MANUAL
V 1.2

This manual should be fully up to date with GWDSK v1.2

CONTENTS

Section 1 Introduction

What is Gravity Waaaves DSK?

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

- a 2 channel video mixer
- a dual digital video delay processor with delay time up to 4 seconds.
- the flagship VSEDSK (Video Synthesis Ecosphere Desktop) video processing tool
- the only video mixer designed specifically around 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.

Gravity Waaaves DSK is the software only descendent of a shortly lived hardware video mixer. Some strange quirks you might find, both in the software and in this manual, are side effects of the porting process, as I did not start over from scratch for the DSK version of either the software or the manual. If you see things in this manual that seem to directly contradict things that GWDSK gui is showing you, guess what, thats some holdover shit from the old hardware version, hopefully I fix it next time I update

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 hostile 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.

---

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

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.

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.

VSEDSK Community Support

Also please get involved in the various online discussion groups available for working with GW and the VSEDSK! 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 reread the manual or "Ask around in the forums first," so probably just a good habit to get in no matter what.

If you would like to assist in funding any future development or upgrades to GW you should also consider joining my Patreon. Pretty much the entirety of VSE stuffs (including the web design and this manual) is the work of exactly 1 one person (me) and I have no other form of funding or support (or really even like any viable lines of credit) beyond whatever income i get from selling video synths. Which really isn't all that much in the grand scheme of things. Support from Patreon is what even made GW and the VSESDSK possible in the first place. Even if you think GW is perfect and could never be improved upon maybe you'd like to see me actually finish up and release the video sampler thing I've been working on for years now. Sales of these

---

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

instruments are not really up to the point where I'm able to devote much real time to development and I don't do pre-orders until shit is 90 percent finished so options are limited.

Section 2 Getting Started

Registration

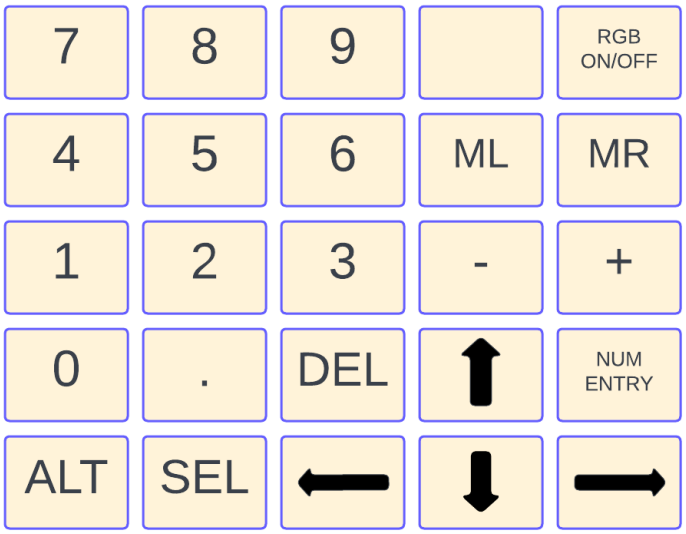
Upon purchasing GW you should have immediately recieved an email containing your password. Certain email redirecting domains and various eurozone domains don't like the service i use for this. If you didn't immediately receive the password, contact VSE right away. You can register one purchase of GW on 3 different sets of hardware, please contact us if you need to use more than 3 copies of GW concurrently.

How to Navigate the GUI

You can operate the GUI using standard mouse/trackpad thingies, your keyboard (or the limited edition keypad), and a midi controller

Mouse or Trackpad

Nothing super complicated here. Only things to note is that if you are going back and forth from using mouse and keyboard for navigating the gui, you'll want to make sure that your last click was somewhere on the gui itself otherwise you won't be able to use the keyboard to move around.



Keypad

The keypad was a special option/holdover from VSEJET days that was offered only for sale to folks who supported the kickstarter. To translate into regular old keyboard keys:

Arrow keys, +, _, 0-9 and . all map to the same keyboard things
SEL maps to space bar
NUM ENTRY maps to enter
DEL maps to backspace
ALT does not map to normal keyboard alt key, only to special VSEJET only commands
RGB on/off affects just the leds on the keypad itself.

There are no options to purchase keypads from the VSE shop any more, folks interested in building their own can visit https://andreijaycreativecoding.com/VSEJET-DIY-INSTRUCTIONS

Arrow keys navigate up, down, left, and right. The *sel/spacebar* 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/spacebar*** 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/spacebar*** 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/enter*** key instead and then use the number keys ***0*** through ***9***, the decimal point ., and ***del/backspace*** to manually change the value and then press ***num entry/enter*** again. The ***rgb on/off*** key turns on and off the color thingies on the keypad itself.

Midi Controller

In an unsurprising holdover from both VSERPI and VSEJET instruments, GW is optimized around a very specific scene setting designed around the korg nanokontrol2. It seemed like a good idea like 6 years ago. Clearly this is midly annoying for general purpose software, and I'm working on a general purpose midi mapping update. Until then, its very simple to purchase a nanokontrol2 and flash the default scene setting which you can find here.
https://andreijaycreativecoding.com/VSEJET-DIY-INSTRUCTIONS

Until a general midi mapping update is available if you'd like to use any general purpose midi controller for GW you'll want to find something with 16 continuous controllers and map them, from left to right like so:
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23|
| 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |

The most relevant place to use the midi controller is on the Macros panel.

Otherwise 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. You should make sure to uncheck the midi/gui box upon leaving a menu to make sure this works. But probably no one will really care anymore about any of this b/c of the macros & presets.

A Brief Intro to the GUI

draw blocks: choose to output either BLOCK_1, BLOCK_2, BLOCK_3 or ALL BLOCKS to the output window.

fullscreen toggle: hit this to fullscreen your output. use the 'f' key on your keyboard to unfullscreen if you went and fullscreened the output on top of the gui.

presets
here is where you can load and save presets. All of the presets in 01_basics are fixed and cannot be saved over. If you've made any of your own cool edits to any of these presets, save em in any userPresets folders. When saving a preset make sure to rename it every time, otherwise something goes wrong. yeah its an annoying glitch, i'm working on it.

queue
When loading a preset you have the option to instead place it in the queue. This is a handy little area where you can get 8 different presets ready to access on the fly. Handy use case example: create a preset that mixes both BLOCK_1 and 2 to delay, set bri** to -.05, mix them together in BLOCK_3 at .25 and name your preset 'fadeToBlack'. Pop this in the queue and you've got a handy fade to black whenever you need to reset the framebuffers in a smooth fashion.

macros
You can choose from any of the 100s of GW continuous parameters to assigne to the macros. This can either be a handy place to consolidate all the most useful controls for a specific preset

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, BLOCK tabs, Save & Load sections, and Midi Macro menu.

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 Sources Are Supported?

Input1 and Input2 will take (just about) any class compliant UVC input. This means
- the vast majority of usb webcams
- a certain selection of usb capture devices
- more often than not, virtual camera softwares.

I wish I could be more specific with extensive lists of exact hardwares & softwares the reality is is that theres too much inconsistency out there in terms of hardware chip sets, windows/osx differences, and even individual operating systems within windows and osx to really provide that much useful information. Things I can say for certain

- logitech usb cameras are usually the best, and anything manufactured in the last 12 years should work fine
- magewell capture devices are pretty rock solid
- roland edirol devices with usb streaming seem to work pretty reliably
- the super cheapo hdmi to usb capture things all over ebay & aliexpress are pretty decent for the price
- the cheapo analog captures (ezcap, av2usb, etc) typically will send a signal, albeit highly compressed and likely with frame dropping
- NDI makes the best frame sharing (i.e screen capture) software bar none

Things that don't work
Obs Virtual Camera
many black magic capture devices.

I'm really very unlikely to support spout or syphon any time soon, and I grimace every time someone bugs me about it.

Media player
Each media player supports
- video files with extensions .mp4, .mkv, and .mov
- image files with extensions .jpg, JPG, jpeg, or png.

in terms of broad codec support, this is system specific. Best practices for video codecs is if something doesn't seem to work, use handbrake (https://handbrake.fr) to convert to h.264
In OSX there is an issue where video files that contain anamorphic pixels (individual pixel ratios are not 1:1) become warped. I recommend using handbrake to convert your video to have square (1:1) pixels.

Quirks and Other Things to Look Out For

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.

