the two modes depends on your use case.

In **Free Run** mode, there's no fixed duration to the project. When you press "Play", the clock starts running and the node widgets will keep producing output until "Stop" is pressed. This mode is ideal for dynamic situations where you don't know the exact length of the "show": live video capture, performance, installations...

In **Timeline** mode, the application behaves like a traditional video compositor. The project has a specific duration, and play controls become available in the user interface for moving to a specific time within the timeline. You should use this mode if you're working with on-disk video or image files, rather than live sources.



Screenshot of the project in Timeline mode.

Node widgets in the Project view

Node widgets are the visual boxes in the PixelConduit project window. The widgets and their connections determine what your PixelConduit project actually renders. It's important to understand how node widgets can work together. While it would be possible to create a project which contains only a single node – for example, a Movie Source node would be enough if all you need is video playback – most projects are not so simple and require more functionality which can be achieved by combining multiple nodes to work together.

Node widget connection types

There are two important types of connections between node widgets. The first is a **video stream connection**. This is what you'll use to connect a Movie Source node into a Conduit Effect's image input, for example. Ports that support this kind of connection are marked with a shiny black dot.

The second type of connection is a **value connection**, which represents any kind of data that is not a video stream. The *Slider Bank* node widget has a single value output which provides the slider's values as a "list". To make these values available to a *Conduit Effect* node widget, the ports need to be connected accordingly. (When starting a new project in PixelConduit, this connection is already made.)

Why are sliders separate from the effect that you'd control with the slider values? The advantage of separating the *Slider Bank* and *Conduit Effect* node widgets is flexibility; you can direct the sliders also somewhere else, or you can replace the sliders with another data source from something completely else. As an example, there could be a node widget to fetch realtime data from an online source, and another node widget could condense that data into a couple of important numbers. These numbers could then be used instead of the Slider Bank node widget to "drive" the values in a Conduit Effect. (In fact, these node widgets to fetch web data and process it do exist in PixelConduit; they're *Web Data Source* and *Script Widget*.)

To give you an overview of what's happening in the project, the Project view has a mode for displaying the data types and the current content being passed between node widgets. This is enabled by toggling the checkbox "Show node output values" in the bottom-right corner of the Project window. (It's on by default.)

To convert data into a video stream that can be watched, there is a *Convert Value to Video* node which will output a video stream that represents the value data. For most types, this is an image containing a text listing of the data. This can be tested easily by connecting a Slider Bank to Convert Value to Video: the result is essentially a printout of the slider values.

Convert Value to Video is smart enough to recognize if its value input is actually an Image object, in which case it will output that image instead of a text listing of the object. The Image object could be generated by a "*Script Widget*" node based on some data loaded from a web server, for example.

Display

The project always contains a single *Multi-Display* node widget. Any video stream that's connected to this node widget is shown in the Viewer window. If full-screen mode is enabled, the image is also shown there.

The display node provided in PixelConduit is called “multi-display” because it has four inputs. This allows it to be used to create a layout of up to four video streams on screen. Alpha compositing is supported. This makes it convenient for inserting overlay elements that are always placed at a specific location on-screen, such as subtitles or a TV channel logo – or simply doing a quick preview of multiple streams composited on top of each other.

Multi-screen output using Matrox GXM

The Multi-Display node widget can also be used with the Matrox DualHead2Go and TripleHead2Go graphics expansion modules. These devices are a practical and easy way of doing multi-display output from a laptop or iMac computer even if it has only one display port. PixelConduit includes presets for the DualHead/TripleHead monitor layouts, so setting up the multiple displays is a one-click operation.

To configure the display layout, click on the "Setup Screen Layout" button on the Multi-Display node widget.

Other video outputs

Secondary Display can be used to display an extra image in addition to the one shown in the Viewer. The image can be placed anywhere on the screen, or it can be set to fill another screen.

When combined with Matrox GXM multihead units, this node widget allows you to output on up to 9 screens from a single Mac.

On a computer with multiple PCI Express video cards installed, this node widget can be used to render onto a secondary video card. For best results, you should install the most powerful video card as the primary one, and then use the *Secondary Display* node widget to render images onto the other video card(s).

Video sources

The most common type of input source is *Movie/Image Source*. This node widget loads a video file, image sequence or still image from disk, and outputs a frame at a specific position.

The node widget has the following playback controls: Play/Pause (single button), Rewind/Frame Forward/Frame Backward.

Note that the playback controls are only effective when the project is in Free Run mode. In Timeline mode, the node's output is determined entirely by the current timeline position. (This behavior makes sense if you think of the node as being a video clip within a timeline in a video editing application: you want it to output the same frame each time a particular position in the timeline is being rendered. For more information about Free Run vs. Timeline mode, see *Free Run and Timeline modes*.

There is also another kind of image input source, the *Embedded Image* node widget. It only supports still images. Once an image is loaded into Embedded Image, it's stored as part of the project and does not require the original image file anymore.

Another type of input source node widget are generators. These output an image without any external feed. There are a few generators included in PixelConduit. *Test Pattern* outputs a test image which can be chosen from a list of typical patterns such as color bars, gradient color bars and grid. *Noise Generator* outputs video noise. Stage Tools (part of PixelConduit Complete) also includes the *Live Titles* node widget, which can be used to generate complex titles and has a cue list for controlling which title is displayed.

Supported file formats

The *Movie/Image Source* node supports at least the following formats. (*Embedded Image* supports the same still image formats, but not QuickTime video.)

- QuickTime RGB, all codecs (with optional alpha)
- QuickTime "raw YUV"
    - This mode offers best quality with pro video codecs and includes 10-bit support; see *Pro Pixels* in this guide for more information.
- Cineon/DPX image sequences: RGB 8/10/16 bits per channel, YUV 8-bit
    - 10-bit Cineon/DPX files often use a logarithmic color space; to decode this format, you can use the "Cineon to Linear" node within a *Conduit Effect*.

