MANUAL
V 1.01

The manual itself is still in copy editing stages
and yalls should expect to get pretty regular updated editions
over the next couple of weeks :)

CONTENTS

Section 1 Introduction

What is Gravity Waaaves?

From here on out lets just call it GW! GW can be described as

- a 2 channel video mixer
- an SD video upscaler
- a dual digital video delay processor with delay time up to 4 seconds.
- the flagship VSEJET (Video Synthesis Ecosphere Jetson) video processing tool
- the only video mixer designed specifically for performing with video feedback
- a performance oriented video artist workstation
- and probably more things that all of you will figure out through your personal usage.

Note on Tone

I[1] write manuals in the first person and have a tendancy to use colloquilaisms, idioms, and other such folky vernacular informalisms that some folks find to be incredibly inappropriate for contexts like this. I, on the other hand, feel quite strongly that the kind of dry, detached, second/third person omniscient tone that most manuals use exclusively is radicially inimintable towards the kinds of useful communication of information that any manual should be striving towards.

Consider this a fair warning: if this kind of conversational tone is going to seriously harsh your mellow, I'd recommend just reading the Quickstart portion and then skipping right to the Glossary to read the definitions and check out the graphs and examples over there. I would also ask you to think and consider why exactly it is that you feel that a dry, impersonal, and emotionless tone is somehow inherently superior to a conversational and informal voice for communication, especially communication of information that is often times hidden behind various forms of elitist gate keepers.

How to Use This Manual

There are a couple of different approaches towards reading/writing manuals.

- *The Atari Approach:* give me the bare minimum of facts needed to get into the world and i'll teach myself the rest through trial and error
- *The Engineers Approach:* give me an abstract and context free list of every bit of technical information relevant to operating this as a tool.
- *The Older Sibling Approach:* give me a conversational and context sensitive walkthrough that is centered on achieving goals.

In the interests of covering each of these approaches in a hermetic manner, I've split up this manual into a couple of sections. QUICKSTART (*The Atari Approach*) covers the basics of just identifying hardware stuffs, plugging things in and powering on, and navigating the GUI. WALKTHROUGH (*The Older Sibling Approach*) is an in depth series of excercises that are intended to give you an experiential approach to understanding the most unique aspects of GW's signal flow, working with dual video delay lines, and the other video processing things that are unique to GW. GLOSSARY (*The Engineers Approach*) is a vast list of definitions, diagrams, screenshot examples, and various other reference materials.

1 I am Andrei Jay, the human who designed GW.

I don't particularly recommend reading this manual word by word, start to finish, before ever turning on your GW[2]. The best approach, I think, would be to first read the Quickstart, get GW running, and play around with it a bit to start with. This will give you a good grounding on how the GUI is organized for when you read things a little deeper. At this point you can choose to either just keep the GLOSSARY open and use it to fortify your experiments by looking up definitions whenever you feel like your experiments arent giving you enough information to work on, or work through the WALKTHROUGH and get a detailed introduction to various features, quirks, and techniques.

VSEJET Community Support

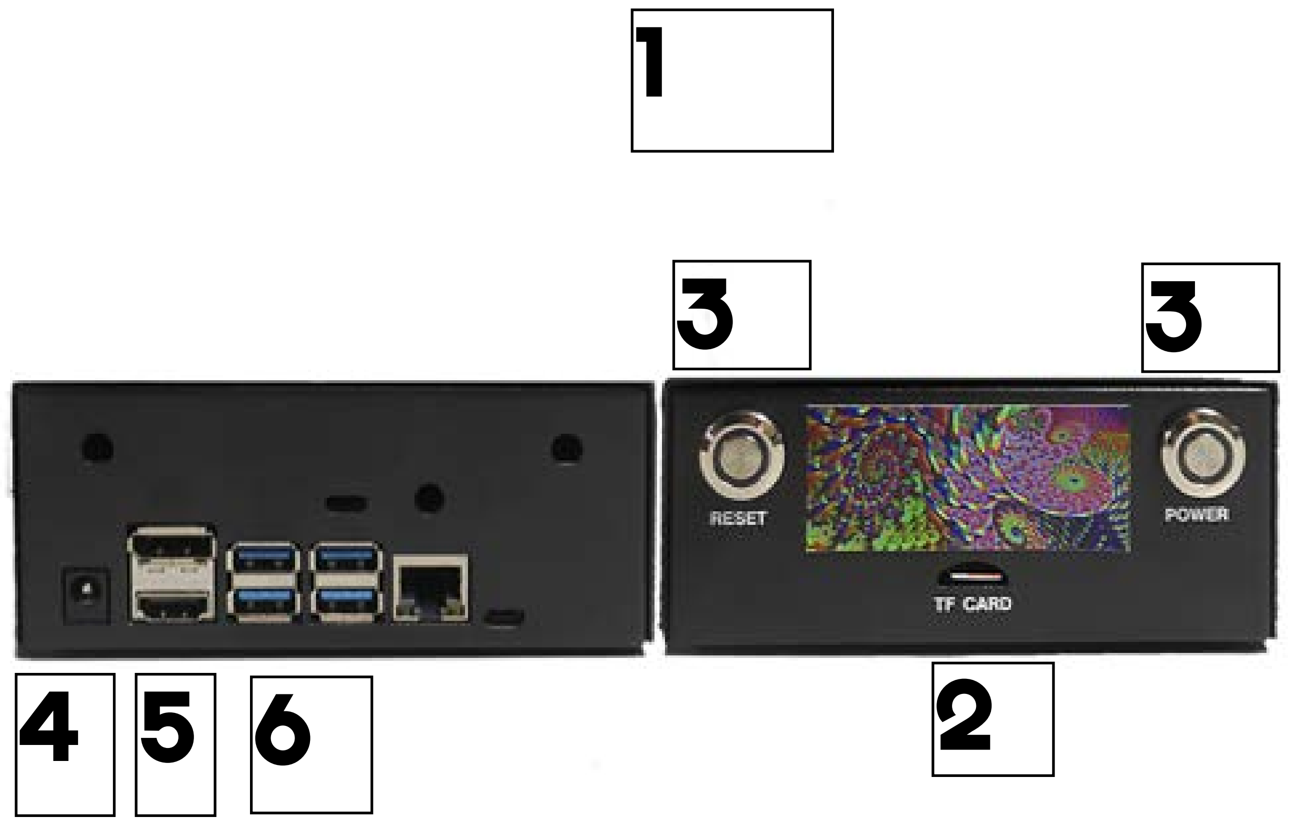
Also please get involved in the various online discussion groups available for working with GW and the VSEJET! If you like message boards, check out scanlines.xyz and you'll find folks working with GW as well as other VSE synths. If you visit andreijaycreativecoding.com you can find many various invite links to join the VSE discord if you are more of a fan of that kind of interactions.

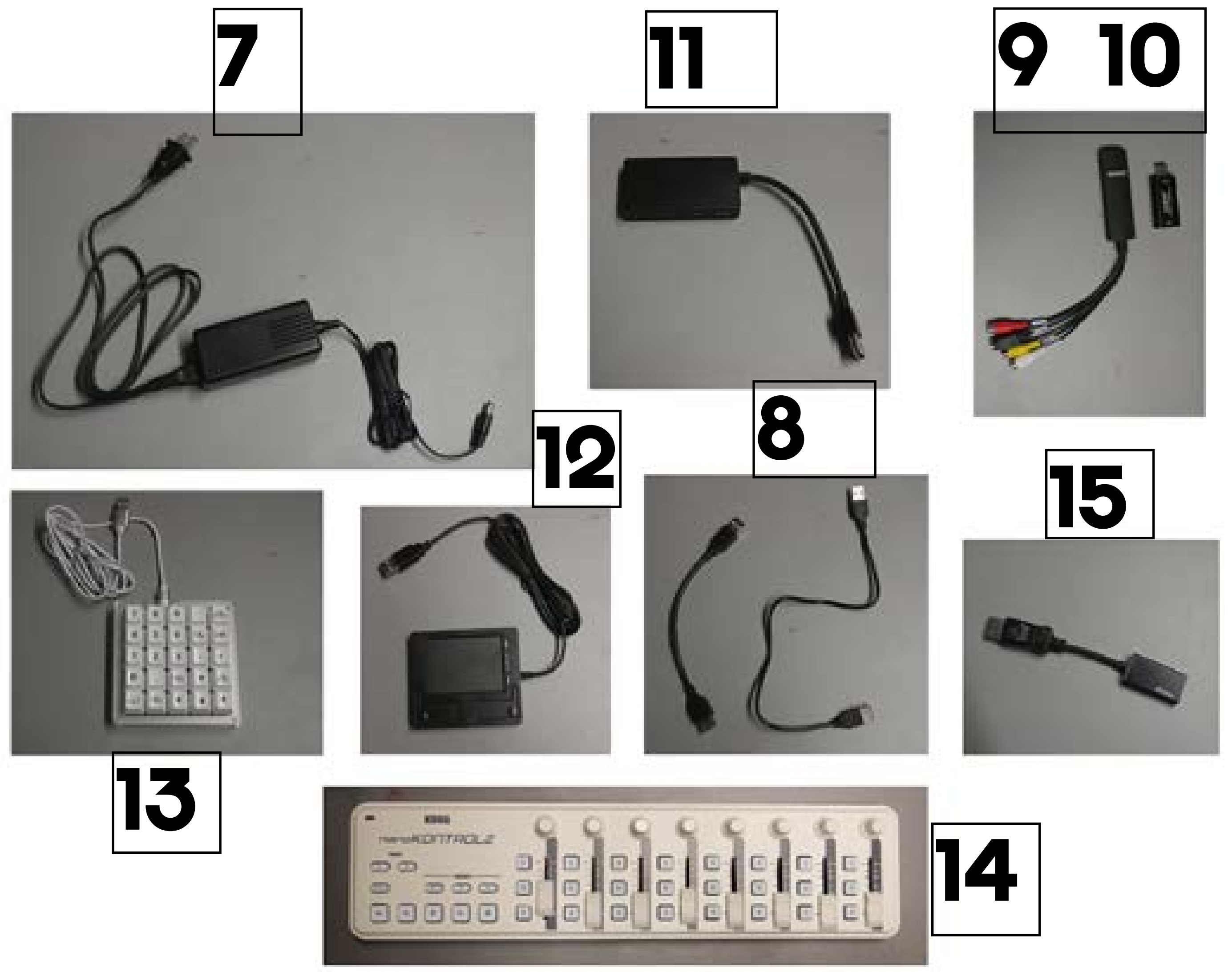
This manual can provide you with many things, but it can't replace the experience of actually working together with a large group of like minded folks. GW is a deep and somewhat fathomless system and it seems overwhelmingly likely that the community of users is going to learn exponentially more things about how they like to use this instrument than I ever will. Abstaining from participating in the community will only be a handicap to your GW experience. Plus if you email me directly for any help with GW, 99.99 percent of the time I will respond with "Please re-read the manual or "Ask around in the forums first," so probably just a good habit to get in no matter what.

2 I don't recommend doing that with any manual whatsoever, unless you are the kind of person who read every single footnote in Infinite Jest but never finished the novel itself

Section 2: Quickstart

What Is All Of This stuff?





1. Gravity Waaaves Processing unit
This is the main brain doing all the video processing and whatnot.

2. SD card slot
This contains the firmware that GW runs on. You can update GW by removing the sd card and reflashing it with a new image. The SD card is mounted in a spring loaded slot so a gentle but firm push is how to remove the card. DO NOT try to pry the card out, it is very likely you will damage the card and/or the card reader if you do so.

3. ON/OFF and RESET switches
These do pretty much exactly what you would expect

4. PSU jack
Plug the Power Supply into here

5. Displayport and HDMI video output
GW requires 2 monitors to use. One monitor is used for the GUI and one is used for the video output. If you don't have a native displayport monitor to use for GW, you can use the enclosed active DP to HDMI convertor to use with any HDMI display.

6. USB ports
this is where you plug in most all of your peripherals

PERIPHERALS

7. Power supply

8. Usb Extension x2

9. Analog Video Adapter
Use with Usb Extension. Plug either Composite (yellow RCA) or S-Video (6 pin din) cables in for input. GW completely ignores all audio so the red and white jacks do nothing. Do not connect yellow RCA and S-Video at the same time.

10. Digital Video Adapter
Use with usb extensions. you can plug hdmi digital video cables in here.

11. Usb Hub
Plug into GW Usb Port.

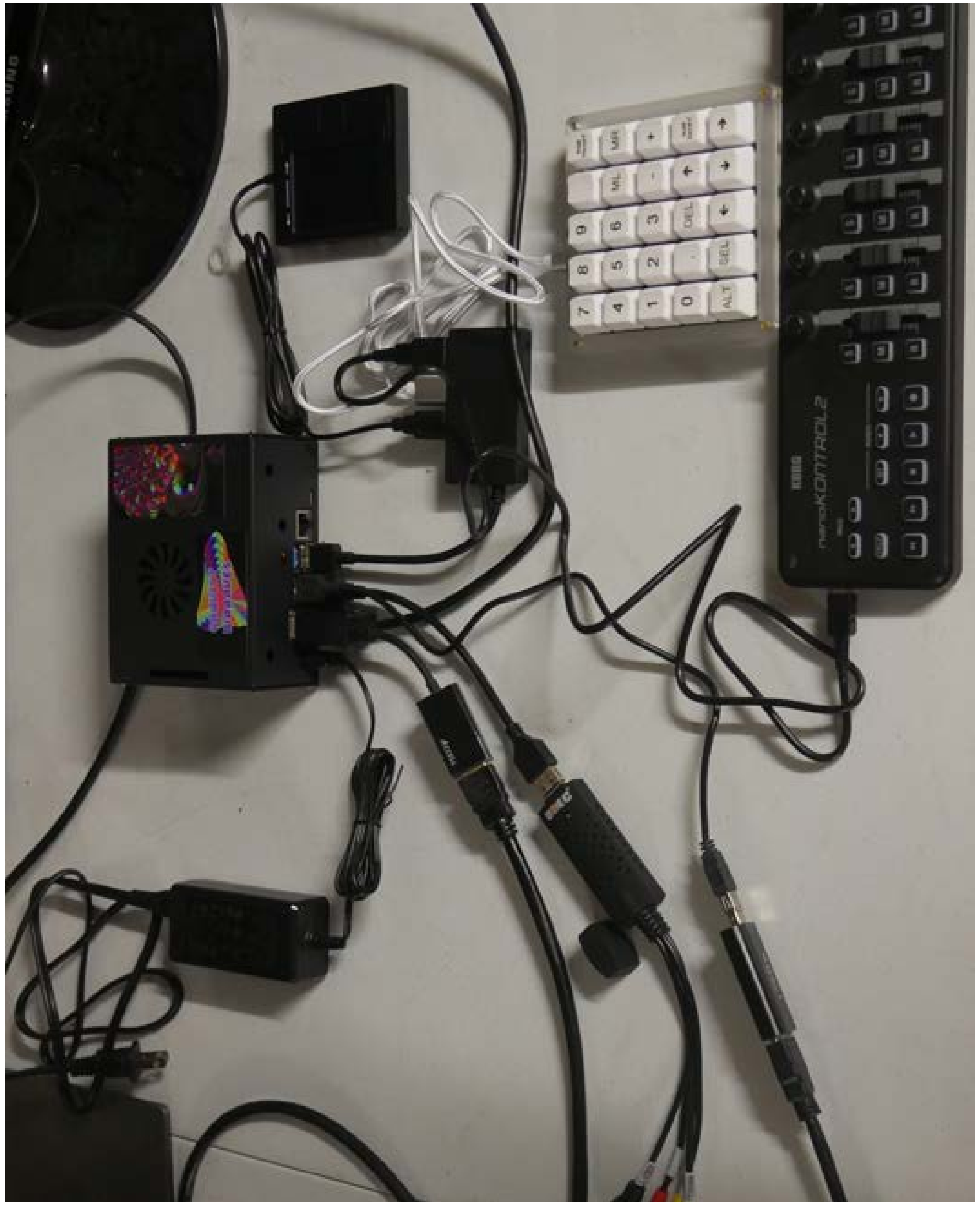
12. Trackpad
Plug into Usb Hub.

13. Keypad
Plug into Usb Hub.

14. Midi Controller
Plug into Usb Hub.

15. DP to HDMI convertor
Use with Displayport if needed.

How Do I Plug All This Stuff In?



How to Power On and Get Started

1. Plug everything in except for the video adapters. It is good practice to use the displayport for your GUI monitor.

2. Press the ON button. You will see various boot screens, text, and the GW logo and then a desktop with configureDisplays.sh and runGravityWaaaves.sh icons.

3. You will want to configure your displays first. using the trackpad click on configureDisplays.sh and then select execute in terminal. follow the instructions on screen to make sure the resolutions on both monitors are 1280x720, the monitors are located next to one another, and the correct monitors for GUI and video output are assigned. You will most likely have to do this once whenever you use a different monitor set up, so keep this in mind when you are setting things up for use with a projector.

4. Now plug in one of the video adapters, wait about 5 seconds and then plug the second one in. Click on 'runGravityWaaaves.sh', select 'execute in terminal' and after a couple of seconds you'll see the gui and video output show up!

How to Navigate the GUI

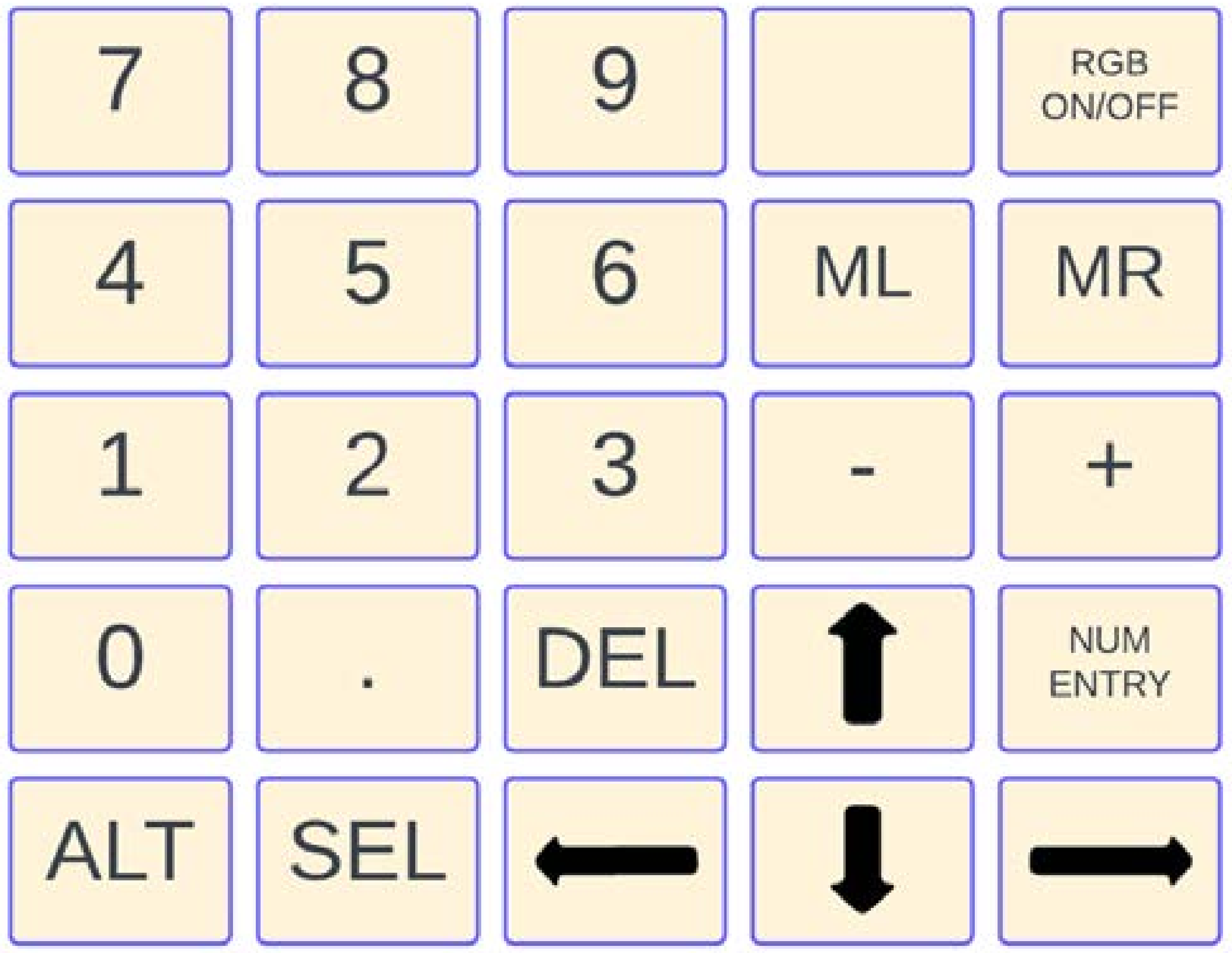
you have 3 ways to use GUI, each of which can be used in concert. You can use the trackpad on its own, the keypad, and the midi controller

Trackpad

You can use the trackpad to navigate by moving the cursor around via the paddy part and clicking via the left clicky parts. Sometimes the actual clickers on the trackpads don't seem to work. If this applies to you, you can also tap on the pad part itself instead of using the clicker, and/or use the ML/MR buttons on the keypad as backup clicking buttons.