Long video cables used for analog or digital captures 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 shielding and quality of the cable you can sometimes run into issues over 15 feet.

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 whatever processing width you've chosen. Say you're working in 1280 x 720 then a visual value of 1 would be an x displacement of 1280 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.

Choppy Input Video/Stuttering Video Delay

If you've set input or processing resolutions to be higher than your hardware can handle you'll see frames dropped in your video source and/or stuttery video in either delay line. Try bringing down input (or media) resolution first. Try turning off any and all other programs using your gpu/vram. Make sure that your computer has automatic updating turned off (if possible).

Displace & zoom effects work differently at different input/processing/output resolutions

I'm working on it

media/inputs i'm selecting don't match up with whatever actually gets loaded or changed

Make sure to rescan video inputs and/or media folder and check back, it is most likely that an input got dis/reconnected and/or you've added or removed files from one of the media folders with out rescanning.

my input or media looks like its getting cut off

GW automatically scales any input or media to be centered and fill the processing frame. You can still use the x, y, z <-> and see that no information has been actually lost, if you prefer pillar/letterboxing your formats that is totally possible.

saving/loading presets is sometimes a little funky

When saving a preset make sure to rename it every time, otherwise something goes wrong. yeah its an annoying glitch, i'm working on it.

merge presets and mutate parameters is a little funky

yes they are.

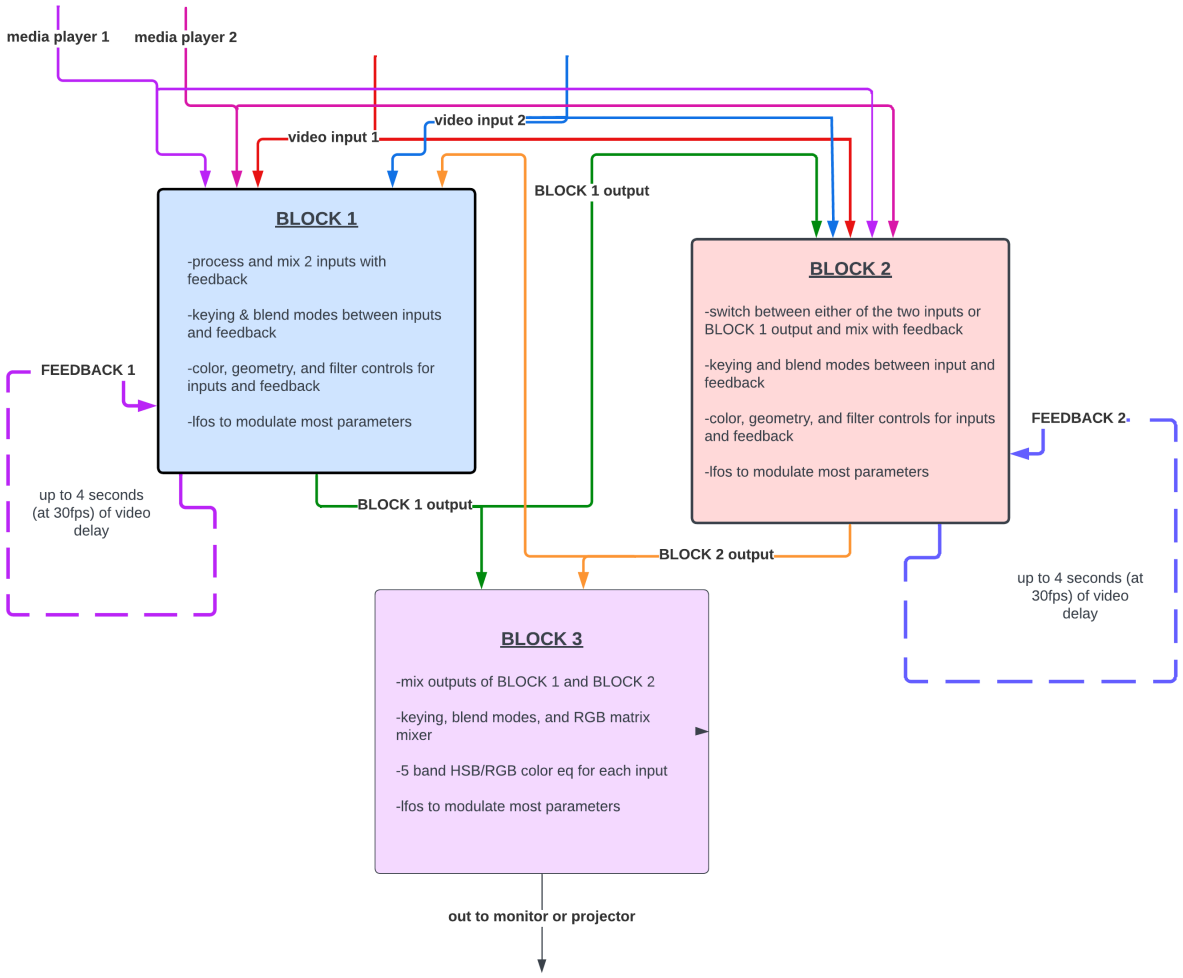
obs virtual camera doesn't work

yeah i dunno how many times and in how many places i need to mention this: obs virtual camera is not supported. figure out something else. either way you should probably prioritize using obs to screen record your output no?

Signal Flow and BLOCK Structure

We have sources (media/inputs/internal framebuffers), processing, and output that can all work at different resolutions. For example, you could have 1 usb camera that is set to 1920x1080, a video file at 640x480, processing resolution set to 1280x720, and your final output set to 4k. The way this works internally is that your webcam will get downsampled to 1280x720, the video file will get upscaled to 1280x720, all the feedback, fx, and other processing will happen at 1280x720, and then your output video window will be at 4k. There is not any kind of particularly fantastic upscaling or downscaling algorithms baked into GW, tho the ones for bringing inputs and medias into BLOCK_1/BLOCK_2 are a little bit smoother than the output scalers. Best practices is to work

mainly with inputs, files, processing, and output resolutions that are pretty close to one another. Unless you are specifically looking for wonky artifacting somewhere in the chain, which can be it's own source of entertainment.
The 3 most likely resolutions you'll be working with would be 640x480, 1280x720, and 1920x1080. A good rule of thumb is that you can go up or down one step in that chain without things looking too gross. Jump two steps and you'll probably see something a little unseemly.



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

BLOCK_1

Process and mix 2 inputs (either live uvc inputs, media players, or BLOCK_2 output) with 1 Video Delay Line.

BLOCK_2

Process and mix 1 uvc or media player 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 fiiltered 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.

Settings Page

The settings page is where you can tweak details that live outside of signal flow and performance. These are all things you shouldn't need to access during a live performance.

rescan video inputs

Updates the list of sources you can assign to input1 or input2. Use this if you have plugged in/installed a new uvc video source while GW is active. Also helpful if a uvc source has gotten disconnected while GW is running. Sometimes when you rescan video inputs, the specific device id for inputs will change. This won't propagate to any actively running inputs until you hit change input 1/input 2.

rescan media folders

Updates the list of sources you can assign to media1 or media2. Use this if you have added any new files/deleted old files from your data/videos or data/images folders to ensure that the menus

in 'select media1/media2' reflect the actual contents of these folders.

sync medias
restarts both media1 and media 2 at the same time. Still kinda wonky and doesn't work perfect every time.

media players
select media1/media2
Displays a list of all of the identifiable files in the data/videos and data/images folders. Only files with extensions .mp4, mkv, mov, jpg, and png will be identified. You can use irfanview or xnconvert to bulk convert image files and handbrake to bulk convert video files to these acceptable extensions if necessary.

load media1/2
Loads whatever media file you selected with the select media1/2 menu. Note that simply selecting a file using the previous menu will only queue it up and nothing will change until you hit this button.

mute/vol cntrl
All video media files are muted by default. If you'd like to pass through the audio, uncheck the mute box and you'll see a vol cntrl slider pop up. Note that there is no built in audio processing so it is quite possible that if you have unmuted and cranked up media1 and media2 you'll hear some clipping or other undesirable audio distortions. It is completely up to you to avoid these by judicial use of the vol cntrl slider.