- OpenEXR image sequences: floating point HDR
    - Multiple channels are supported; see *Pro Pixels* in this guide for more information.
- TIFF image sequences: RGB 8/16 bits per channel (with optional alpha)
    - CMYK TIFF images are interpreted as RGBA and will look incorrect; however, you can do a CMYK->RGB conversion using *Conduit Effect* to fix this.
- PNG, JPEG, GIF, PSD, and other file formats supported by the operating system

Live Source

There are multiple node widgets for capturing live video.

The *Live Source (QuickTime)* node captures live video from any QuickTime-compatible device such as a video capture card, a webcam or a DV/HDV/AVCHD video camera.

The *Live Source (BlackMagic)* node is what you should use if you have a capture device by BlackMagic Design, e.g. a Decklink, Intensity or UltraStudio card.

The *Live Source (Quicktime)* node can also be set up to record the incoming data to disk. This is particularly useful to record the original data in a compressed format from a camera, e.g. HDV or AVCHD. This functionality is part of Capture Tools which is included in PixelConduit Complete. For more information, see *Recording video to disk*.

Effects

Conduit Effect is the most common type of effect node. The effect that is produced by this node widget is determined by the *"conduit"*; that is, an effect setup that you can edit in Conduit Editor. For more information, see *Designing effects: the Conduit Editor*.

Scripted Effect is a node widget that renders a completely custom effect programmed in JavaScript. There are some examples of how to use this node included in the default templates as part of PixelConduit. They are shown in the start-up screen.

Script Canvas 2D Effect is similar to Scripted Effect, except it provides a 2D Canvas by default, so it's an easy starting point for anything where you want to render 2D graphics or text.

Processing Effect is similar to *Script Canvas 2D Effect* except that it is programmed in the Processing language instead of JavaScript. Note that the implementation of the Processing language doesn't contain all the features available in *Scripted Effect*, for example there's no equivalent of the Surface 3D JavaScript API for accelerated graphics rendering. Therefore Processing Effect is primarily useful if you already know Processing and want to get started quickly, or if you have code in Processing format that you'd like to reuse.

Capturing video

Rendered Capture writes the incoming video stream to disk. The video is written as an image sequence. You can choose the image format (e.g. DPX or PNG) from the node widget's settings. This functionality is part of Capture Tools which is included in PixelConduit Complete.

Note that PixelConduit also allows the original video stream to be recorded from a camera or other capture device (or indeed multiple devices simultaneously). If you need to record non-video data such as timecode and audio, this is the recommended way to keep the data together. You can always store your effect setups separately from the recorded streams and use PixelConduit later to composite the effects.

Stop-Motion Capture will capture single frames from a video stream. This is useful for creating stop-motion animation or timelapse videos.

This node widget can be programmed to take pictures at a regular interval, for example every 5 seconds. The stills are saved in a new folder under the project's current capture session – see *Recording to disk.*

To ensure the best image quality, the still pictures will be saved in a format that most closely matches the video source. Often this results in DPX images being saved. You can use PixelConduit to convert the recorded DPX image sequences to another format like QuickTime.

Other node widgets

- *Animation Curves* outputs values determined by keyframe animation curves. See chapter *Creating animation in PixelConduit.*

- *Color Pickers* outputs color picker values as set in the Sliders and Color Pickers window.

- *Slider Bank* outputs slider values as set in the Sliders and Color Pickers window.

- *Convert Value to Video* – see explanation in *Node widget connection types* above.

- *On-Screen Points* outputs four point values. These points can be picked and modified visually in the Viewer window.

- *Script Widget* – see section *Scripting the PixelConduit project.*

- *Switch Values* can be used to change the order of a list of values; for example, to make "Slider 4" be the first value in the list.

- *Value Printer* displays the incoming input value. This can be helpful in figuring out what's happening within the project. The value is displayed in the JavaScript Object Notation (JSON) format, so the printed values can be copied directly into a JavaScript program.

- *Web Data Source* loads the contents of a web page. This can be used to load text, images or any other kind of content from a web source. The output is a ByteBuffer object. You can use a Script Widget node to convert the object into another type (for example String or Image).

Stage Tools, an add-on to PixelConduit that's included in PixelConduit Complete, contains a number of additional node widgets for hardware input, perspective correction, triggering clips and events, and creating subtitles. For more information, see *Stage Tools Overview*.

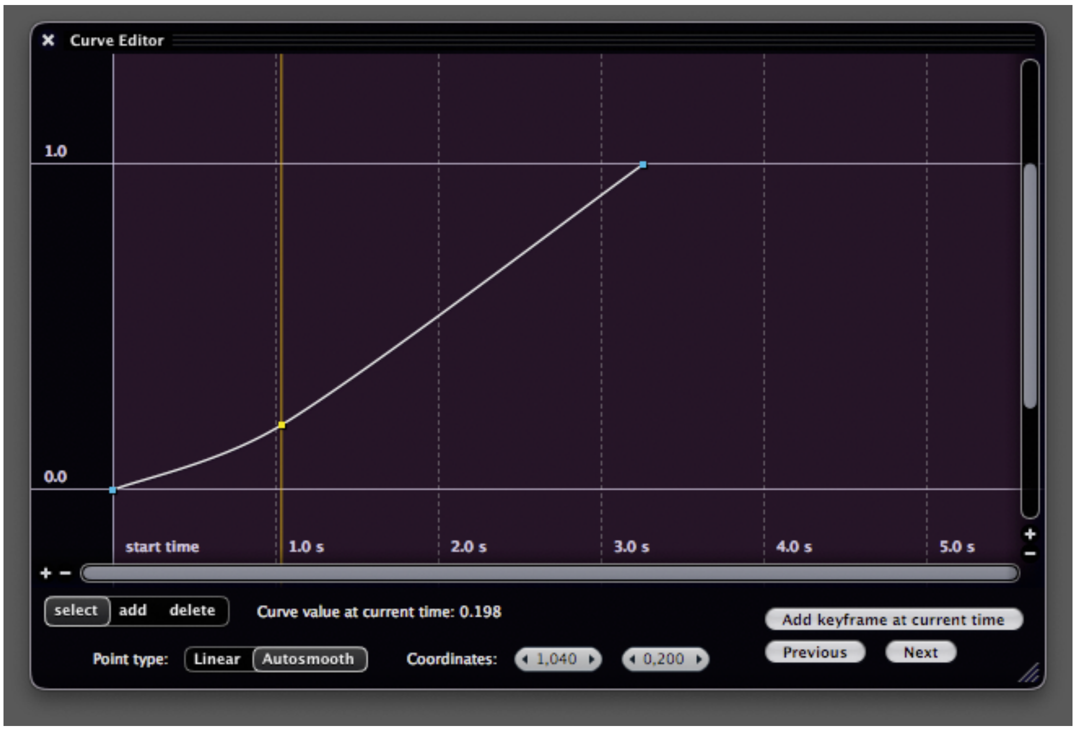
Creating animation in PixelConduit

There are several ways to create animation in PixelConduit. The method to choose depends on the use case, and of course user preference.