Please note that if you accidentally move the cursor off of the gui screen and left or right click, this will disable your ability to use the keypad to navigate the gui. So try not to do that, and if you accidentally do, just remember to use the trackpad to left click anywhere back over on the gui screen to get things working properly.

When using the trackpad along with the keypad note that if you have any parameter selected with the keypad you won't be able to select, alter, or choose anything else with the trackpad until you deselect the parameter with the keypad.



Keypad

Arrow keys navigate up, down, left, and right. The *sel* (select) key opens tabs, nodes, and drop down menus, toggles checkmarks on and off, and activates sliders. When a drop down menu is activated use *up* and *down arrows* to choose between the options and press *sel* again when finished. When a slider is activated, you can use the *left and right arrow* keys to increase or decrease the parameter value and press *sel* again when finished. You'll note that with arrow keys you'll only be able to increase and decrease in amounts of .02. To dial in finer values, use *num entry* key instead of *sel* and then use the number keys *0* through *9*, the decimal point ., and *del* (delete) to manually change the value and then press *num entry* again. The *rgb on/off* key turns on and off the color thingies on the keypad itself. *ML* and *MR* are backup Mouse Left Click and Mouse Right Click for if there is issues with the trackpad. You can shut down GW from the keypad using *alt+7+8+9*.

Midi Controller

The midi controls are context sensitive for each sub menu. Whenever you enter a submenu via a node or tab you see an checkbox labled *midi/gui*. If you turn on the checkbox (via keypad or trackpad) you can then use the midi controller to control all the slider based parameters on each submenu. The way that onscreen controls map to the midi controller is: think of on screen controls as being labled 1,2,3,4 counting from left to right and wrapping around each column the same way you read a book in english. On the midi controller the numbers move in the same way, starting from the left hand.

Midi controls Latch so that whenever you switch menus back and forth using the midi controller, the controls don't just jump to whatever current positions the knobs and sliders are on the physical midi controller.

A Brief Intro to the GUI

The GW GUI is quite vast, so here's a couple of notes on structure and organization. Each BLOCK, 1 through 3 are given their own Tab. As each **BLOCK** works as it's own little sub mixer within GW, you will usually want to make sure you are viewing the same **BLOCK** that you are working in. There is a drop down in the upper left hand corner where you can select **BLOCK_1, 2, 3**, or *draw all* **BLOCKS** for video outputs.

Within each **BLOCK** Tab you will see another series of SubTabs. The organization of the SubTabs from left to right reflects the default layering of video signal flows within the BLOCK. For example, in **BLOCK_1** you will see from left to right, ***ch1***, ***ch2, fb1***. This means that by default, only ***ch1*** will be visible, and that mixing and keying will all start from ***ch1*** and then add bits and peices of ***ch2*** and ***fb1*** into **BLOCK_1** output based on what you do in ***ch2 mix and key*** and ***fb1 mix and key***.

Some SubTabs will also have another set of Nodes within them. You can check the Glossary for a full system map of Tabs, SubTabs, and Nodes. Theres no way around it, GW has a rather large number of parameters. In the interests of not creating any potential fathmoless Marianas Trench opportunities for menu divers the vertical heiarchies only get 3 deep. Unfortunately, this means that horizontal heiarchies can sometimes get pretty wide. Most of the time you won't need to access every single submenu.

The Main Menu will refer to the default GUI you end up on. The Main Menu contains the Global Controls and of the BLOCK tabs

BLOCK tab refers to any of **BLOCK_1**, **BLOCK_2** and **BLOCK_3.**

SubTab: any tab within a **BLOCK** tab. SubTabs can contain Nodes or Controls.

Node: any of the vertically arranged subMenus.

Controls will mean a set of parameters, checkboxes, and dropdowns.

SubMenu will refer to a set of Controls located within a SubTab or Node. All Nodes will be SubMenus but not all SubTabs will be SubMenus.

Examples: the SubTab ***fb1 Lfo*** contains a list of Nodes, each of which contains a set of Controls. The SubTab ***ch1 Adjust*** contains Controls.

What Video Inputs Are Supported?

Analog video:

NTSC or PAL video signals sent through the yellow rca cables or s-video. You will probably not have great luck sending analog video in directly from circuit bent/glitch video devices. It is also not impossible that you can damage GW with circuit bent/glitch video signals. Best practice is to send the circuit bent/glitched video signal through a Time Base Corrector (TBC) before going into GW.

Digital video:

Most any RGB/YCbCr signal up to 4k (3840×2160 at 30Hz). 8/10/12bit color. Unlike analog composite video, there is a very wide range of possible different video digital video signals that can be sent over HDMI, and there always exists the outside chance that some piece of video gear can have an HDMI video output that is not supported by GW.

Common Video Input Issues

Analog Video

Black and White Video, Strange Rainbow Banding, No Video At All

Either you aren't sending in an NTSC or PAL signal, or the NTSC/PAL signal you are sending is underpowered, out of range, or otherwise not a strong enough signal to work. Most often this will happen with a VCR. Best solution is to use a VCR that has a TCB built in or to use an external TBC. Other issues can occur with poorly built video encoders that don't work up to video standard specs.

Lines and/or Noise at the Top and Bottom of the Video

This happens most often with signals from VCRs, but you can also notice some strange business at the top and bottom borders of any analog video signal. This is because most every monitor that displays analog video signals uses Overscan to automatically crop the boundaries of any video signal. The only solution is to use the input adjust controls inside of GW to crop these parts of the video.

Horizontal/Vertical sync

Distortion in the horizontal/vertical directions is usually a sign that you are using glitch/circuit bent video inputs, a VCR playing a degrading VHS tape, or have an issue with the RCA cable. Occasionally it is a sign that there is something going extremely wrong with whatever video device you are sending video out from. There is nothing within GW that can fix issues with Horizontal/Vertical sync in a signal, your best bet is to use a TBC.

Digital Video

Inconsistent Signals from Laptop/Desktop Computers

Taking video out from laptops or desktop computers into GW can be inconsistent. The best way to prevent this is to figure out how to force your laptop/desktop to send out a fixed signal, like 1280x720p at 60hz. Please consult documentation for your operating system on the best way to achieve this.

Analog and Digital Video

Video Freezes

This usually happens when the Usb connectors of the Video Adapter become dislodged. You can hot swap video cables but you cannot hot swap Usb Video Adapters. Make sure that the Usb connectors all have plenty of slack and are unlikely to get pulled or detached while GW is running. You will have to either power cycle GW or exit GW and start over from the runGravity-Waaaves.sh file with the Usb cables plugged in firmly and with plenty of slack.

Long video cables can also be an issue. It is usually not recommended to run a Composite, S-Video, or HDMI cable longer than 50 feet without some kind of active amplification of the the signal. Depending on the sheilding and quality of the cable you can sometimes run into issues over 15 feet.

Quirks and Other Things to Look Out For

Not every parameter has effects between -1 and 1

Example: *posterize* and *kaleid* will only have effects between 0 and 1, and within those ranges they have stepped, not continuous values. The visual Parameter values will always be between -1 and 1 but the actual ranges scale for each different parameter. For example *x <->* is scaled to 640 pixels for live inputs so a visual value of 1 would be an x displacement of 640 pixels. You should consult the glossary for a detailed list of parameter ranges.

Not every parameter will show obvious effects immediately

This is a video mixer and not every Input or Framebuffer delay will be visible if you haven't adjusted the relevant ***mix and key*** menu. If you adjust *fb1 x <->* you won't notice any effect unless you have also adjusted ***fb1 mix and key*** to allow some amount of fb1 to be visible. If you are working within **BLOCK_2** but are only viewing the output of **BLOCK_1** then you won't see any effects of anything you adjust.

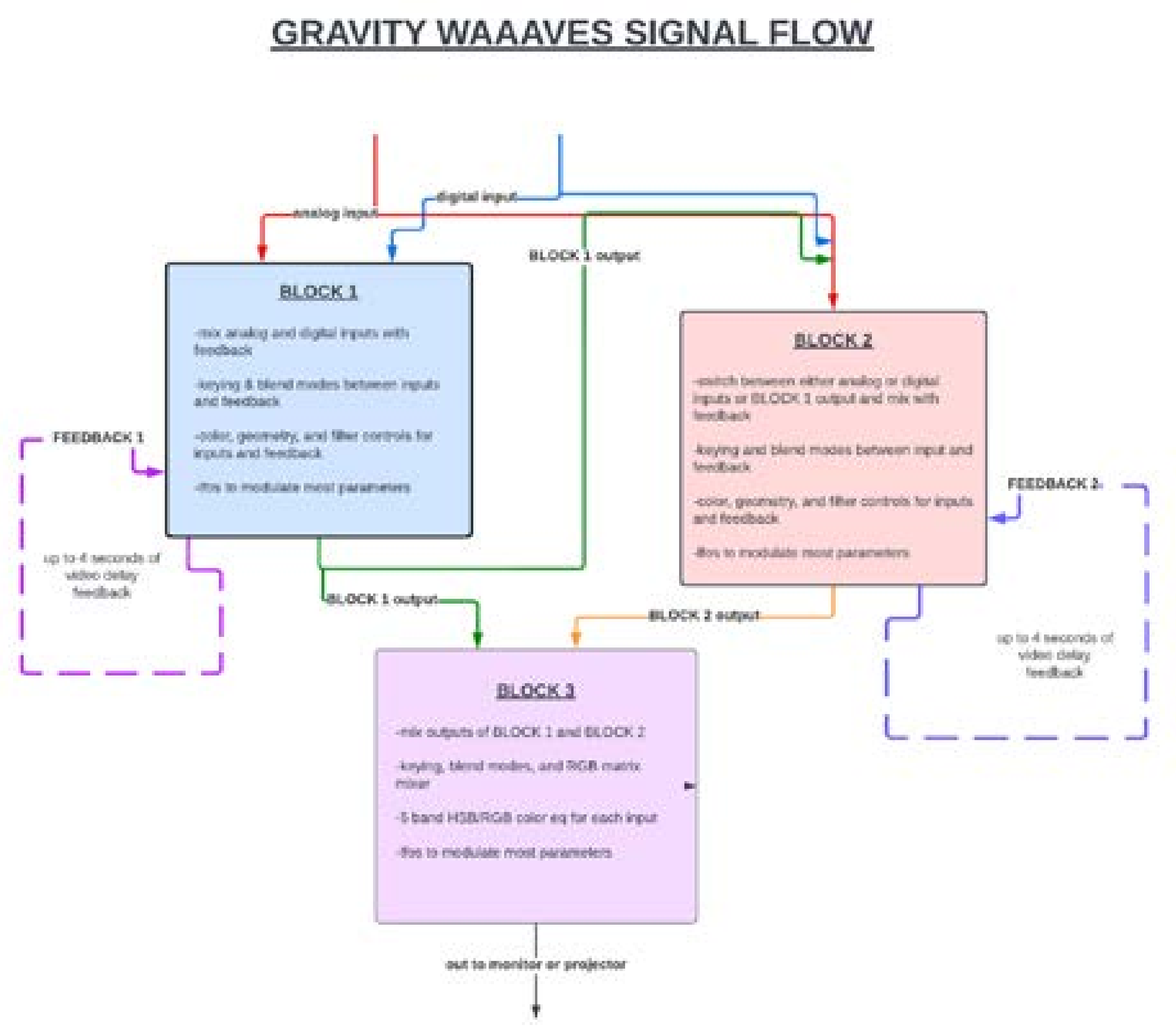
Analog and Digital Video Inputs don't always match up to the same set of input1/input2

They can switch back and forth between them between power cycling. Often times they will stay the same, but not always. Is best to not let this take you off guard. It does not matter which usb slots they are plugged into.

How to disable/enable the fan

This little unit can heat up pretty quick and the fan can be pretty loud! If you are not terribly concerned about overheating you can disable the fan by opening up GW, finding where the fan is connected to the board itself and pulling out the molex pins. Just know that you are doing this at your own risk

Signal Flow and BLOCK Structure


GRAVITY WAAAVES SIGNAL FLOW
analog input
digital input
BLOCK 1 output
BLOCK 1
-mix analog and digital inputs with feedback
-keying & blend modes between inputs and feedback
-color, geometry, and filter controls for inputs and feedback
-lfos to modulate most parameters
BLOCK 2
-switch between either analog or digital inputs or BLOCK 1 output and mix with feedback
-keying and blend modes between input and feedback
-color, geometry, and filter controls for inputs and feedback
-lfos to modulate most parameters
FEEDBACK 1
up to 4 seconds of video delay feedback
FEEDBACK 2
up to 4 seconds of video delay feedback
BLOCK 1 output
BLOCK 2 output
BLOCK 3
-mix outputs of BLOCK 1 and BLOCK 2
-keying, blend modes, and RGB matrix mixer
-5 band HSB/RGB color eq for each input
-lfos to modulate most parameters
out to monitor or projector

Each BLOCK has it's own unique set of video mixing and video processing thingies

BLOCK_1
Process and mix 2 live inputs with 1 Video Delay Line.
BLOCK_2
Process and mix either 1 live input or the output of **BLOCK_1** with 1 Video Delay Line.
BLOCK_3
Process and mix the outputs of **BLOCK_1** and **BLOCK_2** together.

By default, each of the two video inputs (analog and digital) are assigned to **BLOCK_1** and **BLOCK_2**, so if you head into **BLOCK_3** directly after booting and adjust any mixing values you will be mixing between each of the live inputs.

Framebuffers, Video Delay Lines, and Feedback

A Video Delay Line works very similar to an Audio Delay Line. A running buffer of the past 4 seconds of Video Frames is being constantly stored and updated with every new frame. The main difference between Audio and Video Delays is that Audio Delays are typically used primarily for generating effects that only use a limited amount of feedback to prevent harsh and unpleasant sounds, whereas Video Delays are best used with generous amounts of feedback in order to get the most dynamic and aesthetic results.

A Feedback (instead of Feedforward) Video Delay Line is one in which the output of a BLOCK is used as an input to the delay line instead of the input of the BLOCK. Video Feedback is a powerful and dynamic form of video synthesis that allows for a wide range of organic textures and patterns, especially when paired up with live video inputs and keyers.

Dual Video Delay Lines means that there are two Video Delay Lines that are each stored independently. Fb1 is the video delay line that is fed from the output of **BLOCK_1**. Fb2 is the video delay line that is fed from the output of **BLOCK_2**. **BLOCK_3** is processed seperately from any Video Delay lines and can only be used to post process video feedback generated within **BLOCK_1** and **BLOCK_2**. You'll notice that there are many processing features in **BLOCK_3** that don't appear in **BLOCK_1** or **BLOCK_2** and vice versa. This is because there are a lot of video processing techniques that work extremely well within a feedback loop but don't really do anything outside of one, and many techniques that work wonderfully outside of a feedback loop but would result in mostly garbage within one.

Section 3: Walkthroughs

This section is where you will actually learn how to use GW via a series of excercises that will give you an opportunity to craft various 'patches' that will give you experiential insight on how to use the various Video Delay Lines, processing sub menus, and the overall BLOCK structure all together.

These walkthroughs are meant to be worked through in order. Each one builds upon the knowledge and experience you have gained by working through the previous walkthrough. As a result these walkthroughs will start out pretty chatty in the Beginners section and become somewhat more terse as we make our way to Advanced. This is because you should be experienced enough to need less hand holding at every step by the time you've gotten that far.

These walkthroughs will not explore every single permutation of any imaginable GW patch. I could easily spend the entire remainder of my life writing this section of the manual and still not exhaust all possibilities. Instead these walkthroughs should provide you with all the knowledge you need to then explore further on your own without too much in the way of false starts or dead ends to slow you down.

A secondary purpose of these walkthroughs is to show you a general workflows for exploring different aspects of not just GW, but any kind of complex video (or even audio!) synthesis system. When there aren't too many controls or context sensitive situations it can be easy to just try cranking every parameter up to max and min and then get a good idea of whats going on fairly quickly. When working with very large systems like GW tho, its best to start with more of a subtle divide and conquor method. Isolating small parts of each BLOCK, tweaking things, and taking notes is the way to go.

Finally, working with video feedback can be fairly unintuitive for folks who are more used to the kinds of linear systems in most other video synthesis systems. Video Feedback is non-linear. In a linear system, if we know what happens when parameter A is set to 1 and parameter B is set to .5 each on their own, we will know pretty much exactly what happens when we do them both at the same time. In a non-linear system, we do not have that luxury.

How Do We Label Things

BLOCKS are very important and are always capitalized. For the sake of legibility and concision, BLOCK_1 and BLOCK_2 are often abbreviated B_1 and B_2 when referred to in Controls.

Tabs, eg **BLOCK_1, BLOCK_2**

Controls and SubTabs, eg ***B_1 color eq, finalMixAndKey***

Parameters, Checkboxes, and Dropdowns eg *B2 blue, mix type, HSB/RGB*

To notate locations of Controls when necessary **BLOCK_3->*B_1 parameters*** (change this name)->***B_1 color eq*** means ***B_1 color eq*** is a Node located under the SubTab ***B_1 parameters*** under the **BLOCK_3** Tab

BEGINNER

PART 1: BLOCK_3

The main purpose of BLOCK_3 is color Eq and mixing. This block is the most like a normal video mixer.

Controls covered: *matrix mixer, final mix and key, B_1/2 color eq,*

Matrix Mixer

Start out with two live video inputs. Ignore any aspect ratio stuffs to start with, we are mainly concerned with color eq and mixing. Its best to start with at least one input that has a wide range of brightness and at least one input with a wide range of colors. Everything will be in the **BLOCK_3** tab.

Open ***matrix mixer*** and lets map

- *B_2 green* into *B_1 red* (top row middle column)
- *B_2 blue* into *B_1 green* (middle row right column)
- *B_2 red* into *B_1 blue* (bottom row left column).

Keep values in between -.5 and .5 for now and experiment with different combinations of each.

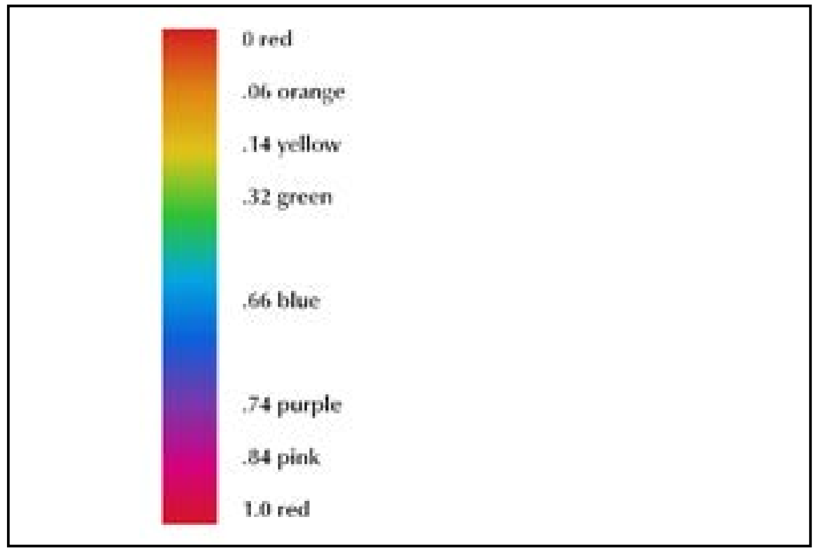
Once you've played around with that for a bit, go up to the drop down in the left hand-corner labeled *matrix mix type* and select *additive*. go back down and try some different settings. Repeat with each of the different matrix mix types.

Next go to *overflow* and select *wrap*. Try out some matrix mix values out that go all the way to -1 and 1. Previously mix values would start to clamp out at +-.5, now they wrap around from zero creating an interesting topographical style distortion. Now try all the different mix modes once again. Once finished, select *foldover* and run through all the mix modes going from -1 to 1 again. Notice that at at values of +-.5 for most mix types they fold back over in a negative direction.