video inputs
select input1/2
Choose from any available uvc (hardware usb or virtual camera software) inputs. Note that when using virtual camera softwares, you may also need to configure settings within their respective interfaces as well. Below this you'll see another drop down where you can select the resolution. Note that just because a resolution is in this list, doesn't actually mean that your uvc source will actually support this resolution. It just means that after you hit 'change input 1/2' that GW will try to grab whatever the closest actual available resolution the uvc source supports. For example, if you try to select custom resolution 1281 x 715, it is incredibly unlikely that any source will support that, but if 1280x720 is supported, GW will just grab that instead.

change input1/2
Actually changes the input to whatever source and the closest available resolution you've selected.

select midi input
Choose an available usb or virtual midi controller. Hit the reset midi devices if you've added or removed any midi devices while GW is running.

internal processing resolution selection
Allows you to select what resolution that all of the fx, processing, and video delays run at.

gui text scaling
choose the size of text on gui. good rule of thumb would be 100% for less than 1080 monitors, 150 for 1080, and 200 for 4k, tho retina displays and other levels of internal app scaling can affect readability as well. Note that this only affects the size of text itself, the gui itself will change relative positions depending on the actual size of the window

set gui//app size and position
Here you can manually set the gui/app window size and position. This will be very helpful for creating init settings for specific use cases. Stuff to note: if you have multiple monitors, the x position is calculated along the assumption that you just have one extraordinarily long monitor. For example: if you are working with 2 1920 x 1080 monitors and you would like the gui to fill the left monitor and the output to fill the right monitor, you'd set gui x position to 0 and app x position to 1920.

save/load initial settings
An initial settings file contains
- input resolutions
- currently selected midi device
- internal processing resolution
- gui text scaling
- gui/app window size and resolutions (as specificed in the above menu)
- preset queue

These are handy to use for if you have specific settings you'd like to quickly pull up for screen recording, projector based installations, working with different numbers of external monitors on your laptop, working wtih different resolutions etc. You can have 8 different init settings and 1 default init that loads whenever you start GW.

merge presets
dunno if this really quite works as promised yet.

mutate parameters
This is supposed to do exactly what it says, tho i don't know i it really works all that hot yet.

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 0: Presets and Macros

Midi Macros

Midi Macros are a way that you can choose a set of any 16 parameters to have permanently mapped to the midi controller. Macros are saved along with all the other information in a preset so lets first explore one of the preset macros.

Open up the ***midi macro*** menu and then go up to the load, select *filterFreak*, and hit *load*. You should see the ***midi macro*** menu is now populated. Try adjusting each of the macros with the midi controller and you'll see that the gui sliders move along with the physical controls. Yes there are knobs on the midi controller but all of the icons are sliders. This will be ok. You'll notice the *macros on* checkbox will turn on automatically when you load presets too.

The *reset* switch in the ***midi macro*** menu resets all of the parameters to 0. The *reset assignments* switch resets all the assigned macros to null.

Close the ***midi macro*** menu for now and hit *reset all* up on the top right corner. Then load the same preset once more. You'll notice that all the parameters immediately change except for *fb1 delay time*. If the midi macro menu is closed then all parameters will change upon loading except for *fb1* and *fb2 delay time*. I'll figure it out eventually but thats just how things are right now.

Open up the midi macros menu once more and you'll notice that the *fb1 delay time* jumps back up to where it was in the original preset. Lets put all of that behind us now and try editing one of the macros.

Head over to the top right macro where *fb1 posterize* is currently assigned. Lets change this to *fb1 z <->*. Go down to the select button underneath the slider and navigate through the menus. You'll notice arrows that imply you should press right on the keypad but then the sub menu shows up to the left. Keep pressing right and more menus show up to the left. Weird I know! Go through and select *fb1 z <->*, then navigate back and hit ***dunzo*** to exit the menu. You'll notice that fb1 z is now zoomed out a bit! Parameters jump to wherever the macro slider/ midi control is at when you choose a new one.

Another weird thing with the macro select menu: you can't navigate back to the b1 b2 b3 part of the menu. Just hit dunzo button wherever you need to and go back if you selected the wrong b.

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.

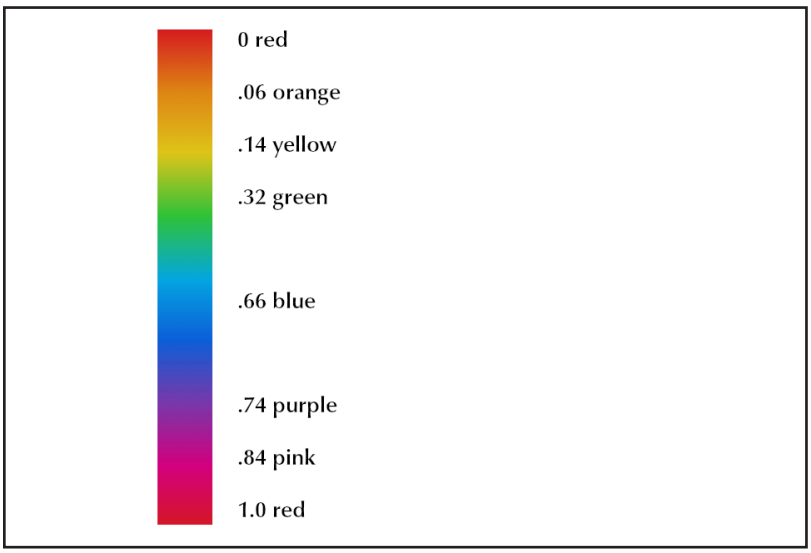
Once you've played around with that for a bit, go up to the drop down in the left handcorner labeled *matrix mix type* and select *additive*. go back down and try some different settings. Repeat with each of the different matrix mix types.

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 worked.

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.

INTERMEDIATE

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 & Color Eq

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 bring-

ing *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

Putting it all together

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**, bypass any inputs, and create another feedback oscillator.

- *fb1 delay time* to 12
- ***fb1 mix and key->mix*** to .5
- ***fb1 geo -> z*** <-> to .1
- ***fb1 color -> bri*** & ***sat invert*** switched on
- *bri* ** to 1.0
- *hue* ++ to .4
- ***fb1 filters -> blur amt*** to 1, ***blur rad*** to .1,
- *temp 1 amt* to .08, *temp 1 q* to .16,
- *temp 2 amt* .1, *temp 2 q* .06

- ***fb1 lfo -> geo lfo 1 -> z*** <-> ***a*** to -.08, ***z*** <-> ***r*** to .16
- ***geo lfo 2 -> x stretch a*** to 1, ***x stretch r*** to .1
- *y stretch a* to 1, *y stretch r* to .04
- *x shear a* to .4, *x shear r* to .67
- *y shear a* to .4, *y shear r* to .085

This should give you an nice and dynamic feedback oscillator in **BLOCK_1**.

B R E A K

Now select *draw BLOCK_2* for the moment to do some exploration with GW and camera feedback. Make sure your camera is pointed fairly squarely at the screen to start. You can use **BLOCK_2->*BLOCK_2 input adjust->rgb invert*** to match up the camera with the screen. After that, staying within ***BLOCK_2 input adjust*** try out:

- all of the *hue*/*sat*/*bri* ^^ postive and negative amounts
- *posterize* postive amounts
- testing out differences between *hue*/*sat*/*bri inverts, rgb inverts*, and *solarize* with camera feedback
- *blur* and *sharp* filters, first on their own and then together
- *kaleido* positive amounts, *kaledio slice* positive and negative
- with *kaleido* active also make sure to try different *overflow* modes and *z* <->

- and with the above all active and *z* <-> postive try out *rotate* <->

- *h/v mirror* and *h/v flips*

- and then head into ***BLOCK_2 input adjust lfo*** and try modulating geometries stuffs!

B R E A K

Select *draw all BLOCKS* for a moment, head on over to **BLOCK_3** and select *block-2->block1* as *final mix order.* Set ***final mix and key -> key threshold*** to somewhere around .4-.6 and *key soft* around .1-.2. Try different ranges on both to see how they work with your personal camera and screen setup. If you are having issues with really crunchy keying, try a bit of blurring and sharpening back in **BLOCK_2->*BLOCK_2 input adjust***.

Now select *draw BLOCK_3* and lets tweak the color eq on the camera feedback. Head over to **B_2 parameters - > *B_2 color eq*** on, *hsb/rgb* set to rgb (checked on)

- play around with mainly band 4 and 5. Because my camera was getting kind of caught up in muddled bluish tones, I set things the following way

- band 5 red to .12, green to .1, blue to -.36

- band 4 red to -.3, green to -.38, blue to .06

- band 3 red to .3, green to, .26, blue to .02

but depending on how your camera sensor works you might want to try something totally different!

back into BLOCK_3

final mix and key reset.