Keyframe animation

The method familiar from many other video applications is keyframe animation, in which key values are set at specific times, and the application takes care of interpolating values for the majority of frames that fall between the key frames. Keyframe animation is usually controlled with curves that offer a visual representation of how the values change over time.

PixelConduit offers keyframe animation in two node widgets: *Slider Bank* and *Animation Curves*. Both use the same interface for creating and editing keyframes, the Curve Editor window:



Curve value at current time: 0.198

Point type: Autosmooth

Coordinates: 1,040 0,200

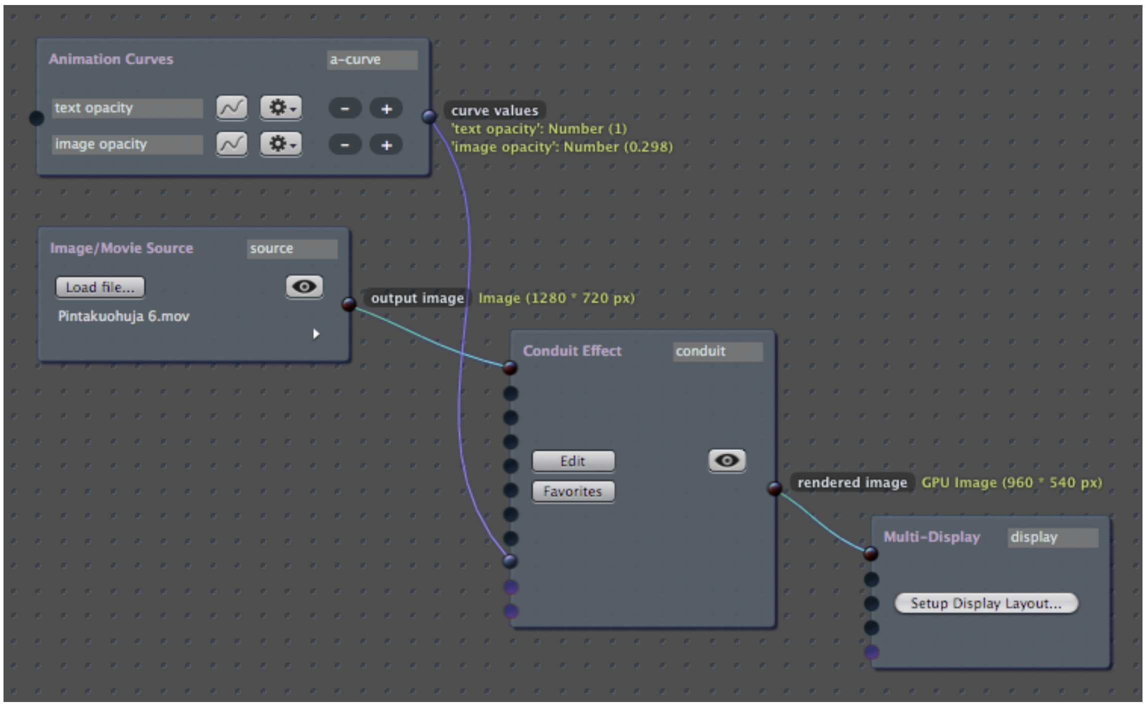
When you set keyframes on a Slider Bank node widget, they will override the slider values that you would otherwise enter manually using the controls in the Sliders window (or using a hardware device like a MIDI controller). This is the easiest way to set keyframes, because you can preview the animation using the slider controls, then "set it in stone" by creating keyframes.

(Note that this method requires that your project is in Timeline mode. If you need to create keyframes in Free Run mode, see below.)

To create slider keyframes, click on the button with the “curve” icon next to the slider's input field (in the Sliders window). This opens the Curve Editor window.

In Free Run mode, it's not possible to set slider keyframes. This is because sliders in typical Free Run projects are controlled by user interaction (either the on-screen controls or through a MIDI connection), and enabling keyframes would obstruct the expected interaction.

Instead, the *Animation Curves* node widget can be used to create keyframes that are computed in Free Run mode:



An *Animation Curves* node widget can contain as many curves as you need. The node has a single output port, through which the current values for all the curves are output as a list. In other words, it works just like the Slider Bank node, and can be used in its place.

Each animation curve has options that can be accessed by clicking on the Tools button – the "gear" icon in the above screenshot next to the names of the animation curves.

These options are:

- Loop mode: Play once / Loop. If "play once" is set, the animation curve's value remains at its last value after the curve's last keyframe has been reached.
- File import/export. The animation curve can be saved as a plain-text file, and loaded back to Conduit.

There is a tutorial called *Drowned World* available in this book that goes into further detail about using keyframes for animation and compositing.

Procedural animation

Animation can also be created procedurally, that is, by writing a program that renders animated graphics. Conduit offers an extensive programming interface that's accessible through JavaScript. Thanks to Conduit's advanced JavaScript engine, it's possible to do realtime and GPU-accelerated rendering in JavaScript programs.

A good place to start learning about graphics programming in Conduit are the scripting-related tutorials in this guide.