Color Eq

Reset the matrix mixer via the *matrix mixer reset* checkbox. Open up ***B1 color eq*** and turn it on via the *on/off* checkbox. You should see all the video signal lose all saturation as the default mode for the colorizer is HSB mode. Lets try solarizing the video first. (link to solarize description). In the brightness (*bri)* column, take *band 1* up to 1 and *band 2* up to .5. You should see what appears to be an inverted brightness on the darkest half of the video output.

Next lets explore *hue*. We will want to crank up the saturation (*sat*) first, so turn each bands saturation up to 1. You should see, one by one, each band turning a fully saturated red, but with the darker bands (2, 3 and 4) looking toned or shaded. To see what all bands look like without shading or toning, try all bands at *bri* and *sat* set to 1. Explore adjusting the *hue* of each band one by one. Notice that on full saturation and brightness the rainbow matches up to parameter values like so:



When adjusting *hue* on adjacent bands, notice how gradients form between each band.

(we should have a little graphic thing that cuts up sections here. just a slice from one of the images we've got so far should work)

Next lets explore *sat* by decreasing the saturation by drastic amounts on each band. For example set *sat* for each band like so

- *band 1* to 1
- *band 2* to .2
- *band 3* to .8
- *band 4* to .4
- *band 5* to 0.

You should notice that some bands now have a pastel look to them. Explore tweaking *sat* values for each band. It will be more drastic and noticable to start by alternating large and small values for each successive band but you should also try things like setting up gradients.

(graphic breaking thing)

Lets switch our view to B2 input and try out RGB colorizing on this channel. Within the global BLOCK_3 tab, change *final mix order* dropdown to *BLOCK_2->BLOCK_1* and the final video output will switch to your secondary input.

Open ***B_2 color eq***, check *hsb*/*rgb* and *on*/*off*, and you will notice that unlike hsb color eq, rgb color eq does not show any immediate effects on the output video. Lets try adjusting only the even numbered bands to get a feel for how rgb color eq works. Unlike hsb color eq where we strip all of the saturation and hue information from the video signal and then reassign from scratch, rgb color eq keeps all of the color information as is and we use the sliders to offset the rgb values on a band by band basis.

Go to *band 2* and crank *red*, *green*, and *blue* all the way up to 1. You should see that band 2 is now almost entirely white, but with some very odd gradients merging into bands 1 and 3. Next try heading up to *band 4* and crank *red*, *green*, and *blue* down to -1 and you should see that band 4 is now almost entirely black, with the same odd gradients merging into adjacent bands. If band 4 is a bit too slim for this effect to be clear, try band 5 instead.

Next explore tweaking different values for ***red***, ***green***, and ***blue*** on each of band 2 and 4, starting with small changes on each and then more drastic. The thing to keep in mind here is that we are still assigning bands via calculated brightness, but on a per pixel basis per band our controls will appear to be much less uniform than in the hsb color eq mode. In general, rgb color eq is much less intuitive than hsb color eq. Offsetting values instead of starting from scratch can make it fairly difficult to intentionally affect the video unless you are well experienced in navigating in rgb space. The plus side to this apparant lack of intentional control is that there are effects we can get from this style of color eq that would be impossible in rgb mode. You shouldn't think of these modes as just two different ways to get the same results, but instead as wildly different tools.

(visual break)

Final Mix and Key

Make sure that in ***B2 color eq*** we have at least one band totally set to black (all values at -1) and in B1 color eq we have a similar setting (brightness value at -1). Reset ***final mix order*** dropdown to ***BLOCK_1->BLOCK_2*** and open ***final mix and key***. During this entire section, it might be very handy to occasionally switch output mode to ***draw all BLOCKS***; this can be selected in a dropdown in the upper left hand side of the global menu.

First we will explore lumakeying. Whenever ***key mode*** dropdown is set to ***lumakey***, changing any one of ***key red, key green,*** or ***key blue*** will change all of them. This is a secret peek as to how the luma/chroma keyer in GW works under the hood, as always you can check the glossary for more info. You'll notice that adjusting these key colors won't actually change anything other than the ***key value color*** swatch. The reason is that we use these sliders to select a value to key around, but we also have to select a threshold that controls how much luma (or chroma) near the key value color will get replaced with the secondary video signal.

Reset any of ***key red/green/blue*** back to 0 so that the ***key value color*** swatch is back to black. Adjust ***key threshold*** slowly up from 0 until you see about half of B_1 is keyed out. Notice that any alterations you made to in ***B1 color eq*** are passed into the keyer. This can help you out a lot in the future, either if you just need to tweak some color values a little bit to get a cleaner key, or if you want to do something crazy like have bands 1, 3, and 5 all set to the same brightness and key them all out.

Next, leaving ***key threshold*** at the same value, change the key value via any of ***key red/ green/blue***. Notice that as the ***key value color*** swatch gets brighter, different parts of the video get keyed out. Bring any of ***key red, key green,*** or ***key blue*** up to 1.0 so that the ***key value color*** swatch is now white and experiment with changing ***key threshold***. You'll notice that now the keying starts at the brightest part of the video signal and threshold adjusts the lower bounds of the keyer.

Lets play a little bit with ***key soft***. Increase the value and take note of what happens in the keyed out portion of output. You should see a linear fade between the keyed out part of B1 and B2, weighted by the brightness value of B1.

Next we will experiment with chroma keying. Select ***reset*** in ***final mix and key***. Lets first make sure we have a decent amount of green in B1. Enter ***b1_color eq*** and make sure at least 1 band is fully saturated fully brightnessified green (***hue*** at .32, ***sat*** at 1, ***bri*** at 1). Head back over to ***final mix and key*** and change ***key mode*** to ***chromakey***. Now you'll be able to adjust each of the

key red, key green, or ***key blue*** values individually. Leave ***key red*** and ***key blue*** at 0, and turn ***key green*** up to 1, then start adjusting ***key threshold*** higher until you've chromakeyed out all of the green from B1. Repeat this process with adjusting values in ***b1_color eq*** and then ***final mix and key*** to key out red, blue, and then any other color you like!

Finally, lets explore combining mix and different mix modes with the keyer. When the keyer is active, adjusting mix modes affects only the unkeyed portion of B1. To illustrate this very blatantly, switch ***mix type*** to ***difference***, ***overflow*** to ***wrap***, and slowly adjust ***mix*** up to 1.0 and then down to -1.0. Try a bunch of different combinations of ***mix type*** and ***overflow*** and slowly scrolling ***mix*** up and down from 1.0 to -1.0 to get a good feel for the different combinations of ***mix modes*** and ***overflow***.
(break)

What did you learn?
-how does the matrix mixer work
-how does color eq work in hsb and rgb modes
-how does the luma and chroma key work
-how to use color eq to optimize luma and chroma keying

Further experiments with this patch
-Try using the ***color eq*** in ***hsb*** mode in concert with Lumakeying to explore keying into more subtle grades between just black and white
-Try using the ***color eq*** in hsb mode in concert with Chromakeying to explore keying first into red and green, and then into more subtle colors outside of the primary spectrum.
-Try doing the same steps as above but with ***color eq*** in ***rgb*** mode.

PART 2: BLOCK_2

The main purpose of **BLOCK_2** is mixing one input with a feedback buffer. **BLOCK_2** is very similar to how Waaave_Pool works.

Controls covered: ***BLOCK_2 input adjust, fb2 mix and key, fb2 geo***

For this lesson, it will be best to have an input that is fairly dynamic and controllable both in brightness and in motion. Either a live camera that is pointed at you or a high contrast video oscillator based input would work great.

Open up **BLOCK_2** and go to ***BLOCK_2 input adjust*** and select your preferred input from the drop down menu. The default aspect ratio is sd (4:3) so if you've got an hd aspect ratio, check the ***hd aspect ratio box***. Lets try tweaking a bit of the input parameters. Adjusting *x* <->, y <->, z <->, and *rotate* <-> should all be fairly intutive, but do a little experimenting with each to get a feel for the ranges involved.

Once you feel comfortable with these parameters, head over to ***fb2 parameters*** and open up ***fb2 mix and key***. Adjust ***key threshold*** to .5 and notice that the darkest half of your video input should be keyed out and replaced with feedback painting. Head up to ***fb2 delay time*** and start adjusting upwards. You'll notice pretty drastic changes in the feedback between 1 and 10. Experiment a bit with different kinds of motion in your input video (waving hands or osc rate) and see how these changes flow through the feedback.

Next lets move to ***fb2 geo*** and play around with geometry a bit. Adjust ***fb2 delay time*** back to 1 for now and then start moving *x* <->, first negative, then positive. You should see some apparant motion in the feedback which gets faster and more distinct the farther *x* <-> is away from 0. Go back up and turn ***fb2 delay time*** up to 8 and try tweaking *x* <-> once again.Now the feedback motion should look more scattershot, with a little bit of jerky motion trapped in the buffer from when you swept the delay time up. Hit ***fb2 framebuffer clear*** button and with the feedback reset you should see the transient jerkiness dissapear. Play around a bit more with *x* <-> and now *y* <-> with ***fb2 delay time*** kept at 8 until you feel like you have a pretty good handle on how x and y displacement work with this length of delay.
B R E A K

Next lets try out some longer delays. GW processes video at 30fps, so if we set any delay times to 30, we are setting up a feedback loop of 1 second of video. Controls start to get a little confusing here, mainly because it takes a bit of time for effects to ripple through the entire 1 second loop. When feedback is in a 1 frame loop. any alterations in the buffer geometry move at a frantic pace at rates that register as smooth movement to human perception. When feedback is in loops at 1 second or greater, its a bit more trickier to maintain an attention span to directly perceive the effect of each loop.

Lets experiment with this a bit. Adjust ***fb2 delay time*** to 30 and increase *z* <-> to .2. Over the course of about 30 seconds you should see the feedback dwindle into a vanishing point. The exact placement of the vanishing point is wherever *x* <-> and *y* <-> are set to. Bring ***fb2 delay time*** time down to 1 and then tweak *x* <-> and *y* <-> a bit to see how this works. Find a nice vanishing point that you like and then start tweaking *z* <-> in a negative direction, up to -.100. You should see the feedback 'zoom in' and fill the screen. If you don't, try centering *x* <-> and *y* <->. Next adjust ***fb2 delay time*** back to 30, and hit ***fb2 framebuffer clear***. Watch how you can see a degradation in each iteration of the feedback as it expands to fill the screen.
B R E A K

Now lets play with ***rotate*** <->. Set ***rotate*** <-> to .180. Depending on how fast or slow your input is moving, it can be a bit tricky to see what is going on at first, but try waiting about 30 seconds of observation first before adjusting ***fb2 delay time*** back to 1. You should see feedback spirals zooming in. Change *z* <-> to .18 and youll see the spiralling going back into a vanishing point. Tweak *x* <-> and *y* <-> a bit more to see how rotations, z, x, and y all interact with one another. Go back and forth between 1 and 30 delay times, clearing the buffer each time to get a feel for how longer delay times contrast with short ones.

What did you learn?
-how does adjusting geometry work with live inputs
-how does adjusting geometry work with feedback
-how does mixing feedback with live inputs work
-how do different delay times affect feedback patterns and behavior

Further experiments with this patch
-try different mixing modes on ***fb2 mix and key*** like linear fade, additive, difference, mult,

and dodge.

-try going way more extreme with values of *x, y, z*, and *rotate* <->. Experiment with differ ent options for *fb2 geo overflow* as well

-try longer delay times, going all the way up to 240 (120??)

PART 3: BLOCK_1

The main purpose of BLOCK_1 is splitting the difference between BLOCK_2 and BLOCK_3. BLOCK_1 allows you to mix 2 live inputs with 1 video delay line. BLOCK_1 is very similar to how VIDEO_WAAAVES works.

Controls used: *ch1 adjust, ch2 mix and key, ch2 adjust, fb1 geo, fb1 color*

We want to have two live inputs, each with a nice full range of brightness to work with. Lets start with exploring multiplicative mixing. Multiplicative mixing is handy for creating masks with plenty of negative space to work with as any black pixels on either input will map to black pixels on the mixed output.

Head over to ***ch2 mix and key***, set *mix type* to *multiplicative*, and *mix* to .5. You should see that the blackest parts of both inputs are pretty close to black now and that the rest of the colors are a bit chaotic. Lets try out a variety of inverts on each channel and see how it affects the mixed output. Near the bottom of ch2 adjust you'll find a row of checkboxes starting with *hue invert* and ending with *solarize*. One by one, try checking, and unchecking each one of the invert buttons to see how it interacts with multiplicative mixing. When you get to *solarize*, leave it on. Head over to *ch1 adjust* and run through the same steps, also leaving *solarize* on when finished. You should see a fair amount of negative space in your video output now.

B R E A K

Now lets start getting feedback involved. Go to ***fb1 parameters -> fb1 geo*** and set *x* <-> to .02 and *z* <-> to -.02. You shouldn't see anything changed in the video output yet because we haven't mixed fb1 into the output yet! Head back up to *fb1 mix and key* and set *key threshold* to .1. You should see a good chunk of the output video keyed out and replaced with feedback. If you arent satisified with this amount try setting *key threshold* higher.

Now that we've got some feedback keyed in with a bit of motion, lets explore color. Open up ***fb1 color*** and you will see a bunch of controls here for hue (*hue*), saturation (*sat*), and brightness (*bri*), but with different glyphs next to each set of three. *Hue/sat/bri* ++ is pedastal or offset which means adding or subtracting the parameter value as constant number to the color value. *Hue/sat/bri* ** is attenuversion which means multiplying the parameter value to the color value. *Hue/sat/bri* ^^ is somewhat similar to gamma and involves taking the color value to the exponetial power of the parameter value.

Lets start with ++ to see how offsets work with feedback. Tweak *bri* ++ up and down, small adjustments at first and then larger ones. For positive values you should see the feedback trails increase in brightness, and negative values should cause the trails to fade away. Leave *bri* ++ set at .02. Try the same process with *sat ++,* observing how the different numerical values affect the saturation of the feedback, and leave it at .02 when satisfied. Next start to bring *hue* ++ up. For small values you should see some slow hue cycling in the feedback, larger values should strobe and create reaction diffusion patterns in the hue. Exact values numerical values for 'smaller' and 'larger' will depend quite a bit on what kind of video inputs you are working with.

Try setting *hue* ++ to some negative values now. You should see some chaotic results, including loss of brightness and saturation in the feedback. Set *hue* ++ to -.320 and start tweaking *sat* ++. Increase *sat* ++ to 1.0 and you should now see a loss of brightness. Head over to *bri* ++ and increase to 1, and you should see the feedback fill the screen, but with a lot of desaturated white. The point here is to show that when working with feedback (in GW and also in general), altering HSB values individually can result in nonlinear and unintuitive results. That is, changing the values of *bri* ++ and *sat* ++ with *hue* ++ at .320 vs *hue* ++ at -.320 will have radically different effects.

B R E A K

Hit *reset* and lets get started exploring *hue/sat/bri* **. Follow the same pattern, first adjusting *bri* **, then *sat* **, keeping both *bri* ** and *sat* ** and values at least slightly above 0, and then adjusting *hue* **. We want to keep following this order of adjusting hsb values because we need some brightness adjustment in order to see whats going on with saturation, and we need some saturation established to get good visual feedback on hue! Once you feel like you have a good handle on attenuversion, hit *reset* again and head down to to *hue/sat/bri* ^^ and follow the same pattern to try and get a decent appreciation for the different effects that exponential HSB mapping offers.

Once you feel comfortable with each **, ++, and ^^ operations try exploring relationships between each set of operators. For example, try setting all ** to positive values, and all ++ to negative values, then tweak the ^^ values up and down. There is not any kind of clear roadmap for exploring all possible reaction diffusion patterns in the HSB feedback world, the important thing to take away from this part is that this is pretty much an inexhaustable source of new potential feedback patterns.

What did you learn?

-How to mix inputs and feedback in BLOCK_1

-How does solarizing, channel inverts, and rgb invert work for live inputs

-How to work with color in feedback.

Further experiments with this patch

-experiment with longer values for *fb1 delay time*.

-experiment with larger values of *x* <-> and *y* <-> in addition to bringing in *z* <-> and *rotate* <-> while playing with color.

-try different settings in ***fb1 mix and key*** like *multiplicative* or *difference* mix types and different values for *mix* while messing around with different feedback color settings. Try out switching the value of key order every so often and see what happens.

<u>INTERMEDIATE</u>

Here we are going to take a look at chaining two blocks together and using the lfos to automate modulations

PART 1: Keying BLOCK_1 into BLOCK_2

In this section we cover using built in geometrical animations as seeds for feedback, automating parameters using lfos, and explore running feedback from **BLOCK_1** into **BLOCK_2.**

Controls used: ***fb1 geometrical animations**, **fb1 mix and key, fb1 geo lfo 1, fb1 color lfo, BLOCK_2 input adjust, fb2 mix and key, fb2 geo lfo 1, fb2 geo***

Lets start in **BLOCK_1**. If we want to test out feedback stuffs we have the option of bypassing any live inputs and just using some built in geometric animations as a seed. Head over to ***fb1 mix and key*** and set *mix* to .5. You should see your live input appear to freeze up as it gets bypassed and the last frame locked into the framebuffer. Head down to ***fb1 geometric animations*** and turn on *septagram*. Go to ***fb1 color*** and set *bri* ++ to -.02 and you should see the framebuffer feedback fade away very slowly, leaving only trails from the septagram animation. If you bring *bri* ++ back up to 0 and wait about 15 seconds you'll see why this animation is called 'septagram.' Set *bri* ++ back to -.02.

Head over to ***fb1 lfo*** and open ***fb1 geo lfo 1***. Set *x* <-> *a* to .4 and *x* <-> *r* to .02 and you should see the feedback slowly displace in the x direction first to the right and then to the left. Experiment with larger values for rate to see how modulation scales up and down.

Open ***fb1 color lfo*** and set ***bri*** ** *a* to .06, *bri* ** *r* to .04, ***hue*** ** *a* to .02, and ***hue*** ** *r* to -.02. You should see the feedback slowly fading in and in, with the occasional hue cycling speeding up and slowing down. Try some higher values for rate (*r*) for both ***hue*** and ***bri*** to see how lfos interact with the color space in feedback.

B R E A K

Next lets experiment with mirrors and flips. Head back to ***fb1 parameters - > fb1 geo*** and, one by one, try turning on and off *h mirror* and *v mirror*. You should notice that feedback will dissappear about half of the time whenever it is displaced too far into the half of the screen that is being reflected into. Switch both mirrors off and now try turning on *h flip*. You'll see some weird strobing for a second until it stabilizes. You should that the x displacement lfo seems to have been bypassed. If you look closely though, you will see that the lfo is still displacing the buffer, but since the entire buffer is being flipped around wherever x displacement is set to, the movement doesn't get fed back and result in such large apparent motion. Do a quick run through of adjusting the rest of the parameters on this page to see how each one, from *x* <->, down to *x shear* interact with the flip. Turn the *h flip* on and off every so often to see the difference between flipped modes and regular modes with all of the different goemetry settings.

Try the same process, but with the *v flip* enabled to get a feel for how geometry works with vertical flips. You'll notice a similar effect where the y displacement doesn't seem to work either, unless you really squint and concentrate.

B R E A K

Now lets play around with bringing the output of **BLOCK_1** into **BLOCK_2** to process it. Leave the *v flip* switched on in ***fb1 geo*** and head up to the draw menu at the top of the screen and

select *draw BLOCK2*. At first we will only see the default input for **BLOCK_2**. To bring in the output of **BLOCK_1** instead open up **BLOCK_2->*BLOCK_2 input adjust*** and select *BLOCK_1* as input and you should now see the v flipped septagram with modulated x displacement feedback.

Head over to ***fb2 parameters*** and set *fb2 delay time* to 8. Open ***fb2 mix and key*** and set *key threshold* to .02. You should now see another layer of feedback kick in with some longish delays. Experiment with slightly larger values of *key threshold* to see how it looks when you key out different amounts of **BLOCK_1**'s output.

B R E A K

Lets explore modulating y displacement in fb2. Go to ***fb2 lfo-> geo lfo1*** and set *y* <-> *a* to -.8 and slowly explore bringing *y* <-> *r* up from 0 to 1. You'll notice something kinda funny a when *y* <-> *r* is at about .5. It will seem as though for low values of rate that the y displacement seems to be much larger and at higher values the total displacement seems to become very small. This is very strange, because you've only been altering rate instead of amp! The reason for that is very simple: whenever you alter a parameter in feedback, the feedback system itself has its own 'intertia' that will amplify tiny adjustments into larger ones with each iteration. Not only that, but the total length of the feedback loop will affect how the motion works as well. Set *y* <-> *r* to 1, head back to ***fb2 parameters***, and set *fb2 delay time* down to 1 and you'll see that the y displacement gets much larger! The moral of this story is: nothing is straightforward when you are working with feedback loops and intuition can really bite you in the butt sometimes.