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

-try out mix types difference and multiplicative, values between -.5 and .5, with overflow clamp and then with overflow foldover

-different mix orders with the same mix and key settings

-color eq hsb on camera feedback

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. A Time Base Corrector ensures that the parts of an analog signal (horizontal and vertical sync pulses) are looking good

HDMI : HDMI is technically a protocol for a video cable terminator and not exactly a signal itself. Most folks just call the signals and cables by their terminators tho.

OVERSCAN: both CRTs and LCDtvs very rarely show the entirety of a video signal. Capture devices will however show the whole thing. This means sometimes the image you see from an analog capture device looks larger than what you see on the CRT or LCDTV.

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 directly manipulating video signals in real time 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's default 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 digital video signals are in some kind of 16:9 format.

ANAMORPHIC PIXELS

Most commonly seen nowadays in camcorders. A video file with anamorphic pixels is storing a visually compressed frame (usually squeezed in the horizontal axis) along with some kind of a flag in its metadata that tells whatever program plays the video back how to stretch it back to get the right size of image. This is not the quite the same as the non square pixel ratios found in NSTC or PAL video signals because these are literally displayed on CRTs in non square pixels.

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's default processing 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.

PIXEL OPERATIONS
color, geometry

FRAME OPERATIONS
filters

MULTIPLE FRAME OPERATIONS
mixing, keying, delaying

Reference

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)

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.

clamp: everything outside of (0,1) gets squeezed back into 0 if negative or 1 if >1.

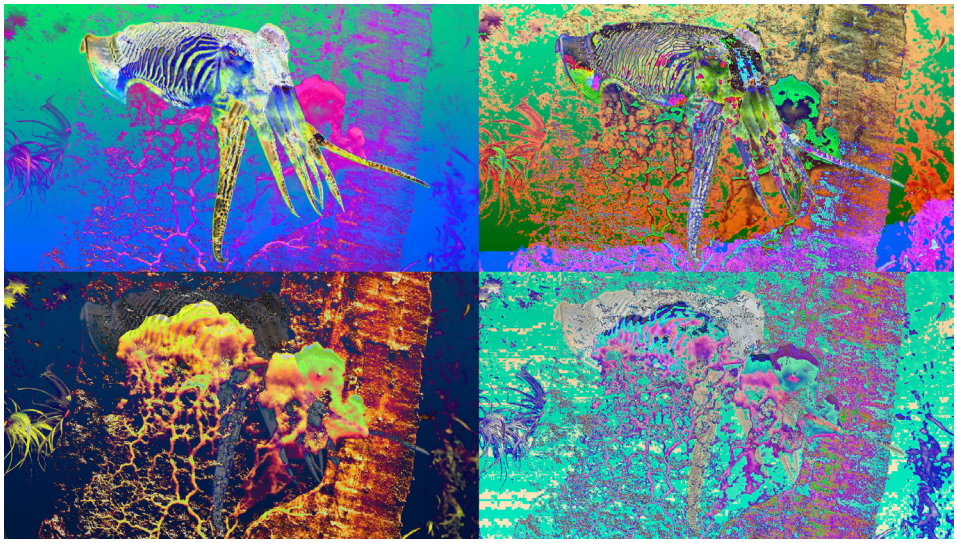
wrap: values that go below 0 wrap around to come down from 1. EG -.2 ->.8. values that go over 1

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

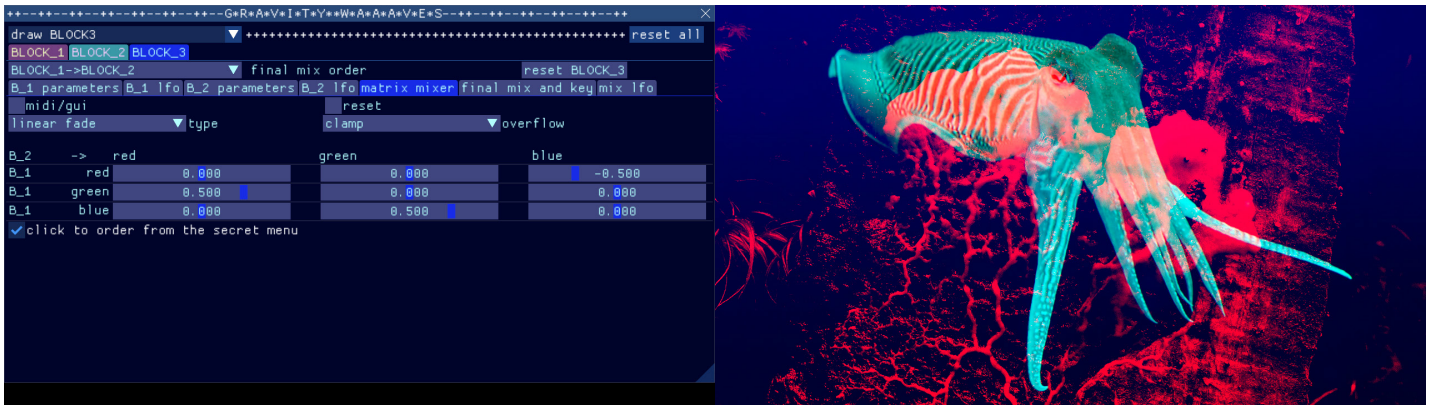
key type: keying is choosing between two video sources using some kind of logical operation based on the value of each pixel in A. The chromakey uses any (R,G,B) value while the lumakey

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.

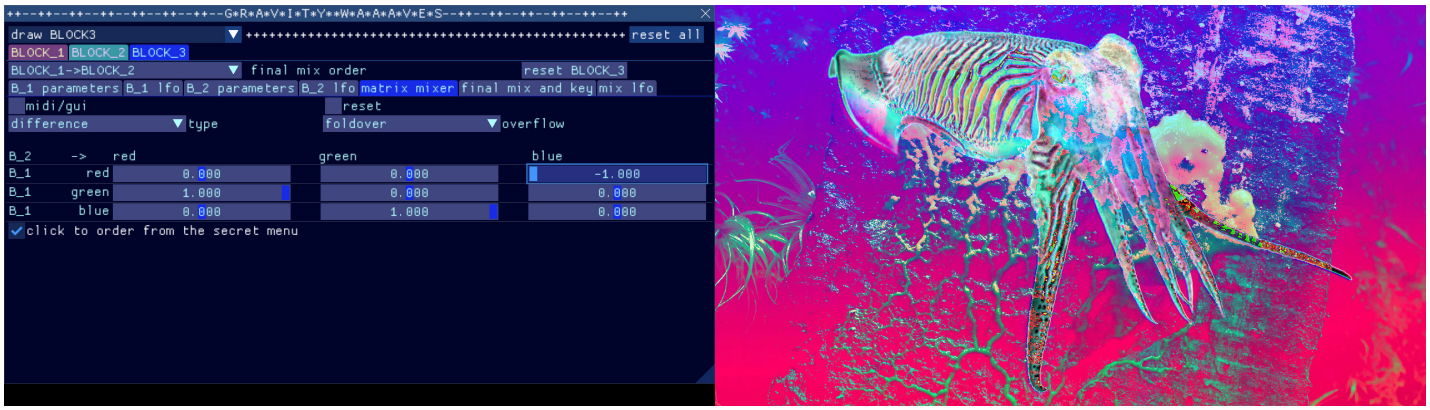
key value color: changes to show you what color has been selected by adjusting *key red, key green,* and *key blue*.

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.