PixelConduit also includes another graphics programming language, the popular *Processing*. The Processing website describes it as follows:

> "Processing is an open source programming language and environment for people who want to create images, animations, and interactions. Initially developed to serve as a software sketchbook and to teach fundamentals of computer programming within a visual context, Processing also has evolved into a tool for generating finished professional work. Today, tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning, prototyping, and production."
>
> *(http://processing.org)*

To use Processing, create a *Processing Effect* node widget in the PixelConduit project.

Note that the implementation of the Processing language doesn't cover all the features of PixelConduit, for example there's no equivalent of the Surface 3D JavaScript API for accelerated graphics rendering. Therefore Processing Effect is primarily useful if you already know Processing and want to get started quickly, or if you have code in Processing format that you'd like to reuse. If you want to create 2D graphics, also look into the *Script Canvas 2D Effect* node widget which uses the native JavaScript Canvas programming interface.

Cue-based animation

You can create animation based on live cues and events using *Stage Tools*, an add-on to PixelConduit. It's a complete toolset for creating live performances, and is included in PixelConduit Complete. The regular PixelConduit contains a limited version of Stage Tools.

Stage Tools makes it possible to create animation through events that are triggered live. Events can be programmed into sequences that can modify almost anything in in the PixelConduit project: animate sliders, load video clips, change Conduit effects, perform JavaScript operations, etc.

For more information, please see *Stage Tools overview*.

Exporting to a movie file or image sequence

To export a movie file, image sequence or still image, choose Export from the File menu. The project needs to be in Timeline mode.

PixelConduit can export to a number of formats, including:

- QuickTime movie, RGB
- QuickTime movie, RGB + alpha
- QuickTime movie from raw YUV input (this is a useful mode if working with raw camera data)
- PNG image sequence (common lossless image format, RGB with optional alpha)
- OpenEXR image sequence (HDR floating point format designed for visual effects work)

- DPX image sequence (pro video standard, supports YUV and 10-bit RGB)
- Web video formats: MPEG-4, Ogg Theora and WebM

Raw YUV modes can be useful if you're working with raw pixel data from a camera. For more information, see section named *Pro Pixels* in chapter *Compositing workflows* of this guide.

Note that you can also export movies using Render Automation. This allows you to create a batch list of jobs to be rendered and the movie files which are to be written. Render Automation is part of PixelConduit Complete. See chapter *Using Render Automation* in this book for more information.

Recording live video

Recording video is included in Capture Tools, a part of PixelConduit Complete. This section provides an overview of the features. For a hands-on tutorial, see the next section *Video capture quick start tutorial.*

The capture session

To record video to disk, you must set up a "capture session". It determines where the recorded video files are stored on disk and how they are named.

The capture session is part of the project's settings. It's located in the Project window under the Project Settings tab.

You must enter a name and a destination folder for the capture. PixelConduit will create a new folder for each capture. If a folder with the given name already exists, the application will append a number to the name, so existing files are never overwritten or modified.

Once the capture session has been set up, the "Start Recording" button at the top of the Project window becomes enabled.

Recording original streams vs. Rendered capture

There are two ways to record live video in PixelConduit.

For setups where the video source is a camera that outputs in a compressed format (e.g. HDV or AVCHD), it is recommended to record the original video stream(s). In this mode, the video data is saved intact in its original format, with no processing done by Conduit. While recording the original video data, you can preview effects using PixelConduit. After the shoot is done, you can deliver both the recorded video files and the effect setups to post-production, where they can be combined using either PixelConduit or one of the Conduit plugins (e.g. within Final Cut Pro or After Effects).

The other way to record video is to use the *Rendered Capture* node widget. This node widget can record anything: you can direct any video stream into it within the Project view, and the video will be recorded to disk as an image sequence. This can be used to record the output of a *Conduit Effect* node widget, for example.

Rendered Capture can be very processing-intensive, so if you don't need to specifically capture the output of an effect, you'll probably want to simply record the original video streams instead.

Recording original video streams

PixelConduit can record the incoming video stream from any live video input, such as a DV/HDV/AVCHD camera or a capture card.

In order to record something, you need to have at least one live video source in the project that supports recording the stream. In PixelConduit 3.0 this capability is available in the *Live Source (QuickTime)* node widget.

By default, recording is not enabled for the *Live Source* node widget. In the Project view, locate the node widget and click on the "Setup recording" button to specify in which format the video data should be written.

Two video formats are supported for recording:

- **QuickTime**. This is exactly the original video data stored as a QuickTime (.mov) file.
- **DPX image sequence**. This format is common in professional video/film post production environments.

Typically you'll want to choose QuickTime because it's the native format for the Mac OS X video capture system, and is compatible with a wide range of software. DPX may be more suitable in professional environments where image sequences are the standard way of transferring files to post-production.

Remember that you can always use PixelConduit to convert from one format to another, so this choice is not final. (For example, if you later discover that you need to have a DPX version of a video that was recorded in QuickTime format, you can load the movie into PixelConduit in Timeline mode and export as a DPX sequence.)

Using Rendered Capture

Rendered Capture is a node widget similar to the display nodes: it has a single input and no outputs. When a video stream is connected to its input, this node widget will write the incoming video to disk as an image sequence. When placed after e.g. a *Conduit Effect*, this can be used to record the output of an effect.

Rendered Capture supports alpha channels and high color depths. For example, you can record a keyed video with its alpha channel intact as a PNG sequence. You can also record 10 bits per channel DPX files at full precision.

Before enabling recording in Rendered Capture, you should specify the destination folder in the project's capture session settings (see above).

Video capture quick start tutorial

This section is a short hands-on tutorial that shows how to set up PixelConduit for capturing live video, applying effects like keying, and recording video to disk. Note that recording video requires the *Capture Tools* add-on (included in PixelConduit Complete).