If you are still feeling confused about this concept I would recommend trying, with just one **BLOCK** active, to set up a patch with one of the geometric animations, and experiment with all of the ***geo1*** and ***geo2*** lfos at various amps and rates, and changing up *delay time* from 1 to 4 to 8 to 16 to 32 for each setting.

Back to the current setup. Lets keep *y* <-> *r* at .8 and start to play with *z* <->. Set *z* <-> *a* to .4 and *z* <-> *r* to .6 and you'll see feedback first zoom waaay out and then blow waay up, filling the screen. This is because all of GW's lfos are bipolar sine waves, meaning that if their amplitude is .4, they will curve up to adding .4 and then curve down to subtracting .4 from the parameter value. Lets figure out how to make the z displacement only modulate in a 'zoomed out' kind of way, instead of zooming in half of the time. Head back over to ***fb2 parameters -> fb2 geo*** and set *z* <-> to .4 and you should see that this does the trick. At first the zooming was going from .4 to -.4, but when we set the *z* <-> to .4, that offset the lfo to instead go from 0.8 to 0.0. This is a handy way to set any lfo to behave in a more 'unipolar' manner.

B R E A K

Lets head back to the ***fb2 lfo -> fb2 geo lfo 1*** and start playing with rotations. Set *rotate* <-> *a* to 1 and *rotate* <-> *r* to .002. Note that you will need to use the NUM ENTRY button on the keypad in order to set rate to such a small value. You should see a very long and slow rotation in the feedback.

Now that we have some modulated zooms and rotations happening, lets explore the different geometry overflow modes. Swing back over to ***fb2 parameters ->fb2 geo*** and select *overflow mode* toroid. You should see some messy confetti looking stuff at first until things start to stabalize, then you'll see some fairly hypnotic fractal patterns start to emerge. To enhance the fractal patterns at this part, you'll want to set *z* <-> even larger. Experiment with values all the way up to 1 and each time you alter the z displacement, give it about 15-30 seconds to see how it works within the inertia of the feedback system.

Once you've had some fun with that mode, try setting geo overflow to mirror, and make sure to clear the framebuffer.

What did you learn?

- How to layer feedback from BLOCK_1 into BLOCK_2
- How to use lfos to automate geometry modulations in feedback
- How to use the built in geometrical animations as seeds for experimenting with feedback
- The difference between Flips and Mirrors in feedback.
- How does 'inertia' in video feedback interact with lfos
- How do different geometry overflow modes work

Further experiments with this patch

- Try out longer delay times for both fb1 and fb2. Make sure to clear the framebuffer each time you change delay time.
- Try out modulating *x/y shear* and *x/y squeeze* for both fb1 and fb2
- Experiment with ***fb params - > fb color ->*** *bri invert* on both fb1 and fb2 while doing further experiments with overflow modes.
- Explore modulating *x/y/z/rotate* <-> lfos in **BLOCK_2->*BLOCK_2 input adjust lfo***

PART 2: Feedback Oscillators in BLOCK_1 & Color Eq in BLOCK_3

In this section we learn how to first use color channel inversions and geometry lfos to create 'feedback oscillators' and then use the color eq in BLOCK_3 as a colorizer.

Controls used: *fb1 mix and key, fb1 geo, fb1 color, fb1 filters, fb1 geo lfo 2, B_1 color eq, B_1 color eq lfo 1, B_1 color eq lfo 3*

At this point in the walkthroughs I think that y'all should be comfortable enough with GW and the gui for us to be a bit more terse in our explanations. If you don't think you are there quite yet, please try going back through a couple of the previous walkthroughs and make sure to give yourself plenty of time working through the Further Experiments portions of each walkthrough.

Lets start out in BLOCK_1 and set up what I like to refer to as a Feedback Oscillator system, using no inputs or geometrical animations at all. Follow each of these steps in order

- ***fb1 parameters -> fb1 delay time*** set to 4
- to bypass inputs go to ***fb1 mix and key*** and set *mix* to .5
- ***fb1 geo*** -> *z* <-> to .02
- ***fb1 color ->*** *bri invert* switch on
- ***fb1 lfo -> fb1 geo lfo 2 ->*** *x stretch a* to .2, *x stretch r* to .12

Using this kind of 'no input invert mode' with a slight zooming out via *z* <-> is a pretty handy way to get a feel for how all the geometrical displacements work as it adds parallel outlines of movement that can work as vector maps of the displacement.

Note that our inverted feedback has a lot of greyish space at the vanishing point. There are at least two ways to deal with this. You can either set *z* <-> to a higher number, round about .2 should do, or go to ***fb1 color1*** and set *bri* ** to 1.0. We do want to have some grey in the mix tho, so after experimenting a bit set *z* <-> and *bri* ** back to .02 and 0 respectively.

While we have this feedback oscillator system running, lets do a bit of experimenting with different delay times. I think its always good to try basic sequences like 4, then 8, 16, 32, etc but you are welcome to try out any variation on this you like, just make sure to clear the framebuffer each time you alter the delay time to get a clearer picture of whats going on. Once again, take note of how the inertia of the geometric displacement lfos gets radically changed with larger delay times.

B R E A K

Lets explore some more of ***fb1 geo lfo 2***.

- *x shear a* to -.160, *x shear r* to -.075
- *y shear a* to .280, *y shear r* to .029

This is a good time to play around with kaleidoscope lfos (*kaleid sl*) as well but we'll need to activate it back in ***fb1 geo*** before seeing any effect from *kaleido sl* lfos. Don't forget to go check out the Glossary for definitions if you want more textual info on things like what these mysterious parameters like *kaleid sl* are actually doing!

- ***fb1 geo ->*** *kaleid* try slowly bring values up from 0 to 1, then set it to .240
- you should notice some strobing after that, try setting *z* <-> to .3 or .4 to smooth it out
- ***fb1 geo lfo 2 ->*** *kaleid sl a* to .26, *kaleid sl r* to .06

Observe how that final step adds extra motion to the pentagonal shape we have set up with *kaleid*.

B R E A K

We have enough going on in our feedback oscillator system to start playing around with color eq!

- go to **BLOCK_3 -> *B_1 parameters -> B_1 color eq***
- band 1: *hue* .74, *sat* 1, *bri* 1
- band 2: *hue* .5, *sat* 1, *bri* .5
- band 3: *hue* .32, *sat* 1, *bri* 0
- band 4: *hue* .22, *sat* 1, *bri* 0
- band 5: *hue* 0, *sat* 1, *bri* 0

You should see the white parts are now replaced with red, the black parts are replaced with purple, and some thin greenish and bluish outlines here and there. At the center where we used to have greyish strobing patterns it should now look greenish blue.

With these color eq settings in place, lets explore what happens when we go back to **BLOCK_1** and play around with filters! Using the color eq to offset brightness, saturate everything, and assign each band a radically different color will put us in a good position to see filter effects in a more obvious way.

B R E A K

Head back to **BLOCK_1 ->** ***fb1 parameters -> fb1 filters*** and set ***temp 1 amt*** to .4. you should see apparent motion slow down a bit and more of the blue and green color bands represented in larger amounts. Patterns that were formerly greyed out or strobing are now given a chance to form. While you are here, try some different settings for ***temp 1 q***. Parameter values between -.5 and .5 will be the most illluminating. Once finished, set ***temp 1 q*** back to 0.

Next lets play with the two temporal filters in succesion. Set ***temp 2 amt*** to .34. Note: running two temporal filters in succession like so is not equivilant to setting temp 1 to .4+.34=.74. Like most things in a feedback loop, they interact with one another in nonlinear ways. For this setting, we want to adjust ***temp 1 q*** back up a bit to get more definition in the patterns. Try bringing ***temp 1 q*** back up to somewhere in between .15-.35 and see how that affects the pattern formations in the **BLOCK_1**.

B R E A K

Finally lets explore adding lfos to the color eq. When modulating color eq, a little bit goes a long way. You can certainly modulate everything all the time, but it can be overwhelming when the goal is to figure out what is actually happening. With that in mind, lets start out by modulating only band 2 and band 5.

- **BLOCK_3 ->** ***B_1 lfo -> B_1 color eq lfo 1***
- ***band 2 hue a*** to -.18
- ***band 2 hue r*** to .08
- ***band 2 bri a*** to 1
- ***band 2 bri r*** to .14
- move to ***B_1 color eq lfo 3***
- ***band 5 hue a*** to 1
- ***band 5 hue r*** to .003
- ***band 5 bri a*** to .5
- ***band 5 bri r*** to .15

See if you are able to pick out the nuances of each individual bands modulation. Some clues to help you see whats going on would be that band 2 is modulating the full range of brightness, so it should go from completely dark to as bright as possible. On the other hand band 2 is only modulating about 20 percent of the total range of hue so we should only see colors ranging from about green to indigo. Band 5 is only modulating about half of the total range of brightness, so it should never go completely black, but it is modulating the full range of hue so you should see every color in the rainbow show up at some point.

What did you learn?

- Using no inputs, bri invert, and geometry modulations to create a feedback oscillator
- Working with stretch and shear lfos
- Working with kaleido in feedback.
- Using temporal filters to create space for more patterns to form in feedback
- Using color eq lfos

Further experiments with this patch

- Using this kind of 'no input invert mode' along with every parameter in the fb1 geo lfos
- color eq -> RGB mode instead
- Modulating each bands HSB/RGB components one at a time
- Experiment with blur and sharpen filters combined with temporal filters

ADVANCED

BLOCK1 internal feedback, BLOCK2 camera input feedback, BLOCK3 processing and mixing

In this section we will learn how to use all three BLOCKS together

Controls used: ***fb1 mix and key, fb1 geo, fb1 color, fb1 filters, fb1 geo lfo 1, fb1 geo lfo 2, final mix and key, B_2 color eq, BLOCK_2 input adjust,***

Since we would like to explore working with camera feedback you'll want to have a nice camera handy and either a tripod or some other way to hold it steady and pointed at your output screen. It won't particularly matter if you are using a camera with analog or digital video output here.

At the top left of the gui select *draw all BLOCKS*. This will allow us to preview what we are doing in **BLOCK_1** and **BLOCK_2** while seeing how we mix them together in **BLOCK_3**.

Next go to **BLOCK 2 ->** ***BLOCK_2 input adjust*** and change input to select live camera input if it isn't already showing up here. Select ***hd aspect ratio*** if that applies as well.

Lets go to BLOCK 1, and lets bypass any inputs and create another feedback oscillator.
fb1 mix and key, mix to .5
delay time to 12
geo-z to .1
color bri & sat invert on
bri ** to 1.0
hue ++ to .4
fb1 filters blur amt 1, rad .1,
temp 1 amt .08 q .16,
temp 2 amt .1, q .06

fb1 lfo
geo lfo 1 z a to -.08, r to .16
geo lfo2 x stretch a 1, r .1

y stetch a 1, r .04
x shear a .4, r .67
y shear a .4, r .085

BLOCK 2
Select draw BLOCK_2 for the moment and play with filters, geometry, and overflows

draw all blocks for mixing
BLOCK3
final mix order block-2->block1
set up your camera on a tripod or something stable to point pretty directly at the output screen
final mix and key: key threshold around .4-.6, key soft around .1-.2 (try things out and see how it works on your camera
now draw BLOCK_3
B_2 parameters
color eq on, HSB/RGB set to RGB
play around with mainly band 4 and 5. Because my camera was getting kind of caught up in bluish tones, i set things the following way
band 5 r .12, g .1, b -.36
band 4 r -.3, g -.38, b .06
band 3 r .3, g, .26, b .02
but depending on how your camera sensor works you might want to try something totally different!
if you haven't played around much with filters and camera feedback, head back into BLOCK_2 and play with input adjust blur and sharpen. notice how blurring with different radius can sometimes clean up the keying
back into BLOCK_3
final mix and key reset.
try out mix types difference and multiplicative, values between -.5 and .5, with overflow clamp and then with overflow foldover

What did you learn?
- Using Draw All BLOCKS to help with mixing BLOCK_1 and BLOCK_2 into BLOCK_3
- Using input adjust filters on camera feedback
- Using color eq RGB mode on camera feedback
- Mixing camera and internal feedback

Further experiments with this patch

Section 4 Glossary & Reference

GLOSSARY

GENERAL VIDEO TERMS

NTSC/PAL: These were the broadcast video standards for most of the world during the analog era. NTSC has a resolution of 720x480 pixels with a pixel aspect ratio of>>> and frame rate of 60hz and PAL has a resolution of 720x575 pixels pixel aspect ratio of <<< and frame rate of 50hz. Technically these are interlaced fields per second, not frames, and the rates are very (very!) slightly slower than those numbers but I'm making the executive decision to say that those facts are getting too far in the analog video weeds for this manual

TIME BASE CORRECTOR (TBC): Video signals need some method of informing monitors (either CRTs or LCDs) when a new frame begins and where exactly each pixel is supposed to go

HDMI : HDMI is technically a protocol for a video cable connector

OVERSCAN: Video monitors (both CRT and LCD)

CRT: Cathode Ray Tube. The old school bulky and kind of loud display technology.

LCD: Liquid Crystal Display. An LCDtv is an LCD display that also has video decoders for analog video inputs & digital broadcast signals.

Video Synthesis: Video which is generated primarily through manipulating video signals via some kind of electronic methods. People can be quite touchy and precise about where the line begins and ends between Video Processor and Video Synthesizer. The answer to the question: "Is GW a Video Synthesizer" is pretty up in the air.

Video Feedback: A two dimensional feedback system in which each successive frame is generated using at least some information from the previous frame. Camera feedback is when you plug a camera into a monitor and then point the camera at that monitor. Internal Feedback is what happens when you plug the output of a video mixer back into an input. Framebuffer feedback is when you store a frame in a system with digital memory and simply use some of the data from a previous frame in order to generate a new frame. GW is inspired by the kind of Internal Feedback that folks enjoy using on other hardware video mixers, but uses Framebuffer feedback so that you don't have to choose between live sources and feedback sources.

PIXEL: a pixel is one discrete unit of color information along with (x,y) indexing to locate it within a frame. In GW, all pixels are ultimately processed as vectors of (Red,Green,Blue), though at times it is creatively and intuitively more useful to convert them into parameters of (Hue,Saturation,Brightness).

FRAME: a frame is one discrete unit of video information in time. A frame consists of a grid of pixels.

FRAMEBUFFER: a framebuffer is some method of storing frame information that isn't just immediately getting dumped out onto a monitor somewheres.

FRAME RATE: the number of times frames get updated per second. The most common frame rates are 30, 60, 25, 50, 24, and the PAL/NTSC ones. GW output frame rate is 60fps but processing speed is at 30fps, meaning every other frame is identical.

FILTER: colloquially one refers to just any kind of video fx as a 'filter', however in GW we use the term in more of a strict image processing sense. A filter is an operation on an entire frame of pixels that, for each pixel in the frame, requires random access to some neighborhood of pixels in order to calculate the value of output pixel. This is a potentially obnoxiously mathematical way to describe it so lets use examples to illustrate what is and is not a filter. A blur is a filter because, for every pixel, we calculate its average with a bunch of neighboring pixels in a grid (diagram). RGB invert is not a filter because, for every pixel, we only need to use the value of that input pixel to calculate the output.

ASPECT RATIO

Commonly written as 16:9 or 4:3. The ratio of the resolution Width to Height. GW output resolution is 1280x720, if you take the fraction 1280/720 and reduce to lowest terms you get 16/9, or 16:9 aspect ratio. Generally speaking, modern HD (hdmi) digital video signals are in some kind of 16:9 format. Aspect ratio (or sometimes DISPLAY aspect ratio) typically only refers to an entire frame, however the PAL and NTSC video signals that come into GW will also have a non square Pixel aspect ratio to consider.

RESOLUTION

The resolution of a video signal is the number of total pixels in each frame written out like "number of horizontal pixels" x "number of vertical pixels". GW output resolution is 1280x720. The possible input resolutions for SD analog video are NTSC 720x480 and PAL 720x576. The possible input resolutions for HD and/or Digital video signals is much much wider. GW samples all input video at 640x480 30fps and upscales everything to be processed at 1280x720 30fps. Feedback framebuffers are all processed directly at 1280x720.

FRAME RATE: The number of times that a new frame is updated, per second. 24, 25, 30, 50, and 60 are the most common. GW processes video at 30fps but outputs 60fps.

PIXEL OPERATIONS
color, geometry

FRAME OPERATIONS
filters

MULTIPLE FRAME OPERATIONS
mixing, keying

MIX AND KEY
For this section, lets consider two video sources named A and B. For most of the mixing and keying techniques concerned, order matters, so using the same notation as in the GUI we will assume that we are mixing A->B. If we think in terms of how layers work in image processing programs, A is a layer on top of B.

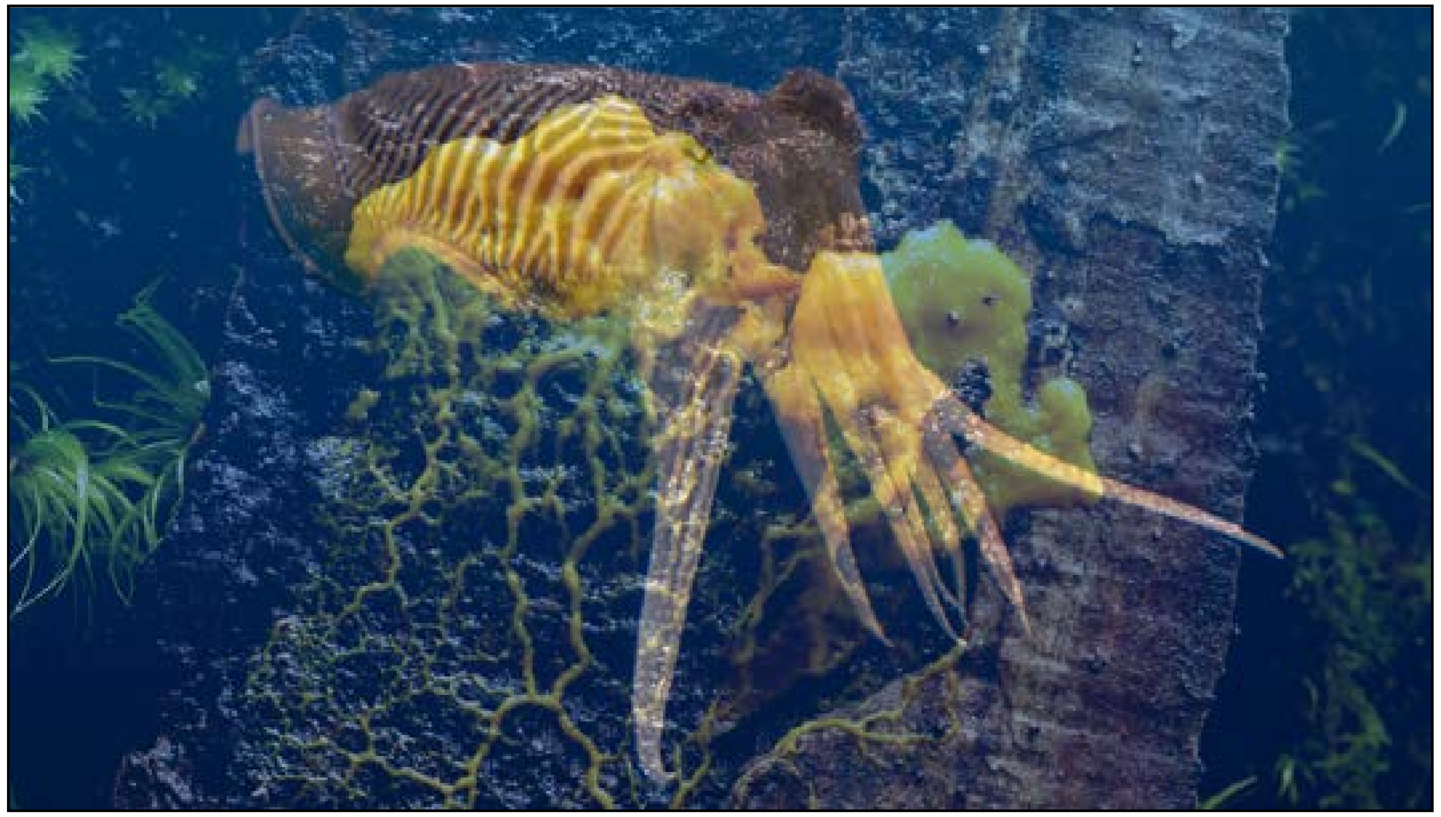
(slime mold is A, cuttlefish is B)ß

mix type: (also called Blend Modes in image processing terminology) Mixing specically refers to combining two video sources (A and B) in such a way that every single pixel of both A and B is combined in the exact same way. For the examples we will notate mix amount as M. All mixing is done directly in RGB space

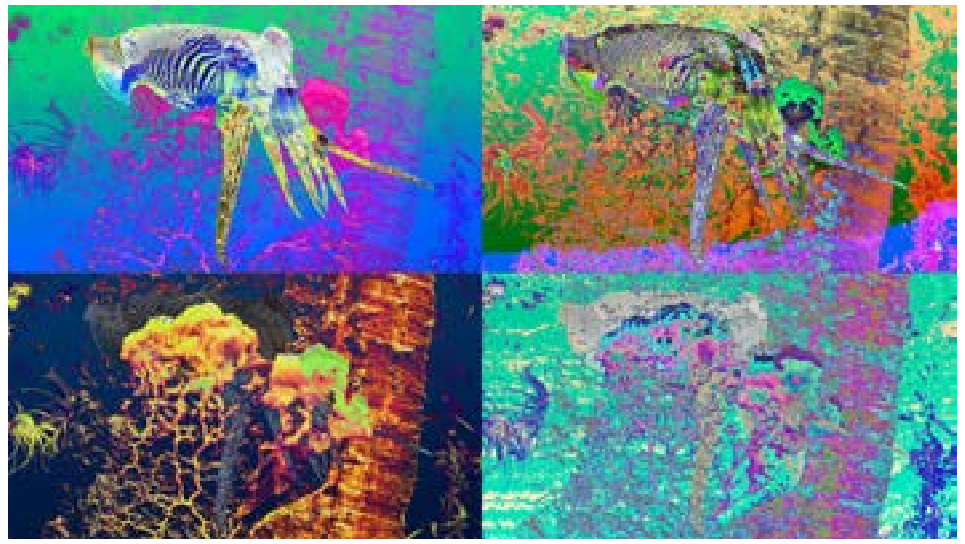
overflow: at the output stage, all pixels are considered RGB vectors between (0,0,0) and (1,1,1). For the purposes of fun, many of these mixing operations can result in values that go below 0 or above 1. Overflow refers to how we deal with these 'overflowing' values. (DIAGRAMS)
CLAMP: everything outside of (0,1) gets squeezed back into 0 if negative or 1 if >1.

wrap: values that go below 0 wrap around to come down from 1. EG -.2 ->.8. values that go over 1 wrap around to pop up from 0. EG 1.2 -> .2. Harsh and topographical.

foldover: values below 0 fold back upwards through 0. EG -.2 -> .2. values over 1 fold back down through 1, EG 1.2->.8.