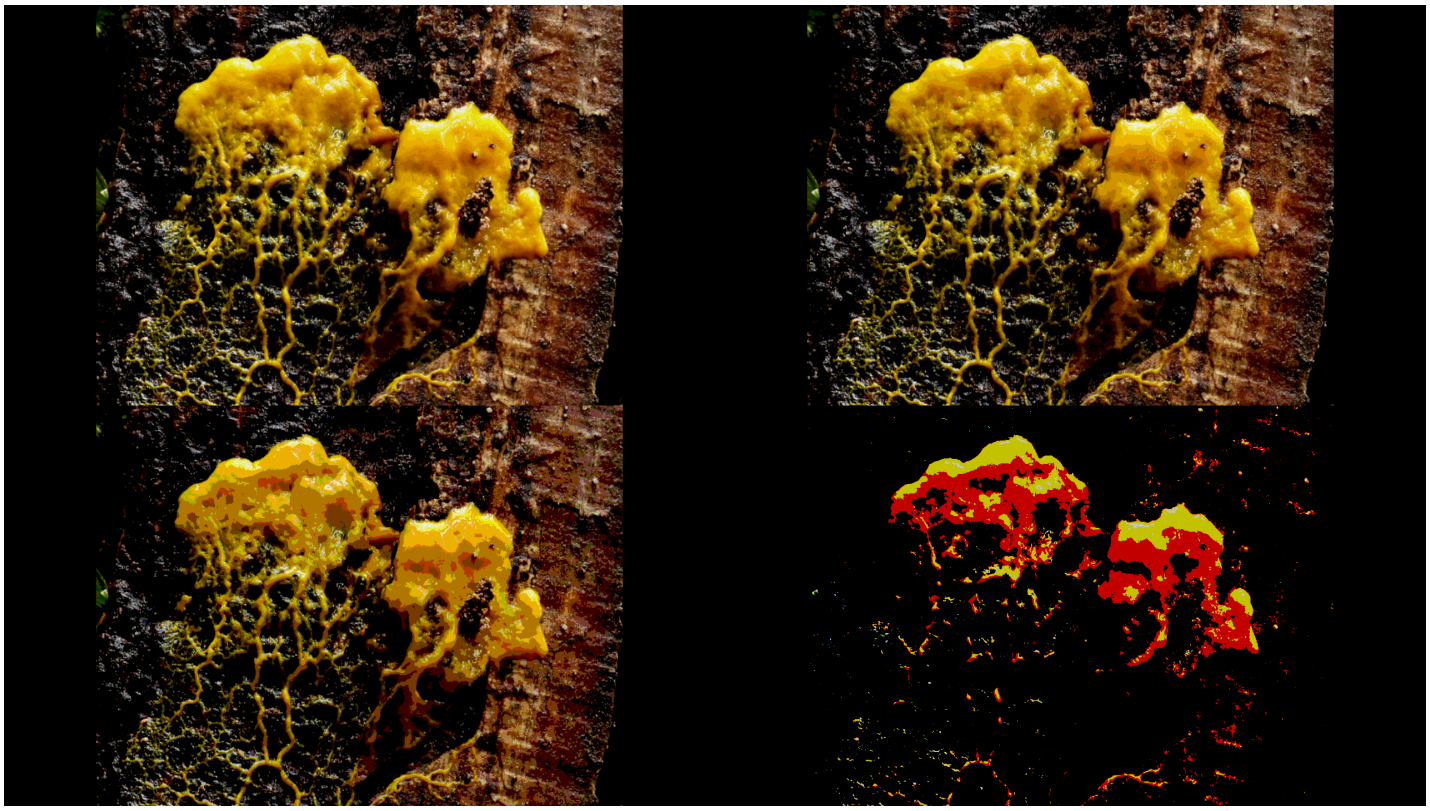
MATRIX MIXER:
Every pixel in each frame can be described as a set of Red, Green, and Blue components. What the matrix mixer does is allow you to directly control how much each of the individual Red, Green, and Blue components of every pixel in each frame are combined.



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.

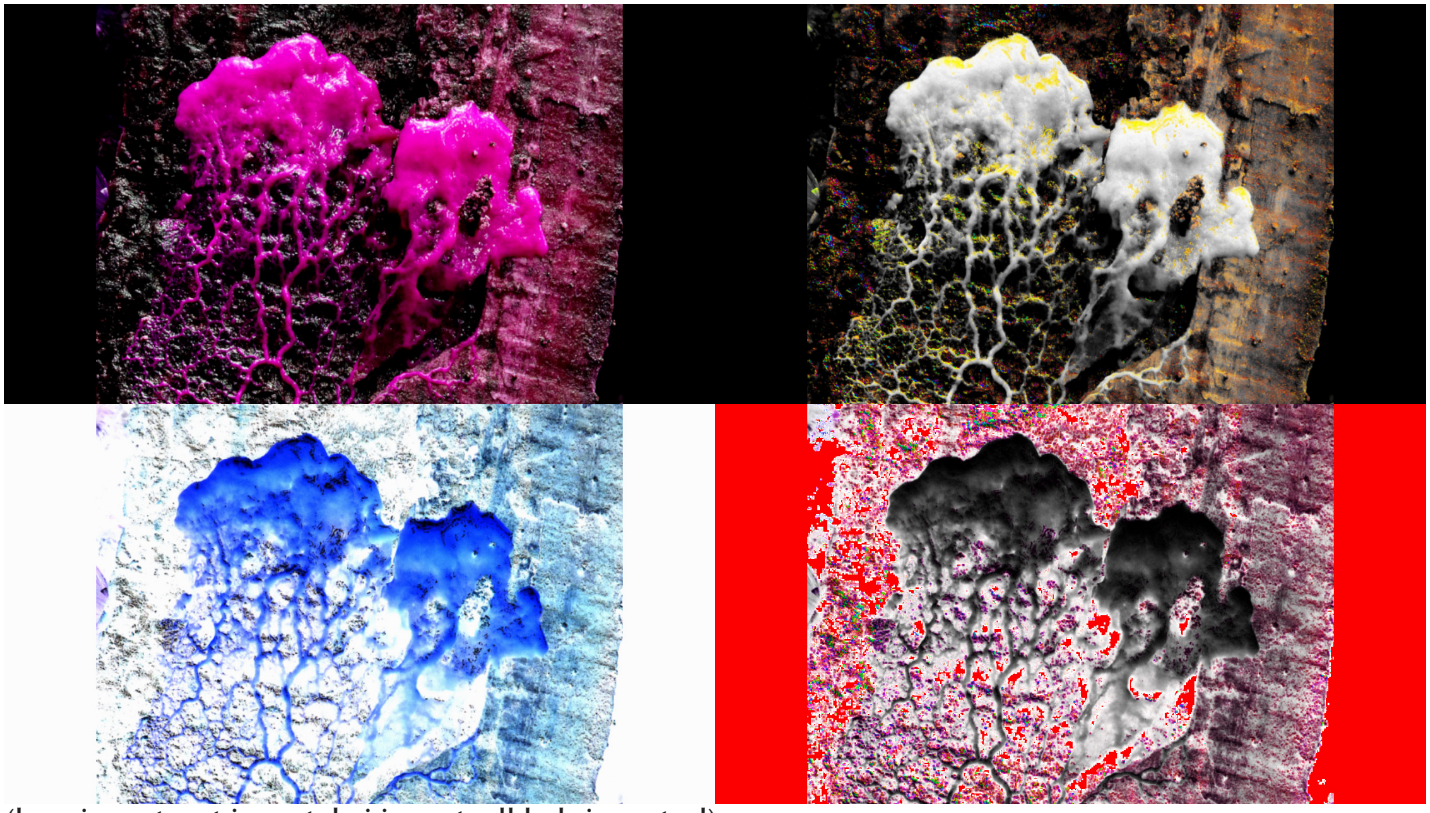
hsb ++: offset or pedestal. The math for calculating hue would be f(P)= H+P. Values are simply shifted up or down
hsb **: attenuation. The math for calculating is f(P)=H*(1+P). The entire range of values are scaled up or down
hsb ^^: power mapping. The math for calculating is f(P)=H^(1+P). Values are 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.