Setting up live video and a Conduit effect

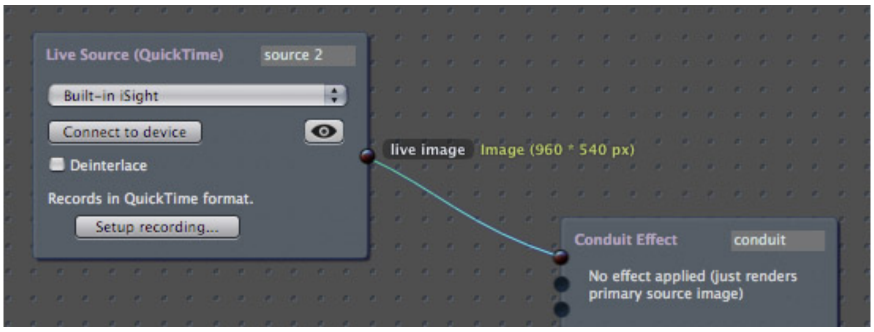
A live video source can be a camera, an internal capture card, an external Thunderbolt capture device, or any other device that has a QuickTime-compatible driver. There is also special support available for certain devices like BlackMagic Design units.

To capture video from a device, create a *Live Source (QuickTime)* node widget. In the Project window, drag the icon (shown on the right) from the node box onto the project view on the right-hand side of the window.

> NOTE: If you have a BlackMagic capture unit (DeckLink, Intensity or UltraStudio), you should use the "Live Source (BlackMagic)" node widget instead. It offers a direct connection to the capture device instead of going through QuickTime, so you get access to possible extra features of the card.
>
> There is also a special interface for the affordable “EasyCap” analog video digitizer which doesn’t have a QuickTime driver. This EasyCap driver can be downloaded from the PixelConduit web site.

We want to process the live video using a Conduit effect, which is like a highly configurable video mixer. It can be helpful to think of the node widgets in the project view as analogous to physical devices like cameras, mixers, and projectors. See the first chapter of this book for more details.

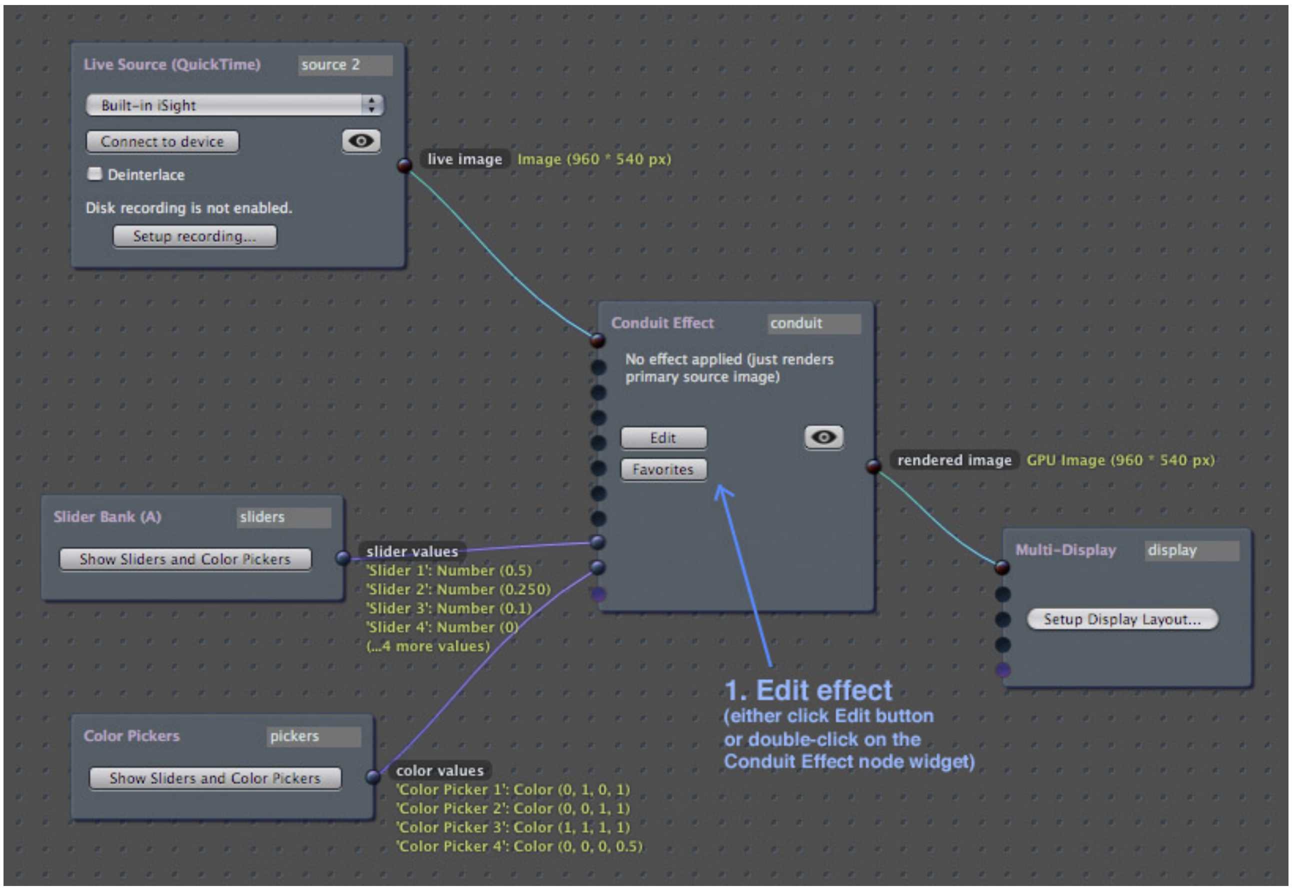
Connect the Live Source node widget's output to the first input on the Conduit Effect node widget:



A newly created project should already contain a Conduit Effect node widget, but if not, you can create one by dragging from the node box.

Now we have the basic setup ready. To view live video, click "Play" in the Project window. The Play function can also be found in the Output menu. The image is shown in the Viewer window.