(linear fade mix at .25)

linear fade: a linear interpolation of M between A and B is calculated. The math is f(x) = (1-M)*A+M*B. Imagine that we have a straight line between two points A and B, M refers to where exactly we are on that line. Note that this only makes intuitive sense if we have a value of M between 0 and 1, however in GW our mix range is from -2 to 2 so any values below 0 or above .5 will be distortions.



(captions for these 4 squares read Top Left, Top Right, Bottom Left, Bottom Right)
(1 foldover, 1 wrap, -.25 foldover -.25 wrap)

(additive mix at .5)

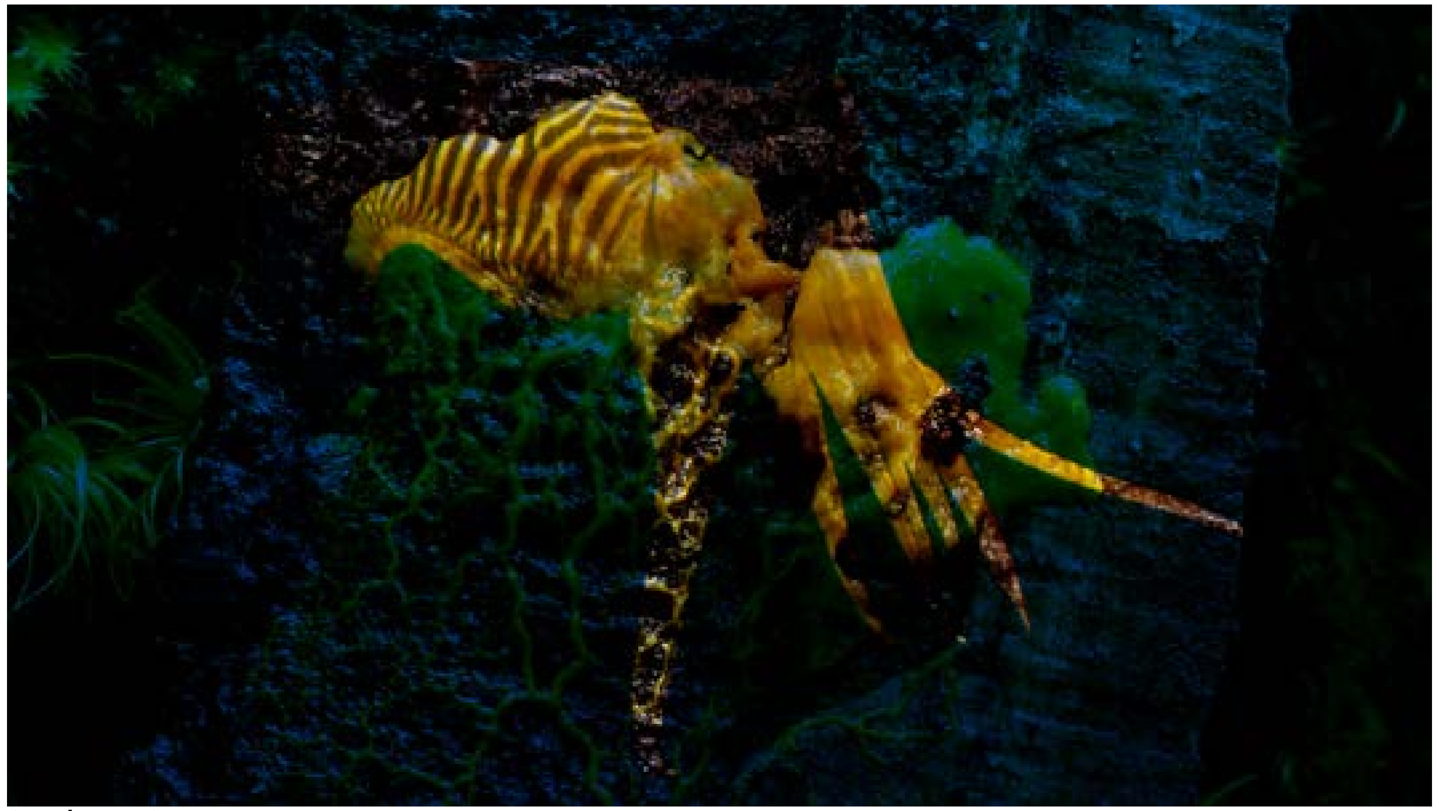
additive: The sum of A and B weighted by M is calculated. The math is f(x) = A + M*B. Note that for negative values of M, this will subtract B from A.

(upper left to bottom right: -.5 clamp, -1 foldover, -1 wrap, 1 wrap)

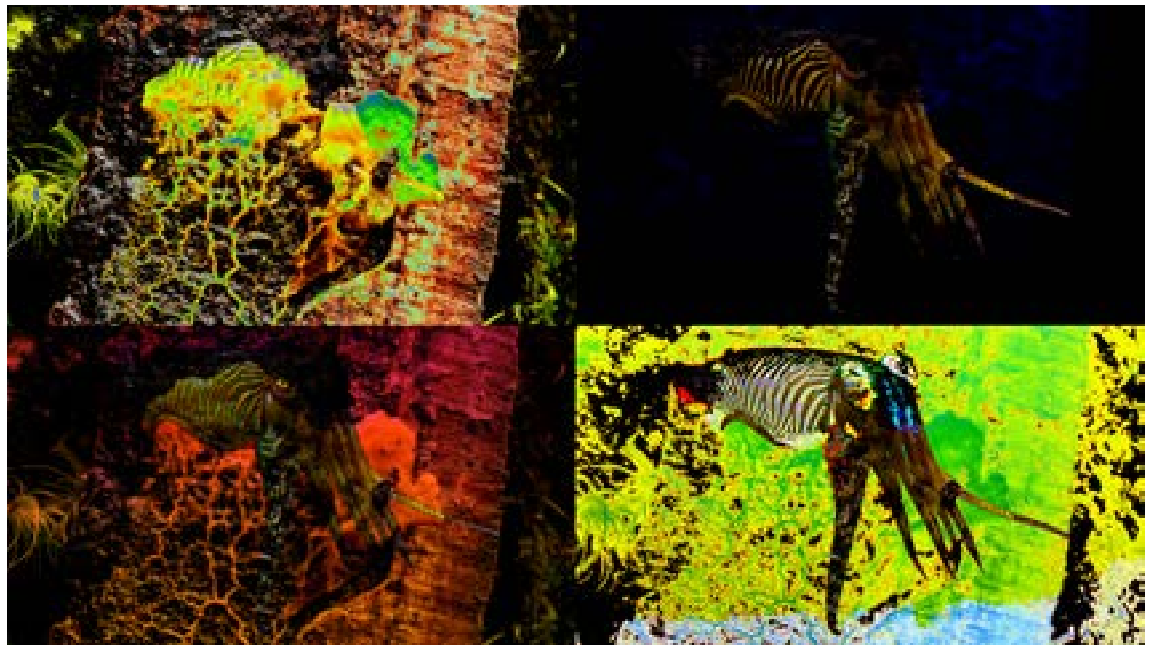
(difference mix at .5)

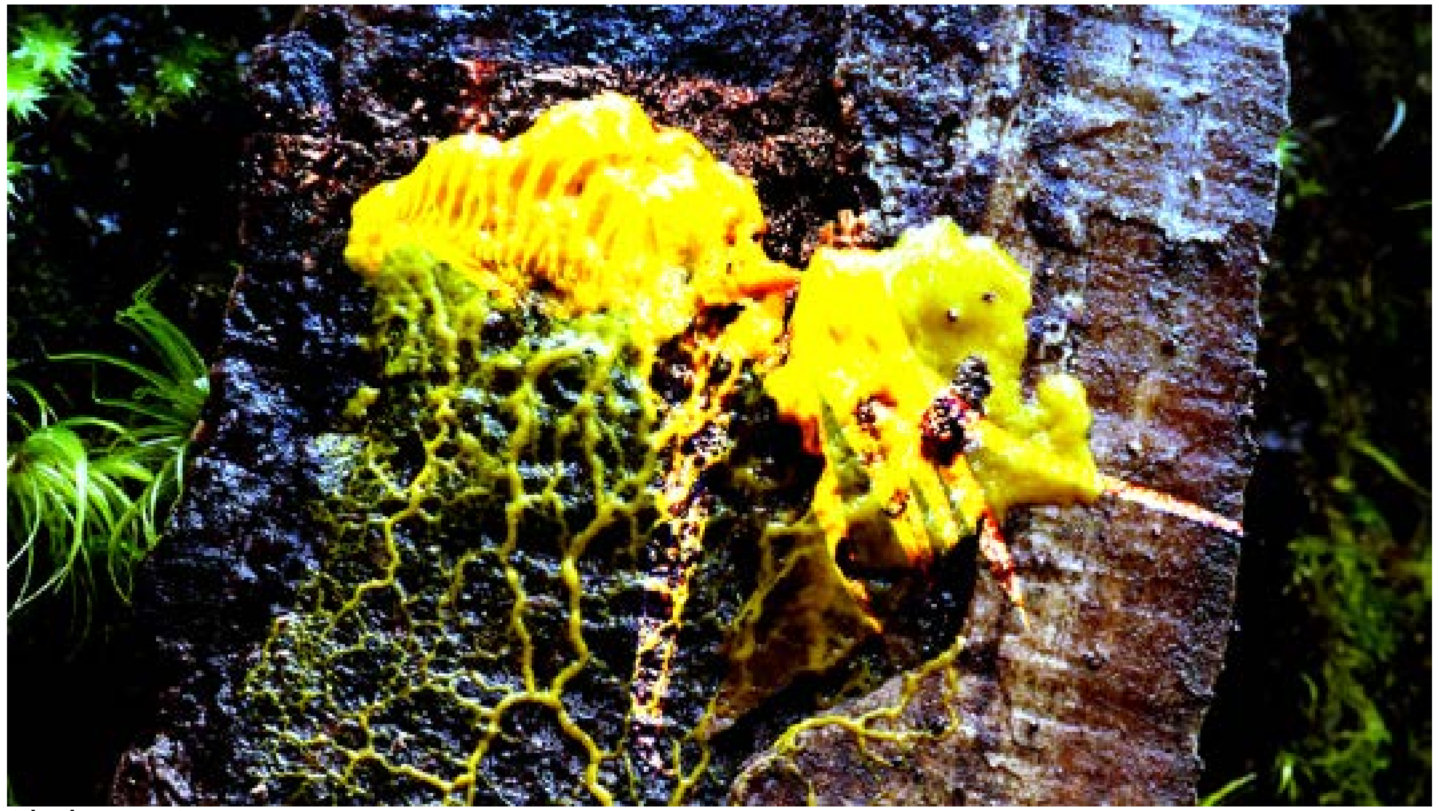
difference: The absolute value of B weighted by M subtracted from A is calculated. the math is f(M) = abs(A-M*B)

(-.5 fold, -.5 wrap, 1 fold, 1 wrap)

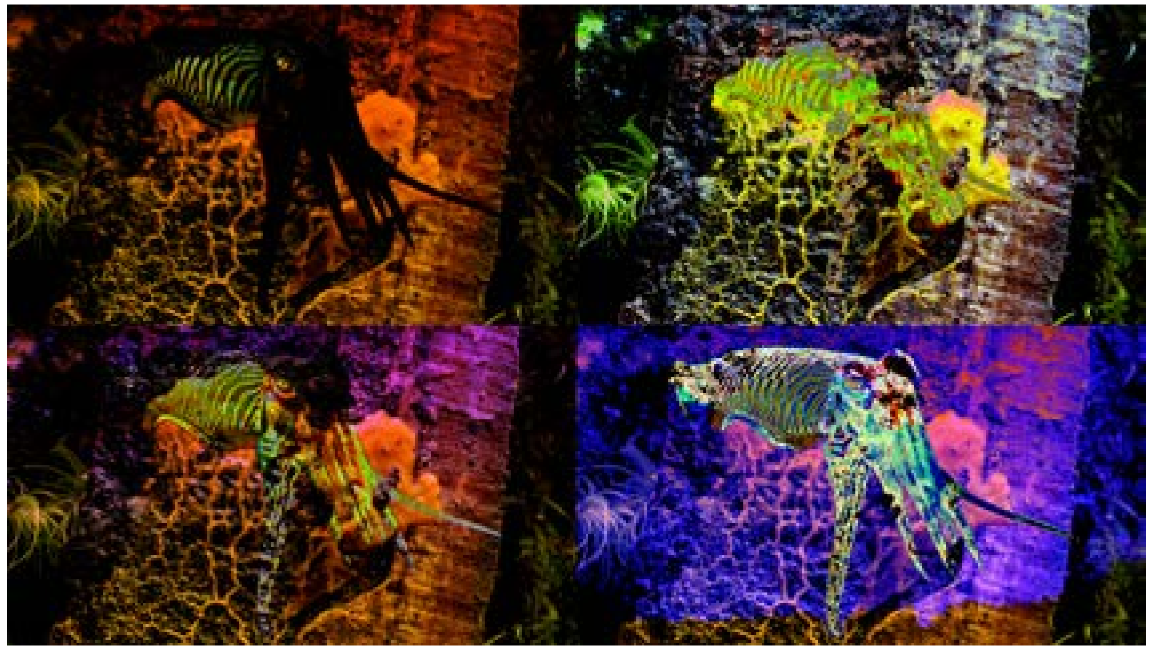

(mult at .5)

multiplicative: The output is interpolated between A and A*B using M. The math is f(M) = (1-M)*A+M*(A*B)


(-.5 foldover, 1 clamp, 1 foldover, 1 wrap)


(dodge at .5)

dodge: The output is interpolated between A and A/(1.0-B) using M. The math is f(M) = (1-M)*A + M*(A/(1.0-B))


(-.5 clamp, .5 foldover, -.5 foldover, -.5 wrap)

(lumakey with value black and threshold .26)

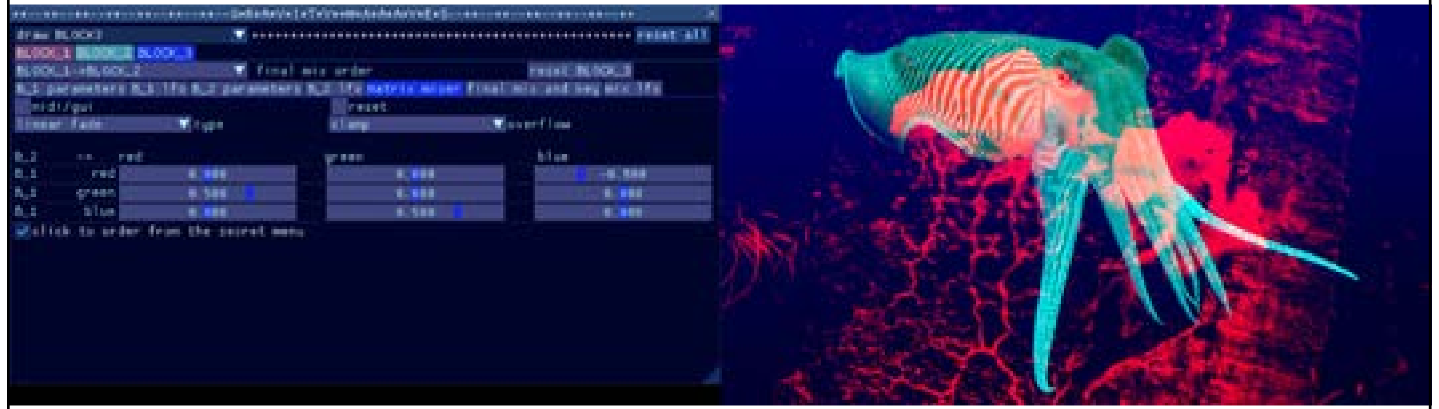
key/mix order: by default we are mixing and keying A->B. changing key order takes B->A

key type: keying is choosing between two video sources using some kind of logical operation based on the value of each pixel in A. The chromakey uses any (R,G,B) value while the lumakey uses only (R,G,B) values where R=G=B (i.e. greyscale). When lumakey is selected, changing any single one of R G B will affect them all. When chromakey is selected you can adjust R, G, and B seperately to choose any color. When A->B we can think of A as being 'on top' of B, and keying is removing pixels from A entirely so that we can see pixels from B underneath.