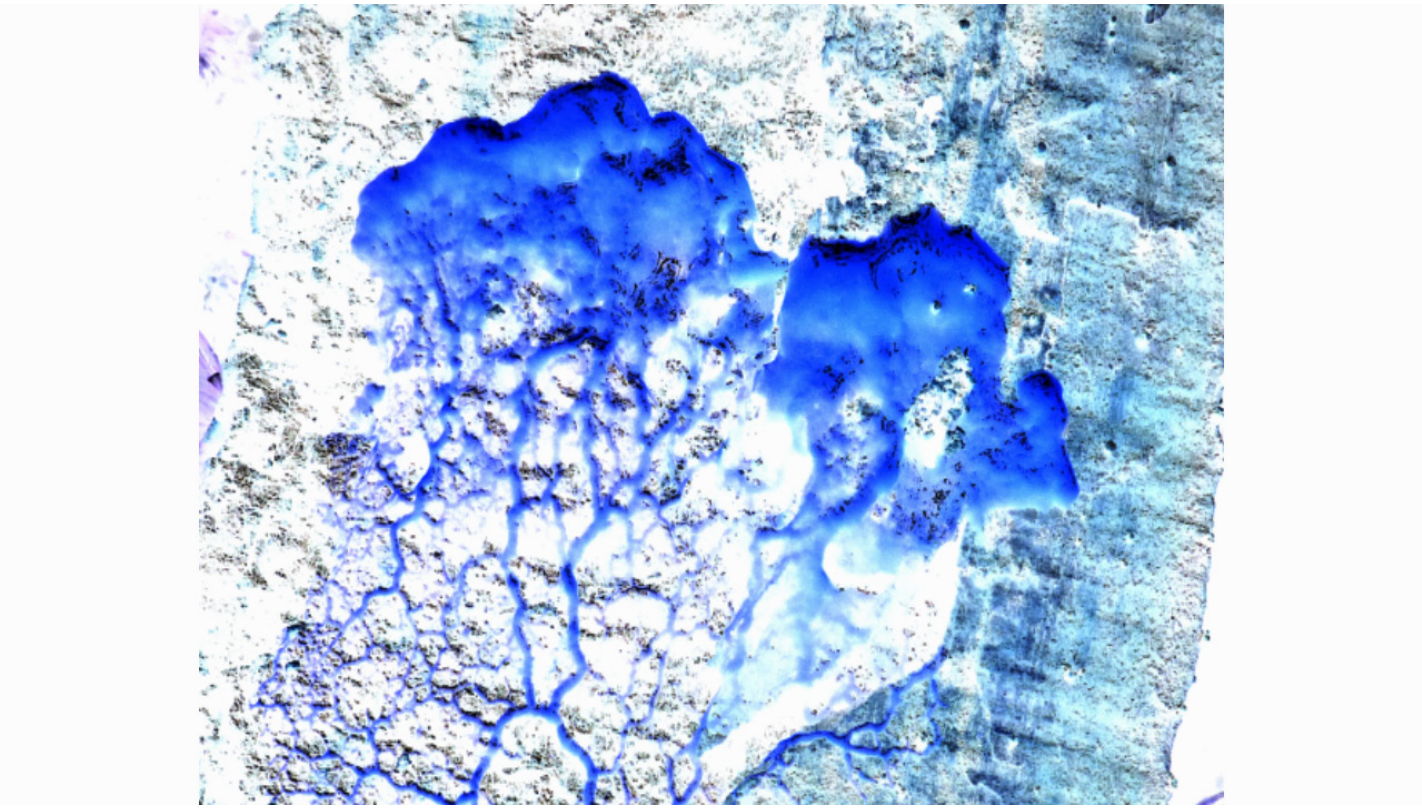
posterize the slime (.2, .5, .78, .98)

posterize: quantize the colors in HSB 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.

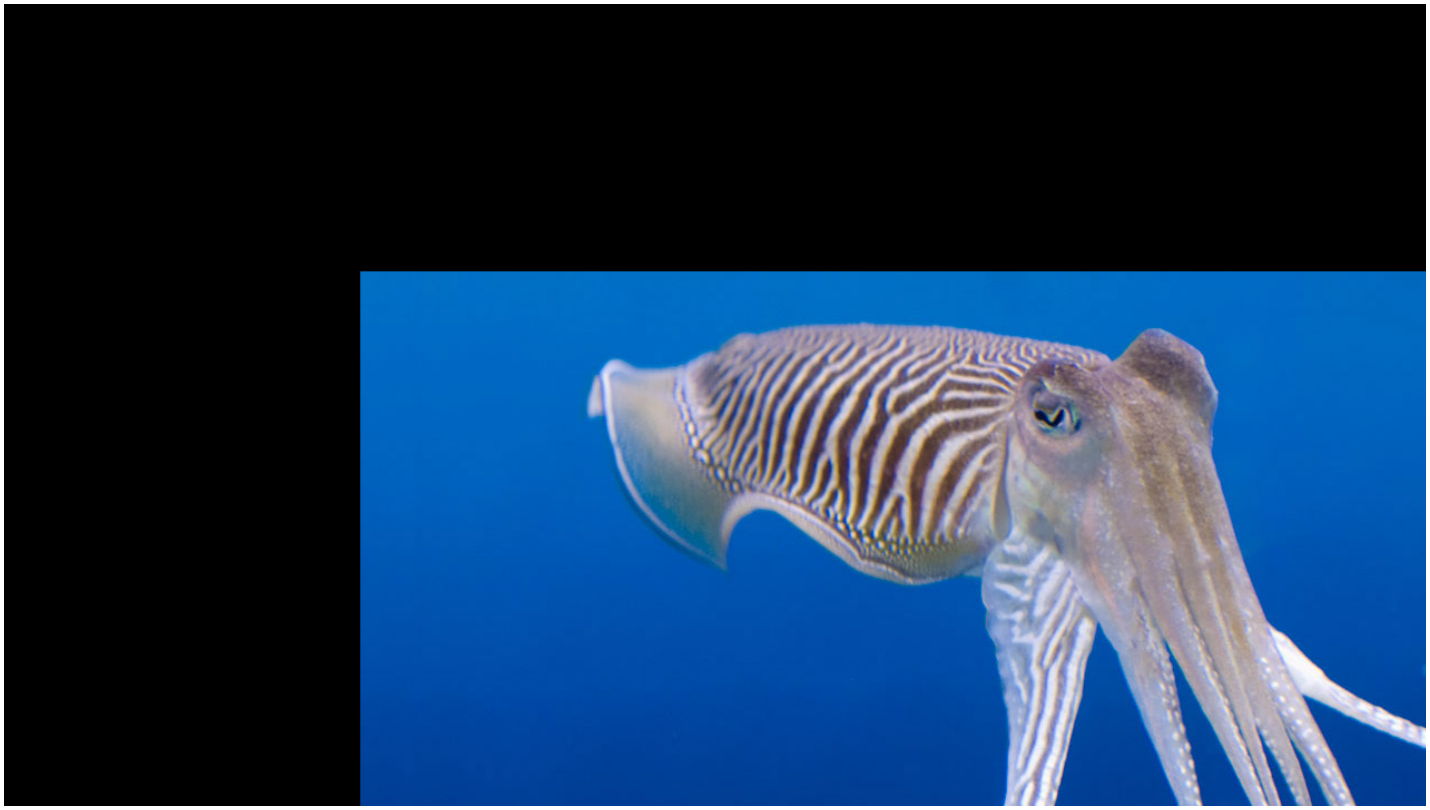
solarize: the darkest 50 percent of the frame is left alone and the brightest 50 percent is inverted.