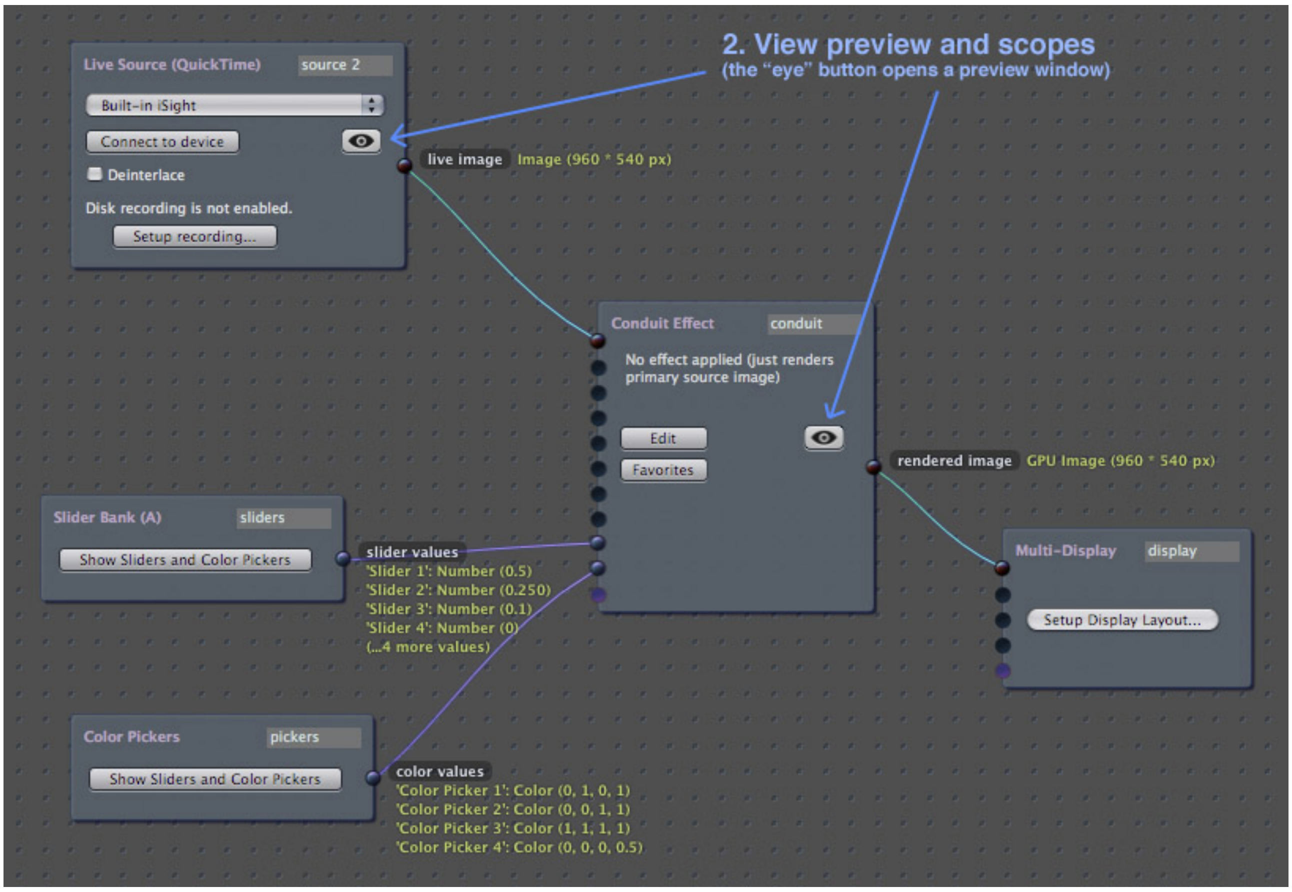
Although we have a Conduit Effect node widget, it doesn't yet do any video processing. To edit the effect, click on the Edit button, or simply double-click on the node widget:



This will open the Conduit Editor window. See Chapter 3 in this book, *Using the Conduit Effect System*, for more information on how to create effects in the Conduit Editor.

Viewing the incoming video image

In PixelConduit, you can open several simultaneous preview windows including image analyzer scopes. Previews are available for live sources, movie sources and Conduit effects. They can be accessed by clicking on the "eye" button in any node widget that has the capability:



Disk recording is not enabled.

Image (960 * 540 px)

No effect applied (just renders primary source image)

GPU Image (960 * 540 px)

'Slider 1': Number (0.5)
'Slider 2': Number (0.250)
'Slider 3': Number (0.1)
'Slider 4': Number (0)
(...4 more values)

'Color Picker 1': Color (0, 1, 0, 1)
'Color Picker 2': Color (0, 0, 1, 1)
'Color Picker 3': Color (1, 1, 1, 1)
'Color Picker 4': Color (0, 0, 0, 0.5)

See *User interface: Previews and Scopes* for more information.