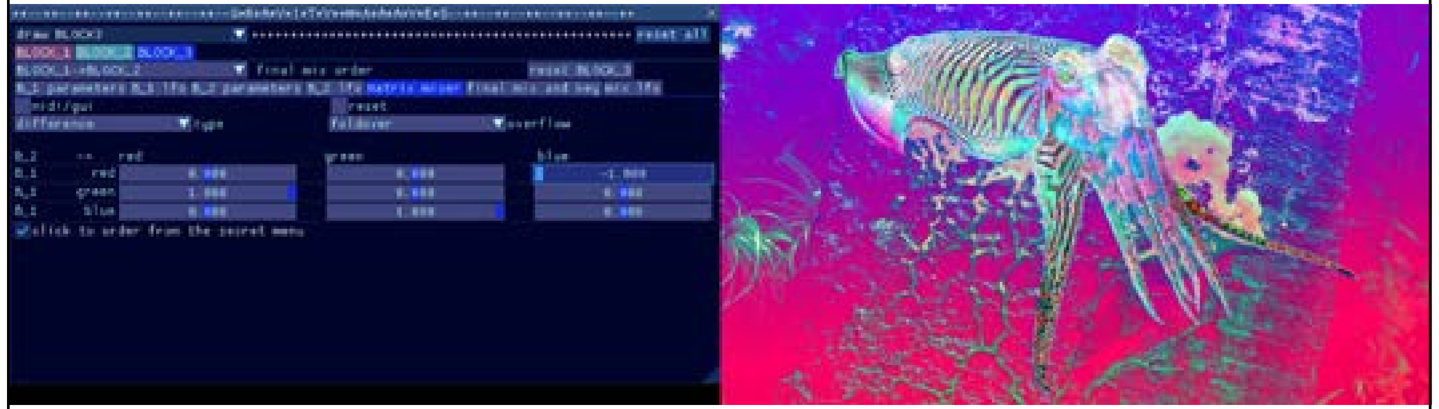
key value color: changes to show you what color has been selected by adjusting *key red, key green,* and *key blue.*

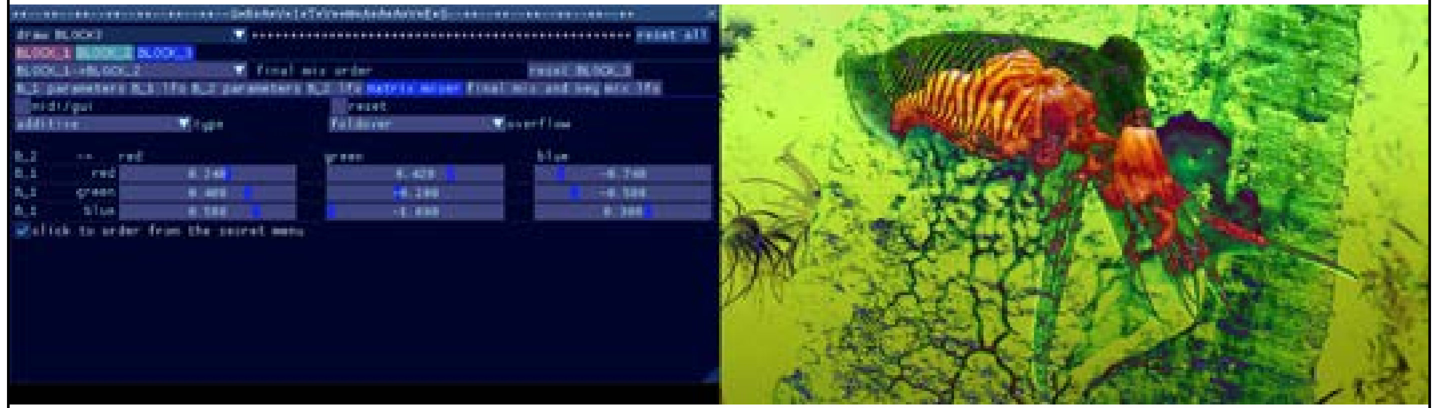
key threshold: selects how much area in the color space around the key value to remove. If *(key red, key green, key blue)* is (0,0,0) = black, and *key threshold* is set to .25, that means all pixels with luma values betweeen (0,0,0) = black and (.25,.25,.25) = charcoal grey will be keyed out. If the key value is (0,1,0) = green, and key threshold is at .25, then all pixels with fully saturated green down to lighly toned green, as well as some values of saturated greenish yellow and greenish blue will all be keyed out.

key soft: adds a linear fade between A and B weighted by the brightness value of pixels in A in the parts of the video keyed out.



MATRIX MIXER:
Every pixel in each frame can be described as a set of Red, Green, and Blue components. What the matrix mixer does is allow you to directly control how much each of the individual Red, Green, and Blue components of every pixel in each frame are combined.



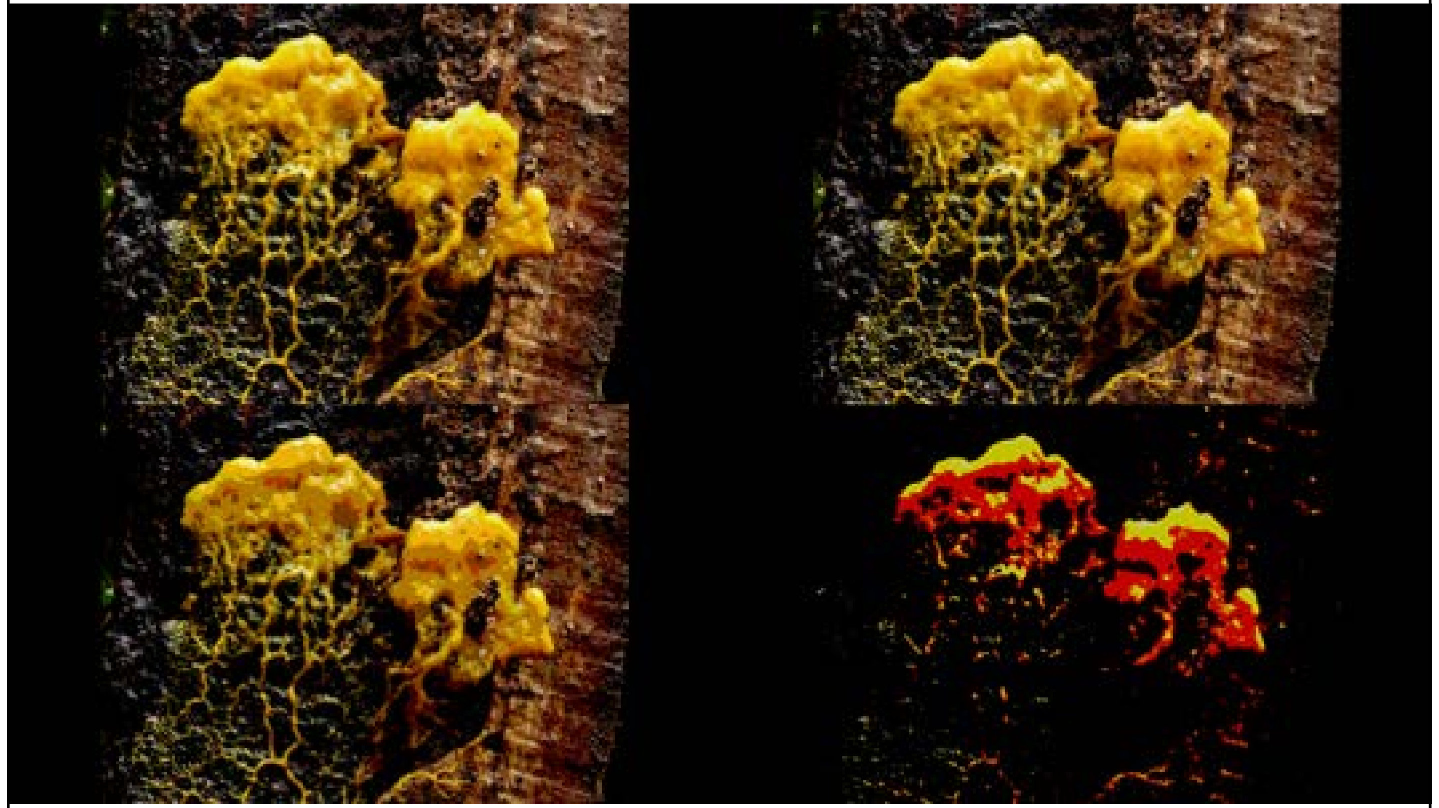


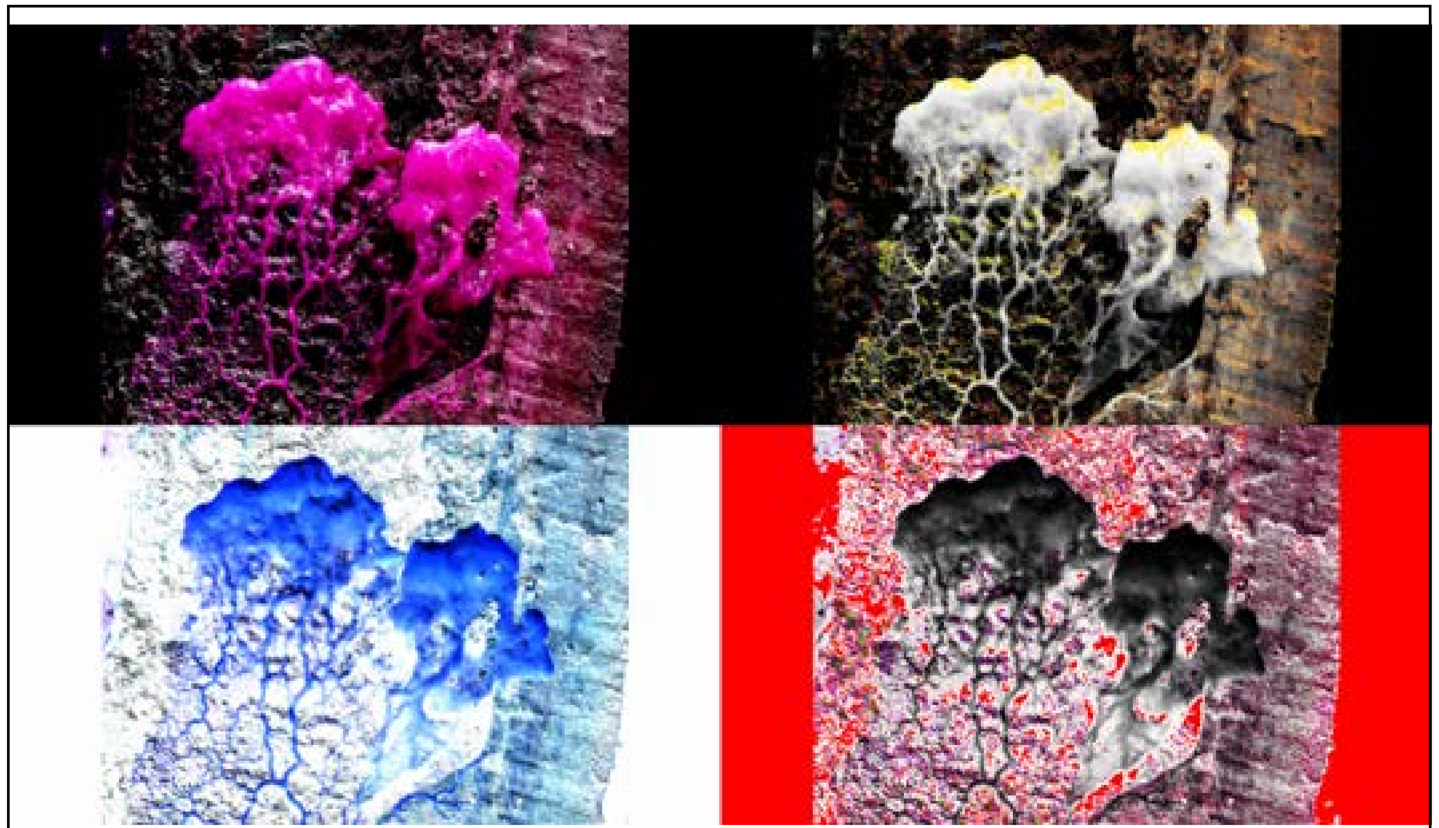
COLOR
for each of these the parameter value will be notated with P

Values of HUE always wrap around.
Values of Saturation and Brightness always clamp at 0 and 1.

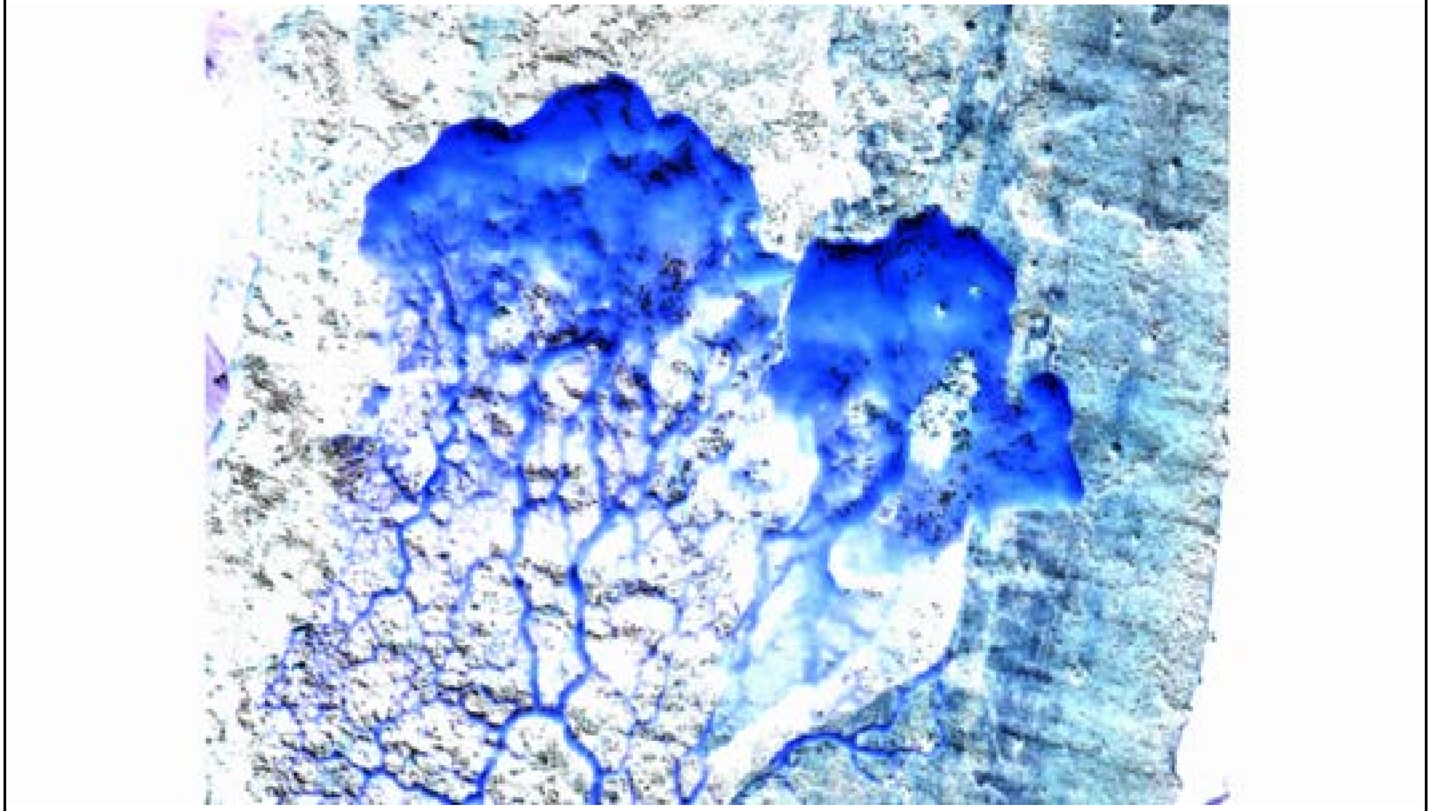
DIAGRAMS FOR EVERYTHING
hsb ++: offset or pedestal. The math for calculating hue would be f(P)= H+P. Values are simply shifted up or down
hsb **: attenuation. The math for calculating is f(P)=H*(1+P). The entire range of values are scaled up or down
hsb ^^: power mapping. The math for calculating is f(P)=H^(1+P). Values are shifted differently depending on the magnitude of each value.

hue shaper: Sends hue into a shaper function. The math is f(P) = wrap(abs(H) +P*sin(H/3)). Helps with guiding more chaotic hue cycling patterns outside of standard rainbow business.

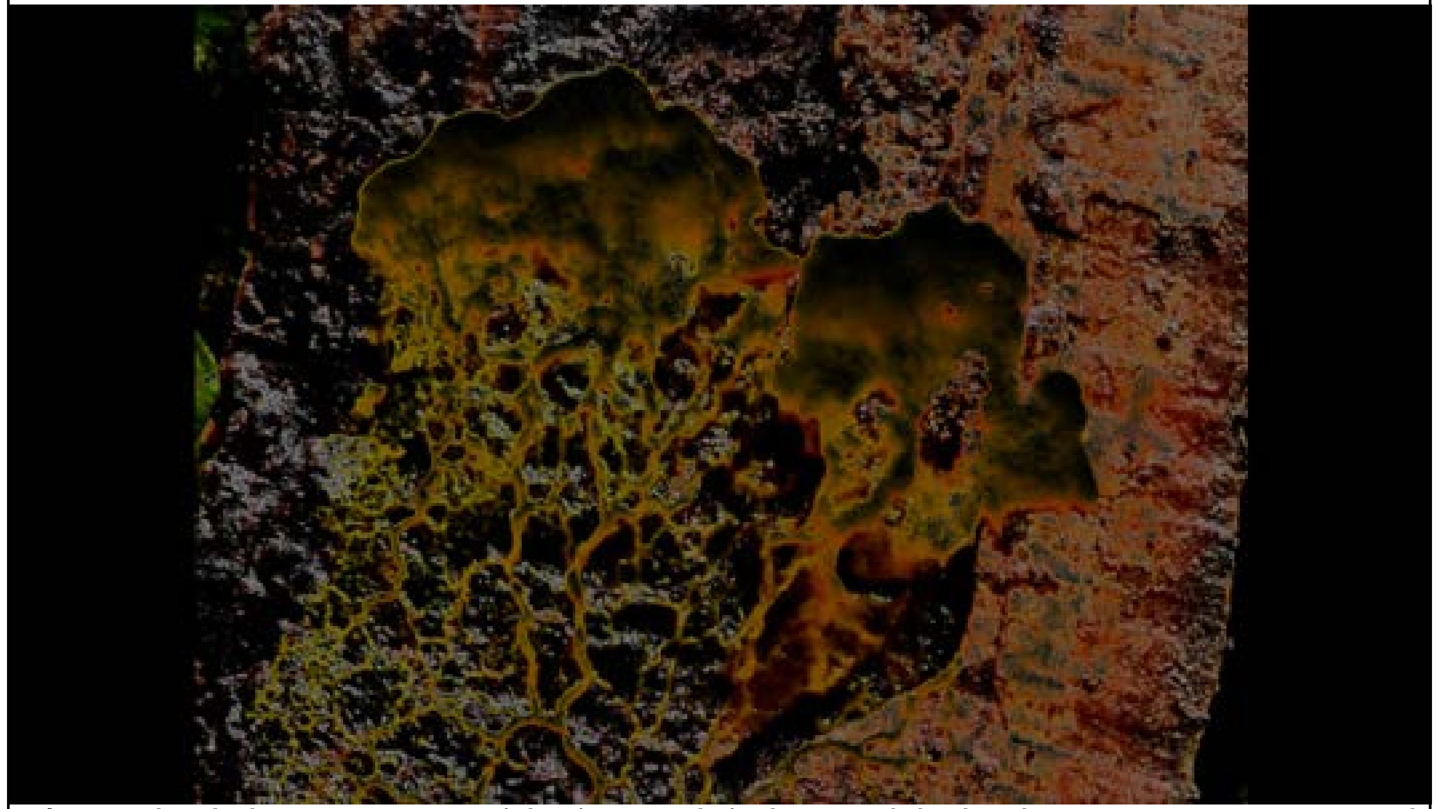

(.2, .5, .78, .98)
posterize: quantize the colors in RGB space.


(hue invert, sat invert, bri invert, all hsb inverted)

h/s/b* invert:** the individual channel is inverted. Note that inverting all of ***h, s, and ***b*** is NOT the same operation as ***rgb invert*** and will definitely look different.