ORDER OF COLOR OPERATIONS

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

GEOMETRY

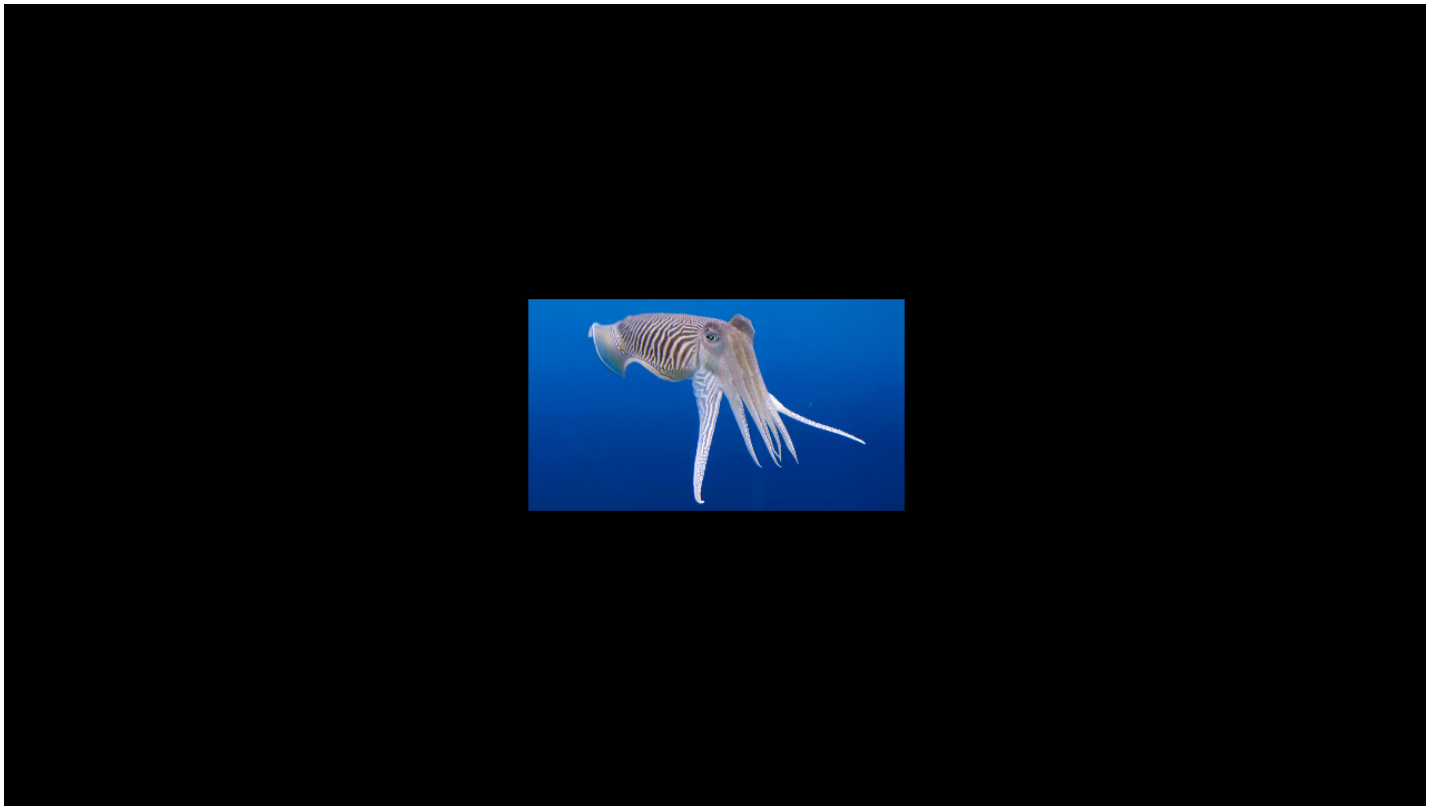
We will use this image of the cuttlefish for examples of how geometry operations work



(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

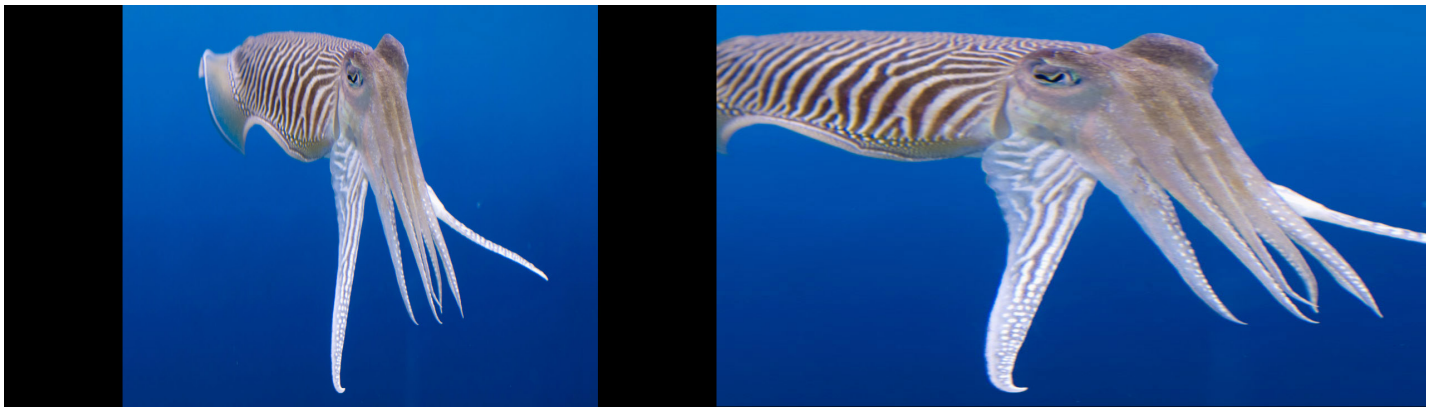
y <->: y displace. moves the entire frame up or down. changing the y position affects the vanishing point of z displacement, the center of rotation, and the center point of y stretch



(z at .240)

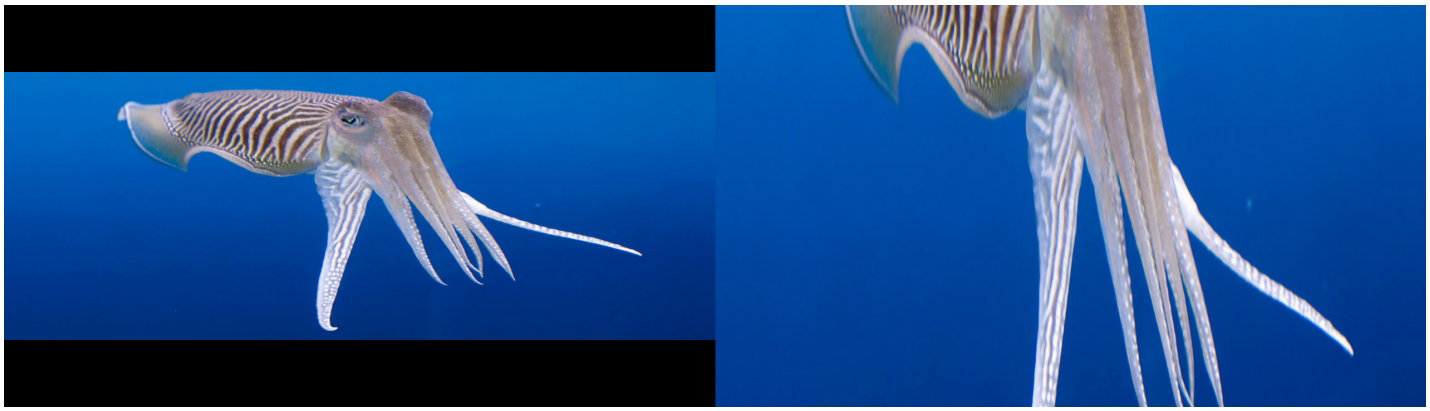
z <->: z displace. shrinks or blows up the frame.

rotate: rotates the entire frame around wherever x and y displace are at. Default rotate mode preserves the aspect ratio of the image while rotating.



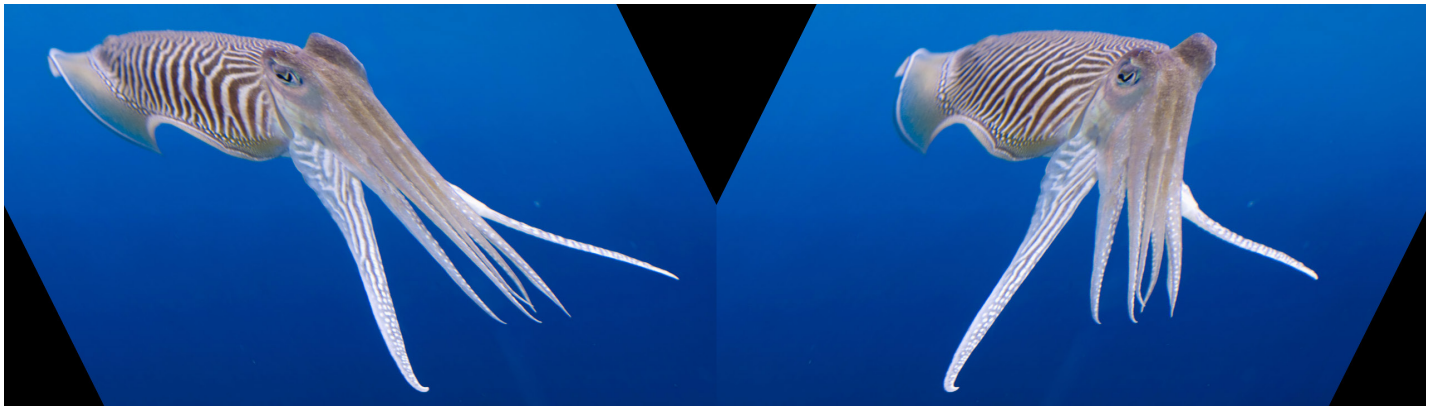
(x stretch at .5 and -.5)

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



(.5, -.5)