Choosing a recording mode

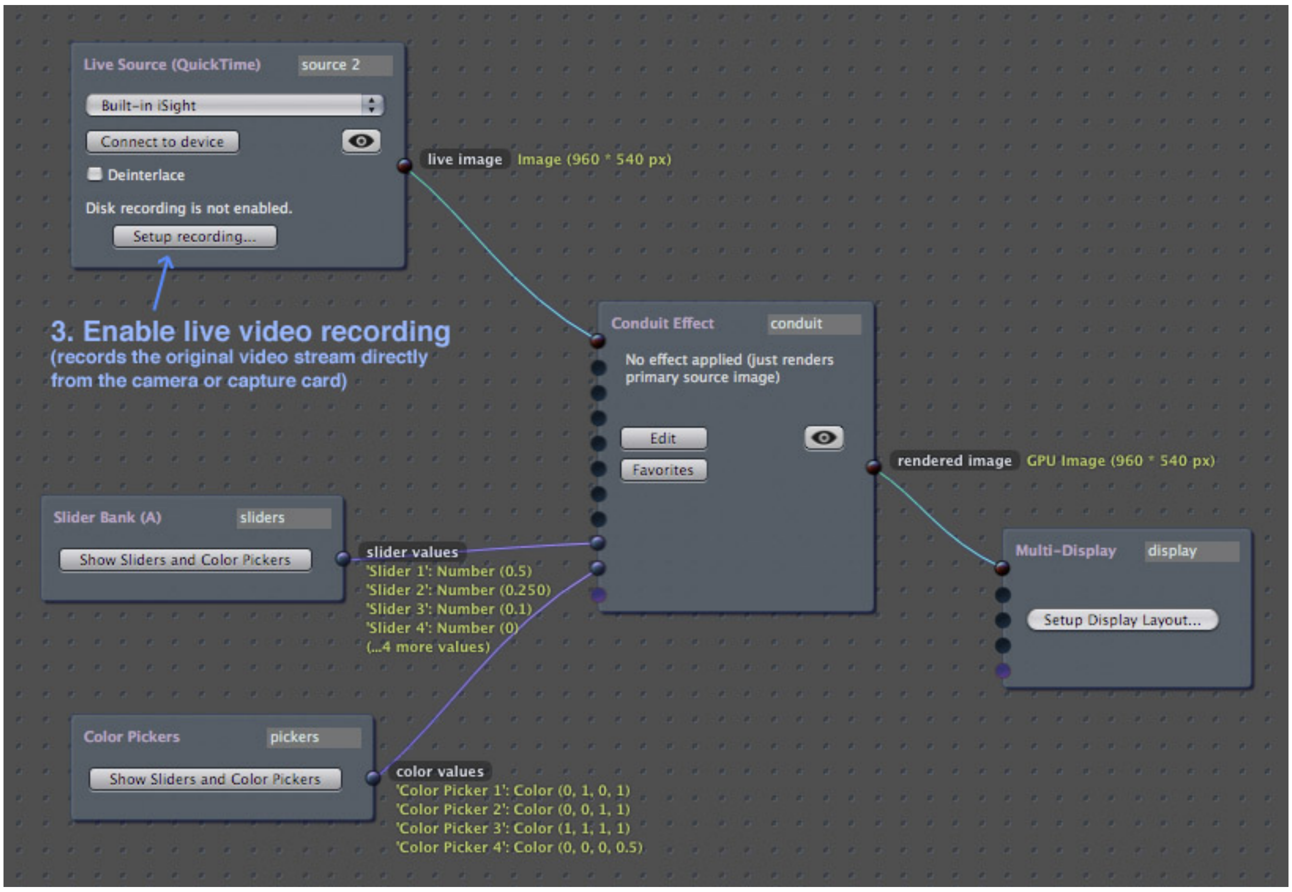
PixelConduit offers two different ways to record video to disk. You can either record the original video stream(s) from the live source(s), or the rendered output from an effect – or even both at once.

However, recording the output of an effect requires much more processing power. For performance reasons, PixelConduit places some limitations on the formats you can use in this mode.

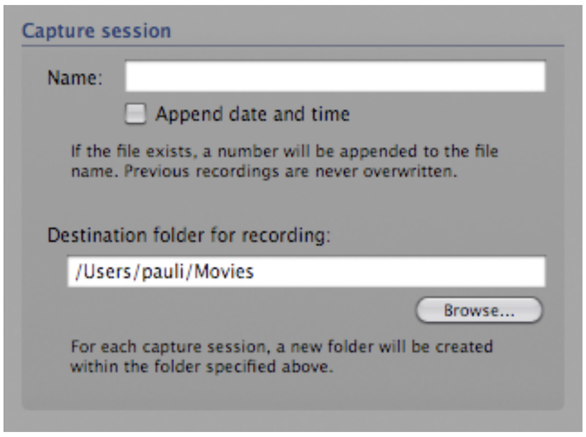
If you need to record effects, there's also an alternative workflow that works very well in many cases. It involves recording the original video streams and rendering the effects later. You can save the effects that are applied to the live video as .conduit files. Later, after recording is completed, the effects are applied during post-production. For this, you can use either PixelConduit itself (in Timeline mode) or one of the Conduit plugins in a host application like After Effects or Final Cut Pro.

This is a convenient workflow because it keeps the recorded video and the effects separate until the last moment, which gives you maximum flexibility in modifying and fine-tuning the effect.

To set up recording, click on the button in the Live Source node widget:


Live Source (QuickTime)
source 2
Built-in iSight
Connect to device
Deinterlace
Disk recording is not enabled.
Setup recording...
3. Enable live video recording
(records the original video stream directly from the camera or capture card)
live image
Image (960 * 540 px)
Conduit Effect
conduit
No effect applied (just renders primary source image)
Edit
Favorites
rendered image
GPU Image (960 * 540 px)
Slider Bank (A)
sliders
Show Sliders and Color Pickers
slider values
'Slider 1': Number (0.5)
'Slider 2': Number (0.250)
'Slider 3': Number (0.1)
'Slider 4': Number (0)
(...4 more values)
Multi-Display
display
Setup Display Layout...
Color Pickers
pickers
Show Sliders and Color Pickers
color values
'Color Picker 1': Color (0, 1, 0, 1)
'Color Picker 2': Color (0, 0, 1, 1)
'Color Picker 3': Color (1, 1, 1, 1)
'Color Picker 4': Color (0, 0, 0, 0.5)

Before recording can begin, you need to give a name for this "capture session". This is used to identify the files as they are recorded. The capture session's name is given in the Project Settings view, in the Project window:



Capture session

Name:

Append date and time

If the file exists, a number will be appended to the file name. Previous recordings are never overwritten.

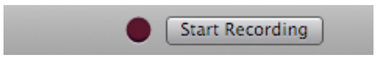
Destination folder for recording:

/Users/pauli/Movies

Browse...

For each capture session, a new folder will be created within the folder specified above.

Recording is now ready, and the "Start Recording" button is enabled at the top of the Project window:



Click on it to start/stop recording at any time. Multiple recordings made within the same capture session will go into separate files.

This tutorial covered recording the original video streams. To record the output of an effect, use the *Rendered Capture* node widget. Its icon is shown on the right.

This node widget works like the Multi-Display node widget that's already in the project. You can connect any input into the node, and it will be recorded.

For more information on Rendered Capture, please see the previous section *Recording live video*.

Performance tuning and troubleshooting

PixelConduit is designed to be a fully realtime graphics system. If you encounter performance issues such as dropped frames or jitter, here are some typical bottlenecks and reasons for degraded performance.