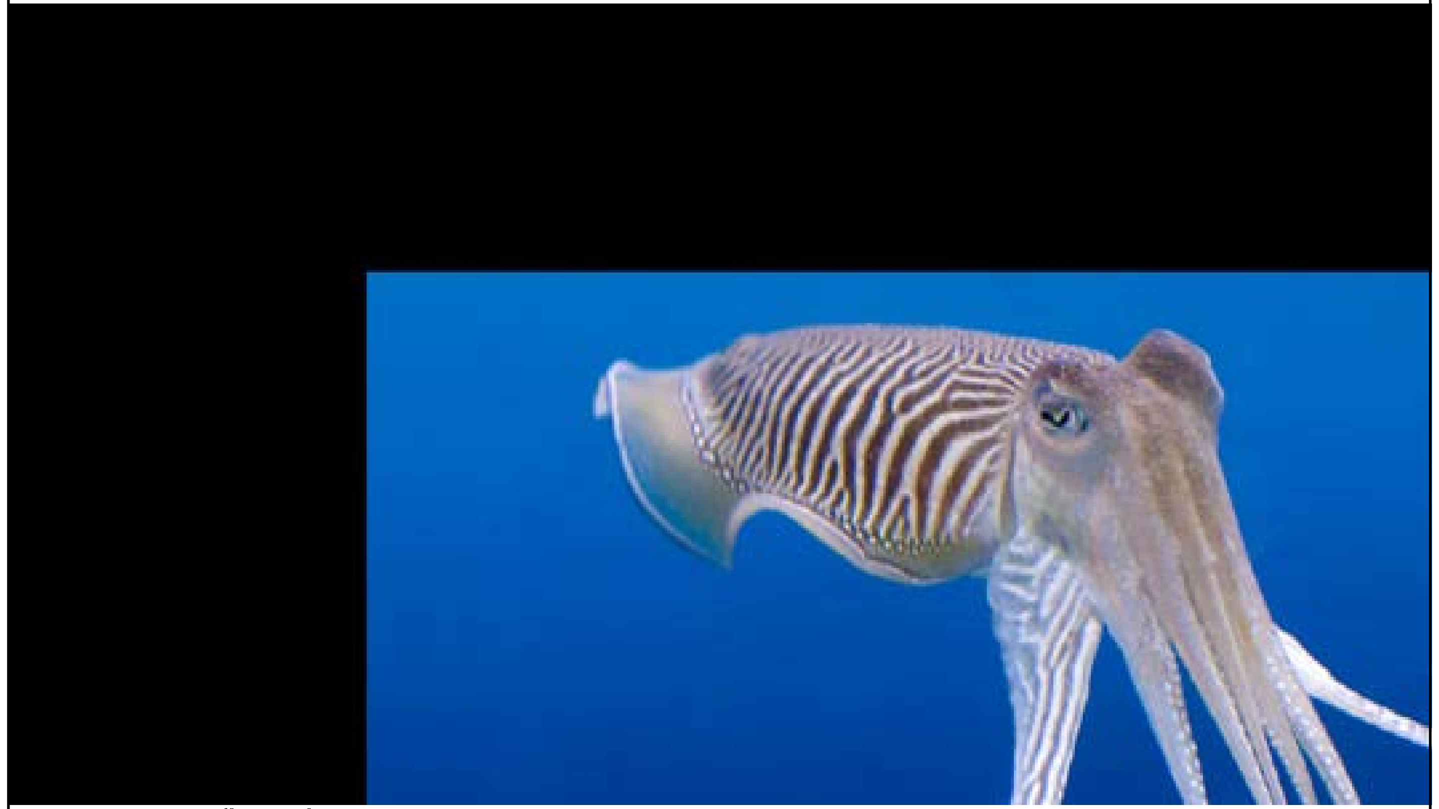
rgb invert: the red green and blue values are all inverted. All color information is inverted, which can make for a cleaner look in some situations.



solarize: the darkest 50 percent of the frame is left alone and the brightest 50 percent is inverted.

ORDER OF OPERATIONS
HUE SHAPER -> OFFSET -> ATTENUATION -> POWER MAPPING -> INVERT/SOLARIZE -> OVER-FLOW -> POSTERIZE

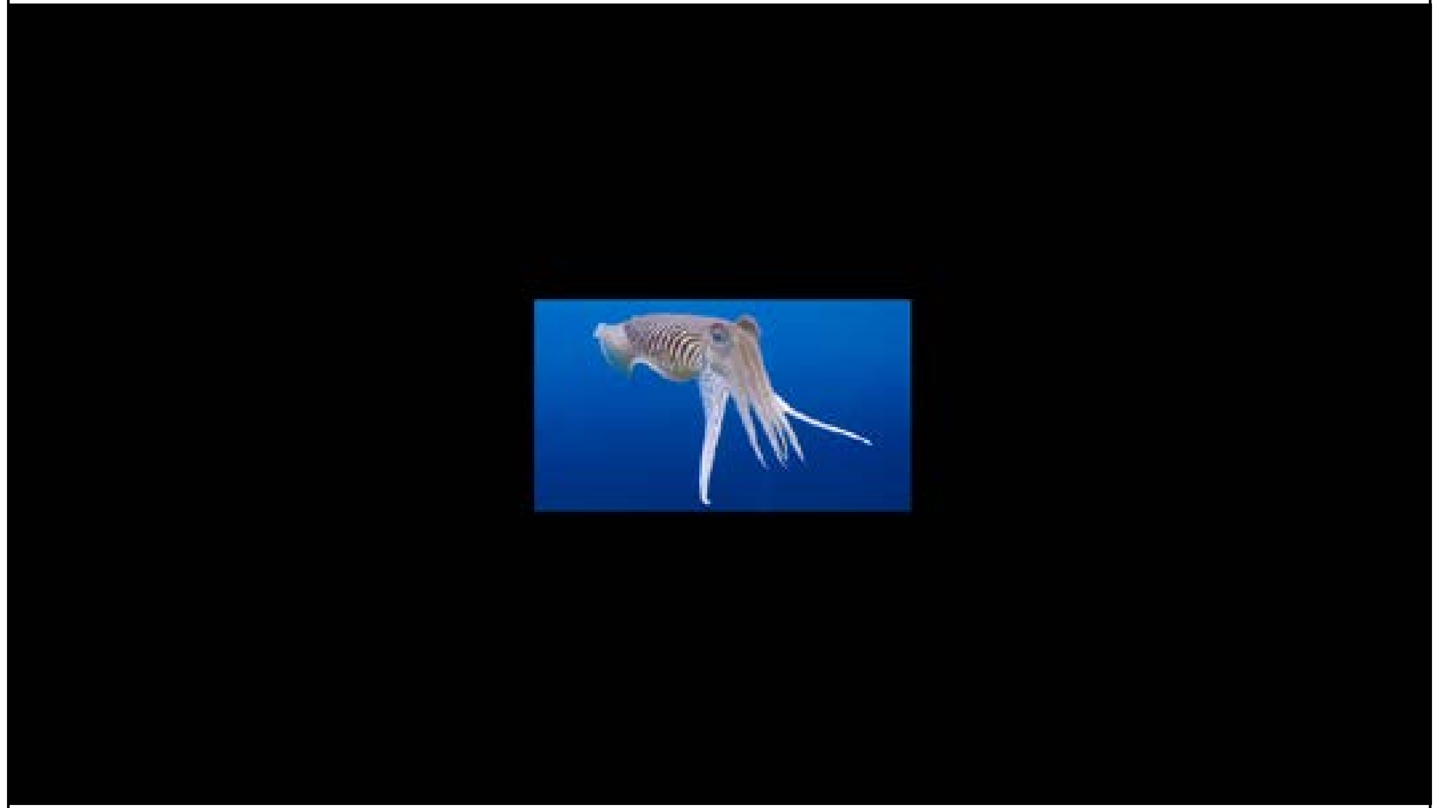
GEOMETRY
We will use this image of the cuttlefish for examples of how geometry operations work



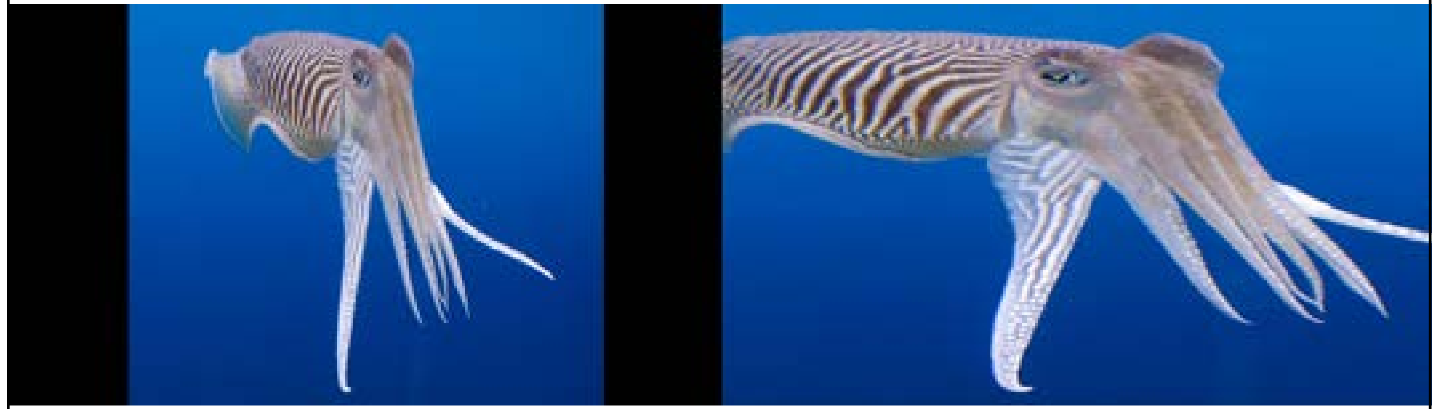
(x .5, y -.5 overflow clamp)

x <->: x displace. moves the entire frame to the left or right. changing the x position affects the vanishing point of z displacement, the center of rotation, and the center point of x stretch

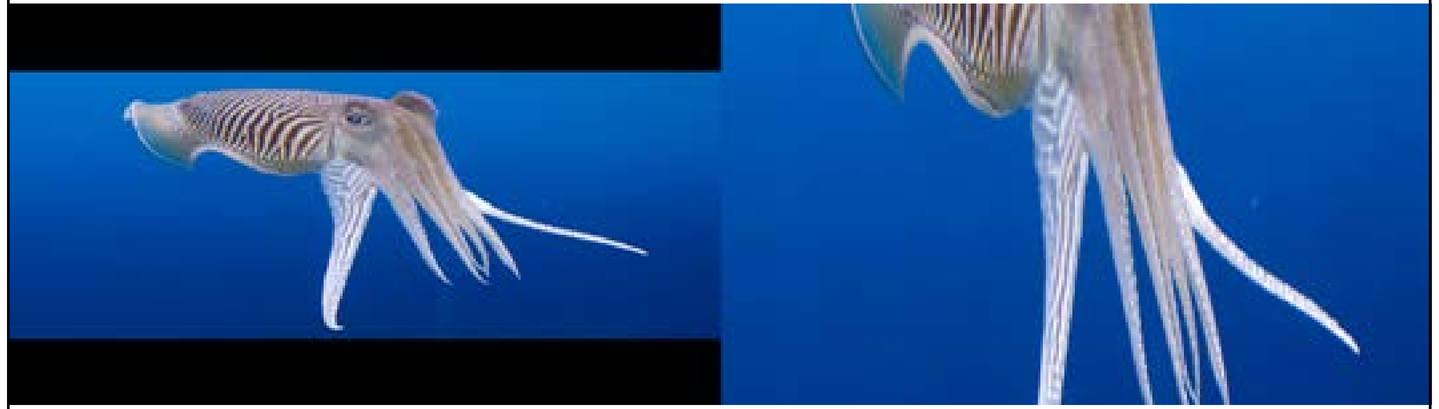
y <->: y displace. moves the entire frame up or down. changing the y position affects the vanishing point of z displacement, the center of rotation, and the center point of y stretch

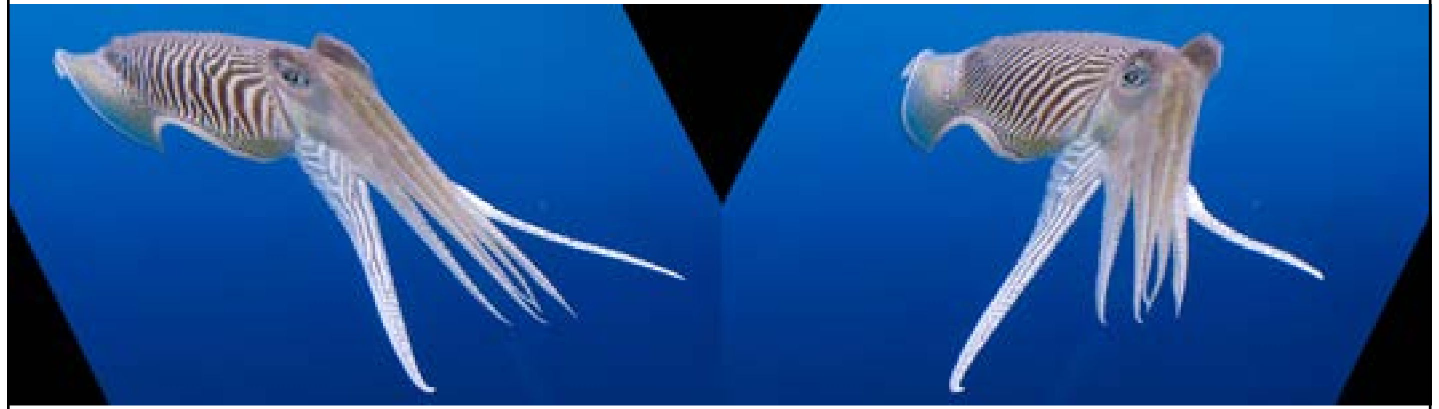

(z at .240)
z <->: z displace. shrinks or blows up the frame.

rotate: rotates the entire frame around wherever x and y displace are at. each rotate mode distorts differently?

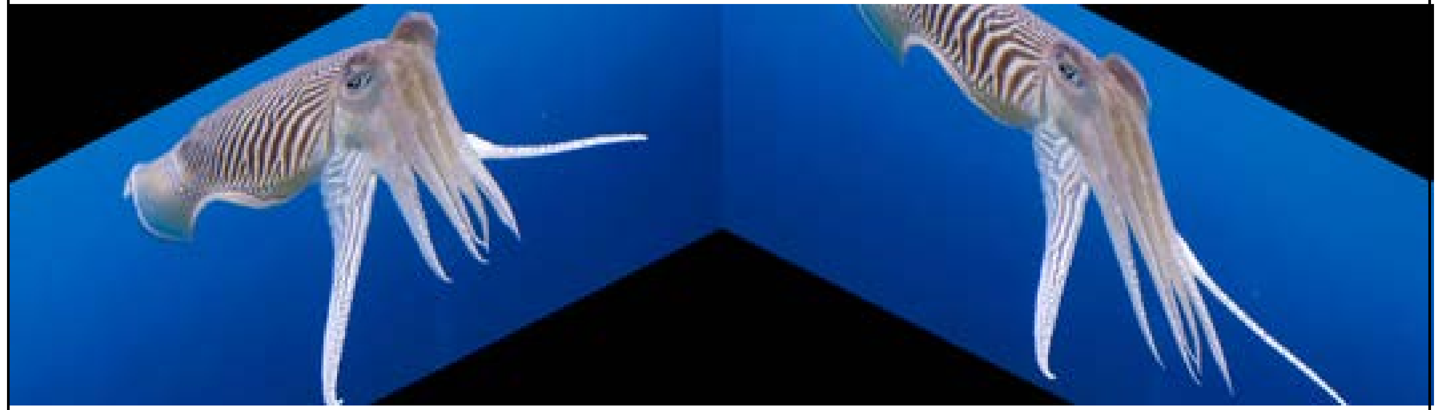

(x stretch at .5 and -.5)

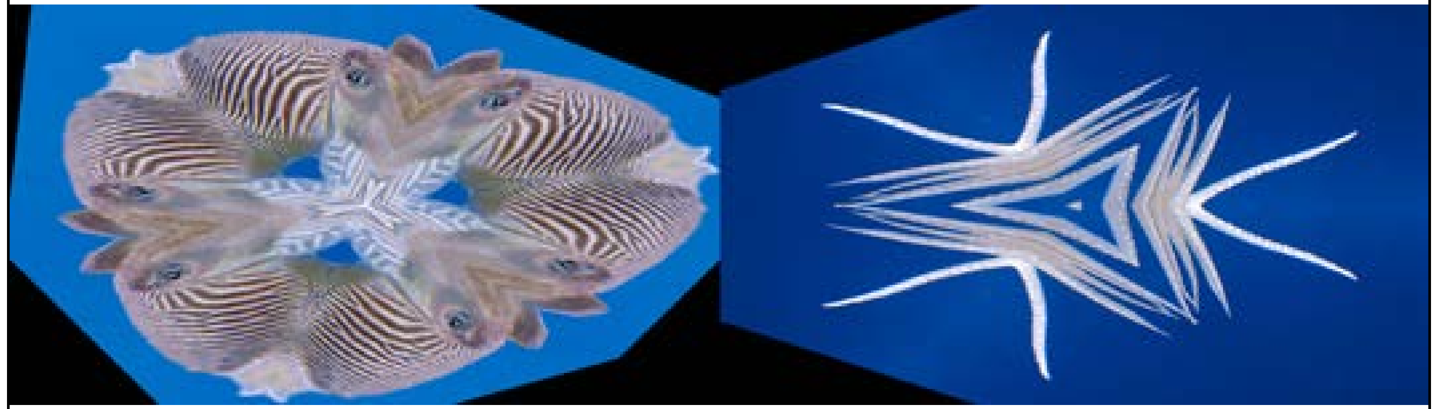
x stretch: positive values squeeze and negative values stretch the framebuffer along the x axis.


(.5, -.5)
y stretch: positive values stretch and negative values squeeze the framebuffer along the y axis.

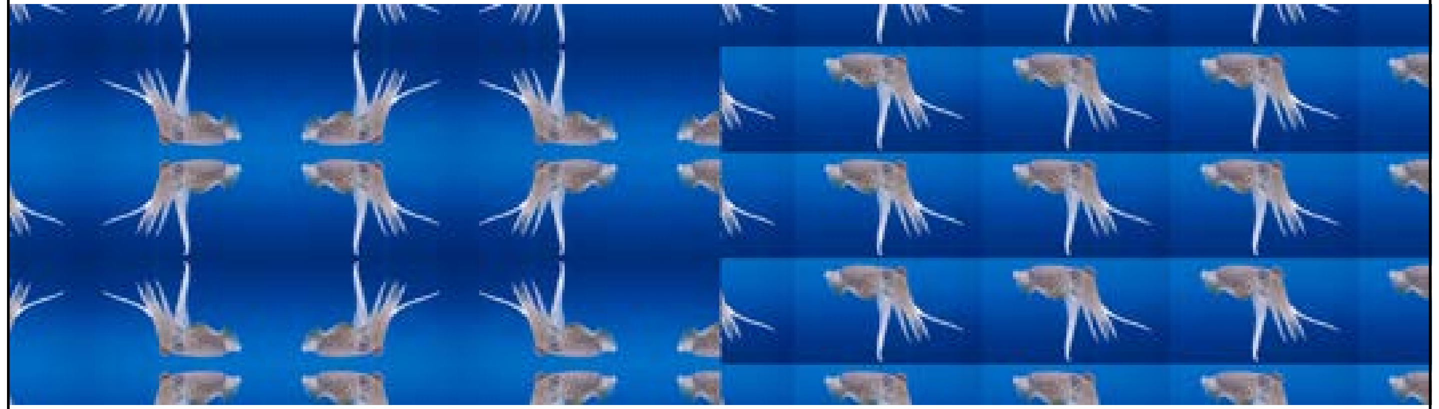

(.5, -.5)
x shear: the y position of pixels remains unchanged but x value gets displaced at a relative

amount to its y position. postive displaces to the right, negative to the left.


(.5, -.5)
y shear: the x position of pixels remains unchanged but y value gets displaced at a relative amount to its x position. postive displaces up, negative down.