GPU limitations

The computer's Graphics Processing Unit (GPU) is used by PixelConduit for all rendering whenever possible. It is therefore a crucial component when choosing your system setup. The Intel integrated GPUs used on many laptop Macs are substantially worse than the 'discrete' GPUs by NVIDIA or AMD which are used in iMacs and higher-end Mac laptops. In addition to worse performance, Intel integrated GPUs have feature limitations compared to their discrete counterparts. Whenever possible, **choose a system with a discrete GPU.**

Another factor is video memory available to the GPU. If you have many applications open while running PixelConduit, much of the available video memory may be taken by other apps already, which will force the system to move data in and out of video memory. **Close other apps when using PixelConduit for performance critical work.**

The amount of work to be done by the GPU depends primarily on the complexity of your PixelConduit project. Whenever possible, you should **arrange your effects so that they fit into one Conduit Effect node widget**. This is because each Conduit Effect can perform "node fusion" within its own contents: the nodes are compiled together so that they run on the GPU in one shot. This greatly reduces memory traffic compared to the situation where the effect would be split over several Conduit Effect node widgets.

Finally, you should check whether your PixelConduit project's render resolution is appropriate and **reduce the render resolution if possible**. This setting can be found in the Project window's "Project Settings" tab. In many situations it may not make a substantial difference if you bump down the display resolution to e.g. 720p even if the incoming footage is 1080p. Remember that you can always record the original video stream at its original resolution even if your effect previews are rendered at a lower resolution. Similarly, you can work in a lower preview resolution while doing effects and then bump up the resolution for your final export render.

Disk speed limitations

Video playback is often limited by the computer's disk speed. Even if your system's GPU is capable of rendering at a 4K resolution, you won't get that kind of video playback performance if your video files are stored on a regular hard disk. When working with high resolution files or high bit rates (or uncompressed video), you should always first **look into a fast disk system** that can handle the necessary read speed. BlackMagic Design offers a useful and free Disk Test tool that you can use to measure your computer's disk system.

Latency and frame rate mismatch

The previously mentioned limitations are fairly obvious, but latency and mismatching frame rates are a very tricky topic that can't be dealt with by simply adding "raw power".

Latency means the delay inherent in getting an acquired frame onto the display. It is the bane of many realtime video systems. PixelConduit has been designed to minimize latency, but unfortunately the software is only a small part of the complete picture that affects latency. The latency chain for a live video frame may look like this:

Camera sensor
→ camera circuitry
 → computer's capture card hardware buffer
  → capture card's driver in operating system
   → PixelConduit
    → GPU buffer
     → projector used for display (projectors also do internal buffering).

As you can see, PixelConduit is only a fraction in the chain of elements that contribute to the delay in getting a video frame onto the screen. To minimize latency, you must primarily **choose your camera, capture card and display/projector carefully**.

Frame rate mismatch is an issue that arises when the incoming video stream(s) and the output to the display are running at different rates. Typically computer displays and projectors run at approximately 60 Hz whereas incoming video may be 29.97 Hz (in USA), 25 Hz (in Europe) or something else. PixelConduit attempts to detect irregular rates and figure out the best rendering combination, but unfortunately it's not guaranteed to get it right always. (This is a difficult problem for the software developer: in a freely configurable environment like PixelConduit on a Mac, there are no guarantees about how fast the frames come in, what gets rendered during each processing pass, etc. – hence the system must try to adapt to the observed situation.)

If you encounter stutter, here's a tip that could make a difference. In the Project window, **try placing two Conduit Effects** in sequence between the video source node widget and the display node. (These can be empty effects; the important thing is simply that one effect feeds another.) Having the two effect node widgets in sequence will trigger a different rate-detection algorithm in PixelConduit. In some situations this might make the difference. (If this tip does help in your case, please get in touch via the

PixelConduit web site or the Help menu in the app! I'd love hear your experiences in order to decide whether these rate-detection settings should be made explicit within the app.)

Extending PixelConduit

PixelConduit supports plugins, extensions and custom content on many levels. The following is a high-level overview of the possibilities, ordered from the 'easiest' to 'most difficult':

- **Conduit effect favorites**.
  You can save frequently used conduits (Conduit effect setups) as favorites for easy access, and store them in the *.conduit* file format for transferring between systems.
  Building a library of conduits is a great way of keeping your favorite effects and visual looks handy, and you can easily share them with others.

- **Conduit's script plugins.**
  Conduit includes two very flexible nodes for creating custom effects and graphics generators: *Canvas* and *Supernode*. Anything created with these nodes can be packaged as a "script plugin" which behaves just like the built-in nodes in the Conduit Editor. Script plugins can be saved in a binary format, so the data inside can't be accessed anymore.
  Supernode also works as an easy way of packaging a conduit into a single node – this is sometimes called a "macro".
  For more information about using the script nodes and packaging plugins, see the tutorials in Chapter 5 of this book, *Custom Rendering*.

- **Node widget scripts.**
  PixelConduit has several powerful scripting tools that you can use as part of the project: *Script Widget*, *Scripted Effect* and *Processing Effect*.
  Of these, Script Widget outputs a value stream. It can be used to process data and provide custom interfaces in the Project view. The two Effect nodes output an image, and can be used for visualization, effects or generating custom graphics. For more information about using the script nodes and packaging plugins, see section *Scripting the PixelConduit project*.

- **Conduit image filter plugins.**
  These are like traditional filter plugins for applications like Adobe Photoshop and After Effects. They are written in the C language for best performance. To create a native plugin, you need some experience with C programming. The Lacefx API provided by Conduit does make life easier by isolating developers from most of

the differences between Windows and Mac OS, so at least it's not all bad.

- **PixelConduit application add-ons.**
  These are extensions to the entire PixelConduit app. They can do almost anything within the app: add new node widgets, have their own windows, capture shortcut keys... Stage Tools is an example of this class of extension. These plugins are developed in Objective-C.

You can find out more about creating extensions in Chapter 5 of this book, *Custom Rendering*.