(1. ka .16, slice .740; 2. ka .16, slice 0)
kaleid amount: simulates the effect of viewing the buffer/camera through a kaleidoscope. amount changes how many angles of reflection there are

kaleid slice: changes the portion of live video that is reflected through the kaleidoscope effect

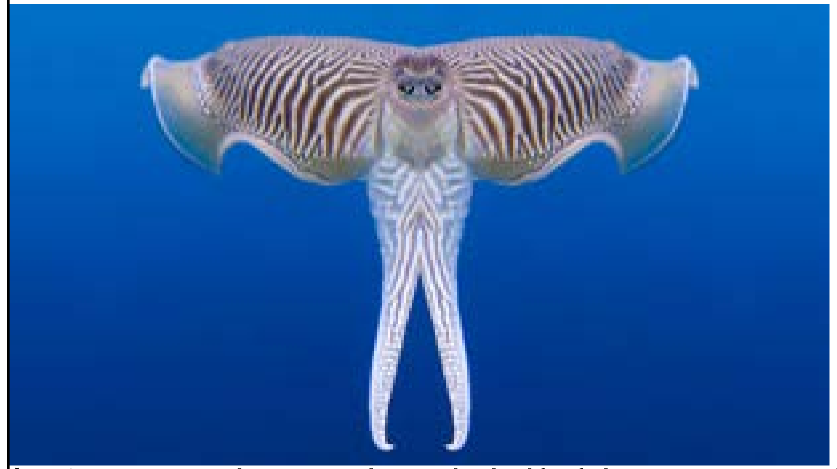


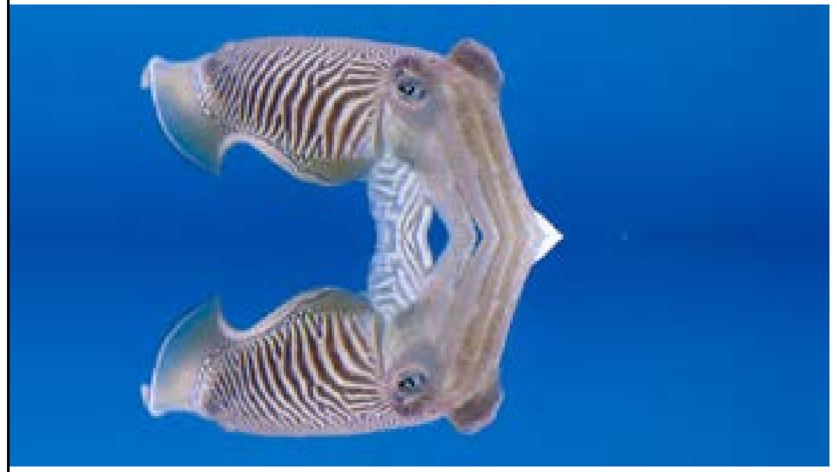
GEO OVERFLOW
overflow in this case is concerned with what happens when the the (x,y) coordinates of the framebuffer/live video are shifted 'off of the screen'

clamp: video is blanked out

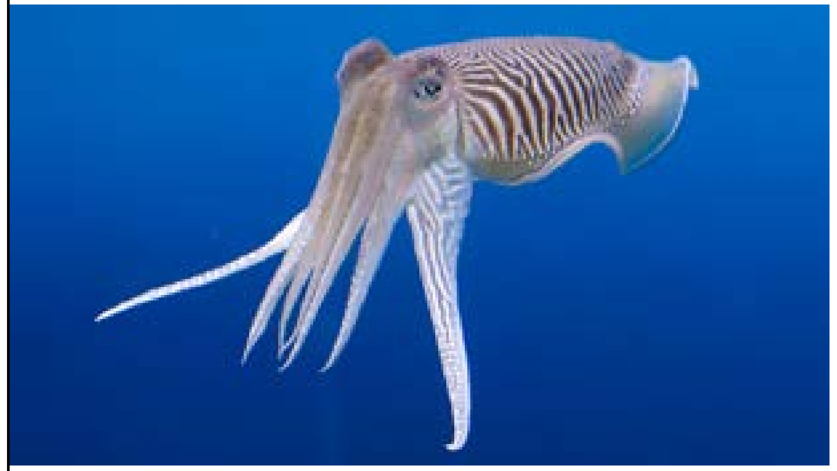
wrap: also known as toroidal universe, things move off the screen on the left x side and pop up back on the right x side, and vice versa. things move off the top of the screen, show up on the bottom, and vice versa

mirror: things are reflected coordinates overflow

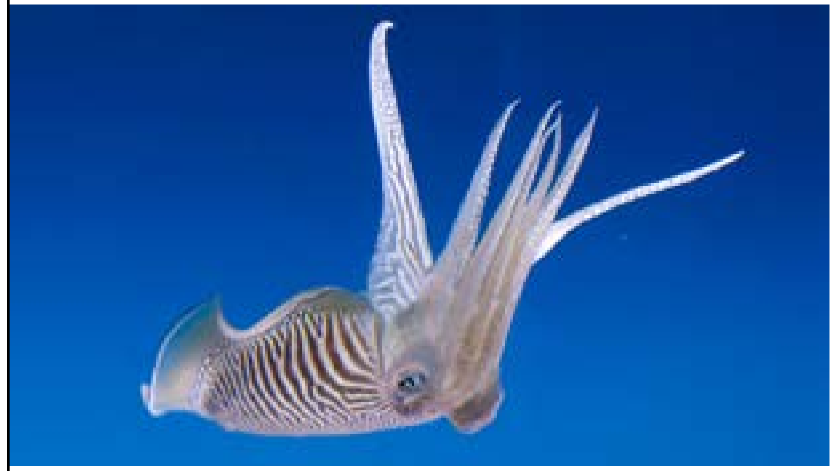

h mirror: everything on the right half of the screen is replaced with a reflection from the left half of the screen



v mirror: everything on the bottom half of the screen is replaced with a reflection from the top half



h flip: all x coordinates are reversed

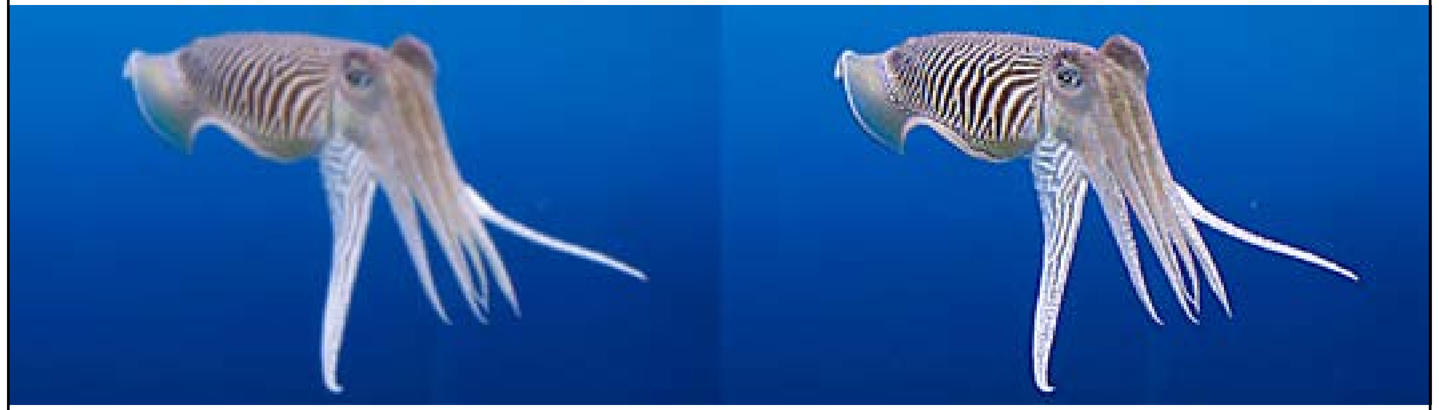


v flip: all y coordinates are reversed

ORDER OF OPERATIONS

flip->mirrors->kaleidoscop->XYdisplace->Zdisplace->rotate->stretch & shear->overflow

FILTERS

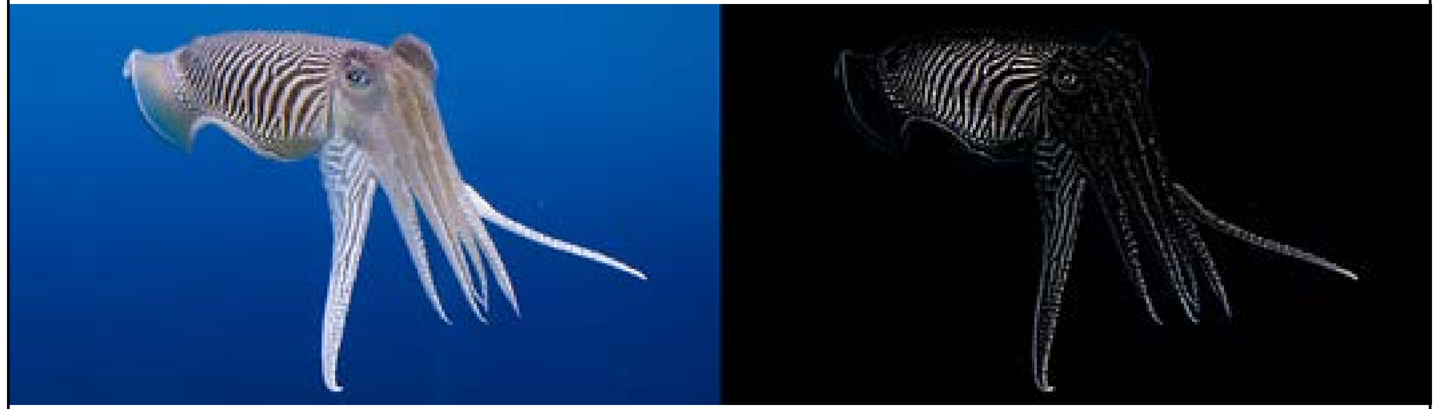


(amt 1 rad .2; amt -1 rad .6)

BLUR: performs an average on each pixel with a neighborhood selected by radius. Blur is performed directly in RGB

blur amount: linear fade between original and blurred video. Negative values for blur amount will perform a sharpen in RGB.

blur radius: selects the radius of how far away the neighborhood is. at 0 this is 1 pixel away, and 1 this is 10 pixels away



(amt .5 rad .26 boost 0; amt 1 rad .5 boost .84)

SHARPEN: accentuates areas of high contrast by taking each pixel and subtracting a weighted value of the sum of neighboring pixels. Sharpen is peformed in HSB and primarily affects Brightness and Saturation. It is necessary to use filter boost along with larger values of sharpening, *sharp amt:* controls how much sharpening is applied to the brightness channel. Large values perform edge detect

sharpen radius: selects how far away the neighborhood is. 0 is 1 pixel away, 1 is 10 pixels away

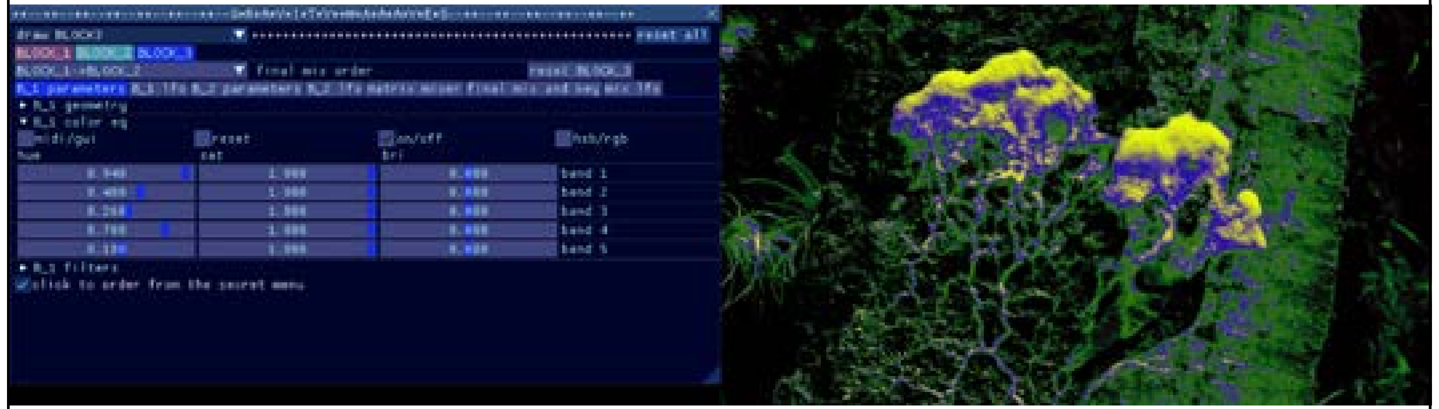
filter boost: at extreme values of blur or sharpen amount, a significant amount of brightness and saturation is lost. filter boost can be used to bring a little bit (or a lot of bit) back into the output.

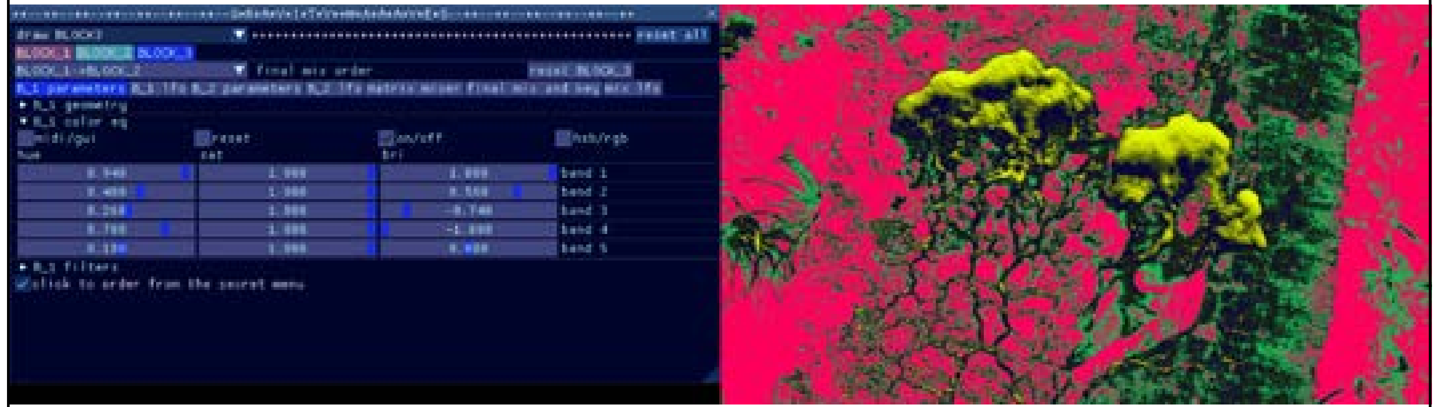
TEMPORAL FILTER: a linear fade between the current frame and the previous frame. NOTE that temporal filters will affect the output video no matter if fb1/fb2 is mixed or keyed in. Temporal filters are difficult to capture in a screenshot as they filter in TIME moreso than SPACE.

temporal filter amount: the amount of fading. Values between 0,.5 will look like an afterimage/ trails style effect, .5,1 will be clamped out looking digital feedback, and between 0,-1 will look a bit like a strange, colorful, mildly strobing sharpen.

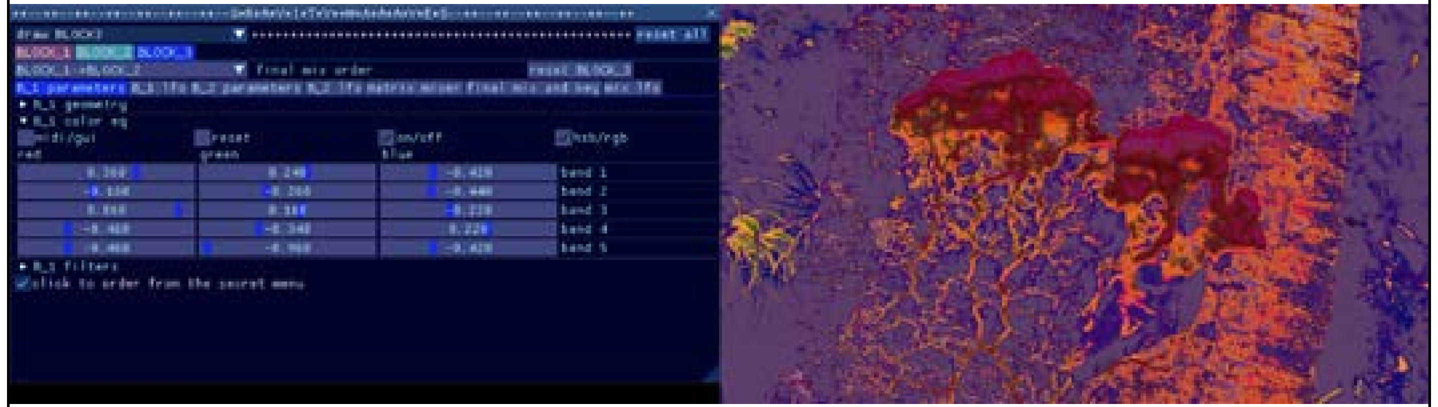
temporal filter q: the amount of amplification of the brightness and saturation in the temporal filter.

COLOR EQ
What makes Color eq different from the color controls covered earlier? HSB based offset, attenuate, and powmapping are centered around tweaking the existing colors of a frame in ways that work well with framebuffer feedback. Color EQ is more about radically transforming the color space entirely.
BANDS:
Whether in HSB or RGB mode, color eq controls are based in dividing the total color space of a frame up into 5 bands, each of which is interpolated with its surrounding bands. The division of bands is based on dividing the brightness space so that band 1 controls affect the darkest 20% of a frame, band 2 the slightly less darkest 20%, all the way up to band 5 being the brightest 20%.


(hsb mode 1)



HSB mode:
All information regarding Hue and Saturation is discarded. For each of the 5 bands Hue and Saturation is added in manually and the Brightness is offset based on parameter values.



RGB mode:
The Red, Green, and Blue values of each pixel are offset for each band.

RANGES
All parameters seem to be in ranges of -1 to 1. On the internal processing side of things, everything is scaled relatively for each parameter set. You'll have to do a bit of experimentation and refer to this documentation to figure things out properly. FOR EXAMPLE: mix amount p(0,1): means mixing is usually only defined for values between 0,1. ALL(-2,2) means that this parameter gets multiplied by 2 before getting applied. Thus to mix 'normally' would be keeping the parameter between 0 and .5. All other values will result in various kinds of 'distortions.'

Ranges for fb parameters are usually much different from input parameters.
For each set of ranges there will be several sets of numbers. P(x,y) is the total parameter range. ALL(xy) is the range for both IN and FB if the same but different from P. IN(x,y) is range for input, FB(x,y) is range for feedback. ALL, IN, or FB only appear if the parameter range isnt the same as the

RANGES AND FEEDBACK. tiny parameter changes have extreme effects in feedback loops.

MIX AND KEY
mix amount P(0,1). 0 is no mixing at all, 1 is completely mixed. ALL(-2,2). Values below 0 and greater than 1 cause various kinds of distortions. Clamp, Wrap, or Foldover for distortions

key red/key green/key blue P(0,1). The color value for keying is a point defined by (R,G,B) in the RGB color cube.

key threshold P(0,cubeRoot(3)) The farthest distance that any two points in the RGB cube can be from one another is any diagonal line (i.e black to white, red to cyan, blue to yellow, green to magenta).

key soft P(-1,1)

GEOMETRY
GEOMETRY PIXELS are interpolated. Meaning if a x <-> is set to displace 5.5 pixels, an average between pixel 5 and pixel 6 will be used instead of a quantised value of either 5 or 6.

x <-> P(-1280,1280), IN(-640,640), FB(-80,80) in pixels

y <-> P(-720,720), IN(-360,360), FB(-80,80) in pixels

z <-> P(0,2), IN(0,2), FB(.5,1.5) in scaling pixels, 0 is zoomed in completely, 2 is zoomed out so far you can see nothing.

rotate <-> P(-PI,PI) in radians
(PI=180 degrees)

x stretch P(0,2), IN(), FB(.75, 1.25)

y stretch P(0,2), IN(), FB(.75, 1.25)

x shear P(-1,1), IN(-1,1), FB(-.25,.25)

y shear P(-1,1), IN(-1,1), FB(-.25,.25)

kaleidoscope P(0,21). Kaleidoscope is quantized to integer values. This means that only integer values 1,2,3,4 etc will affect the signal.

kaleidoscope slice P(-PI,PI)

COLOR all color values are 0-1. Reference values for RGB: (0,0,0) = BLACK, (1,1,1)= WHITE, (1,0,0)=RED, (0,1,0)=GREEN, (0,0,1)=BLUE, (1,1,0)=YELLOW, (0,1,1)=CYAN, (1,0,1)=MAGENTA. reference values for HSB: (H,S,0) is black no matter what H,S. (0,0,1)=WHITE (H,1,1) is a full saturation pure Hue for every value of H. (H,S,1) for S<1 is a tint (color mixed with white). (H,1,B) for B<1 is a tone or shade (color mixed with some kind of grey or black)
List of rainbow values in H:
0 red
.06 orange
.14 yellow
.32 green
.66 blue
.74 purple
.84 pink
1.0 red

hsb ++ P(-1,1), FB(-.25, .25)
hsb ** P(0,2), FB(.75,1.25);
hsb ^^ P(0,2.0), FB(.9,1.1);
hue shaper P(-1,1), FB(-1,1);
posterize P(1,16). In levels of quantisation. Only integer values 1, 2, 3..affect posterization levels

FILTERS
Filter radius pixels are interpolated. If blur radius is set to 5.5 pixels, an average between pixel 5 and pixel 6 will be used instead of the quantised value of either 5 or 6.
blur radius P(0,10), ALL(0,10) in pixels
blur amt P(-1,1), ALL(-1,1)
sharpen rad P(0.10)
sharpen amt P()
temp f amt P(-2.2)
temp f q P(-1,1)

COLOR EQ
no coefficients here. For HSB mode, Brightness values are retained and parameters offset and Hue and Saturation values are dialed in with the parameters. For RGB mode the parameters offset the existing RGB values.

MATRIX MIXER

basically the same as mix amount, but on the individual RGB pixels instead of combined.