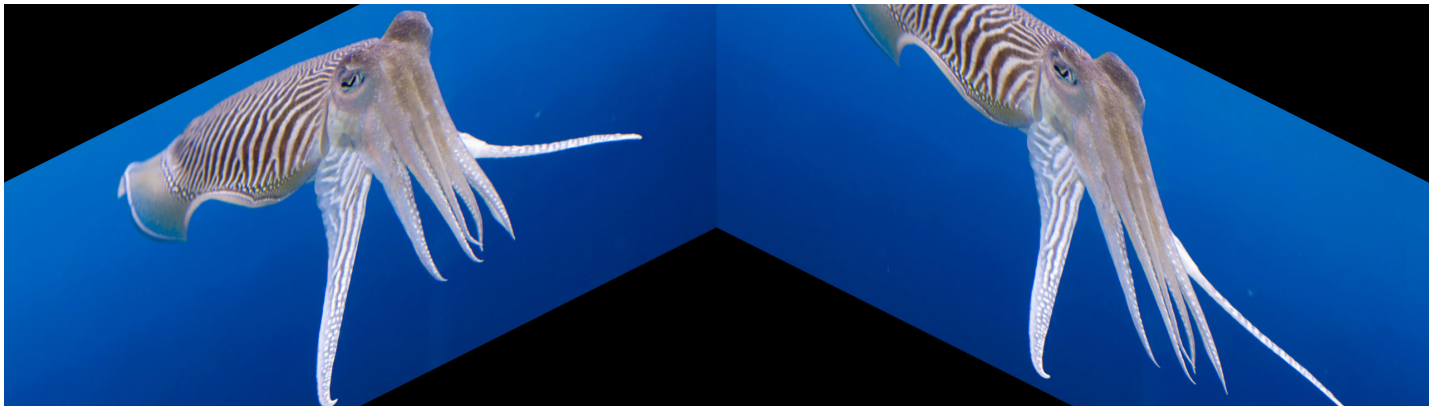
y stretch: positive values stretch and negative values squeeze the framebuffer along the y axis.



(.5, -.5)

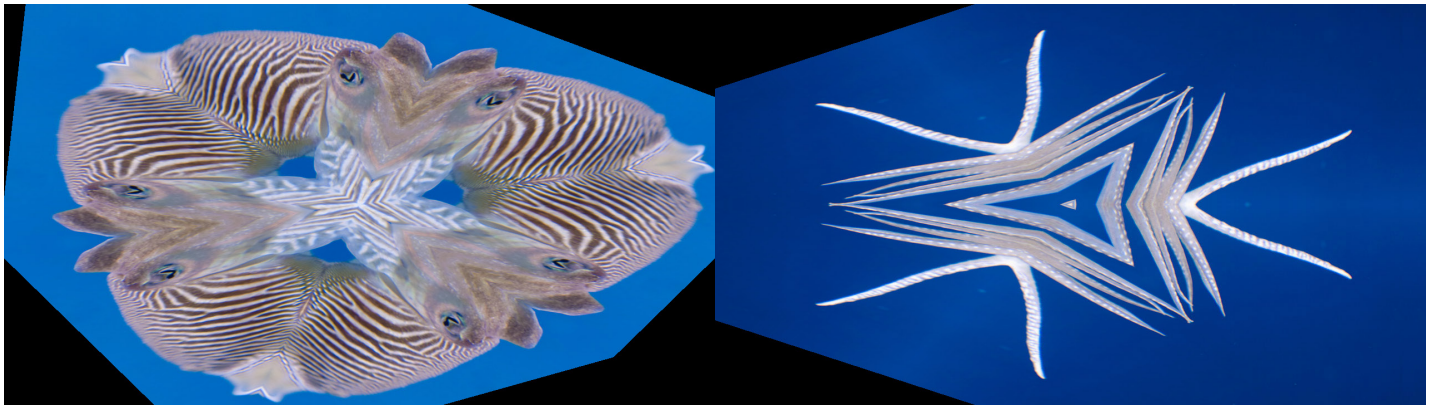
x shear: the y position of pixels remains unchanged but x value gets displaced at a relative

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



(.5, -.5)

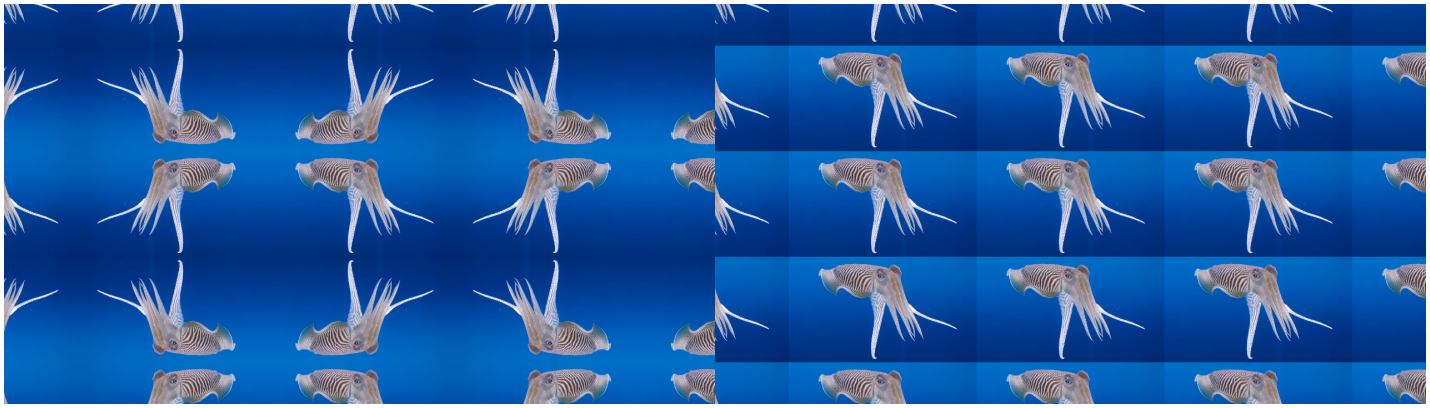
y shear: the x position of pixels remains unchanged but y value gets displaced at a relative amount to its x position. postive displaces up, negative down.



(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 a frame are shifted 'off of the screen'

clamp: video is blanked out

wrap: also known as toroidal universe in older VSE thingies, 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. Pac Man used to do this all the time.

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

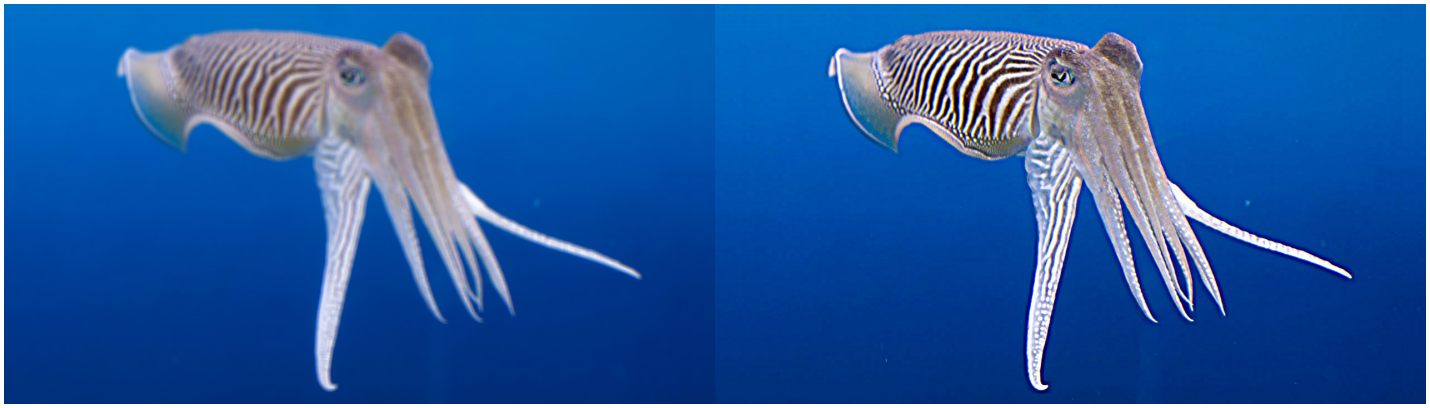
v mirror: everything on the bottom half of the screen is replaced with a reflection from the top half

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.

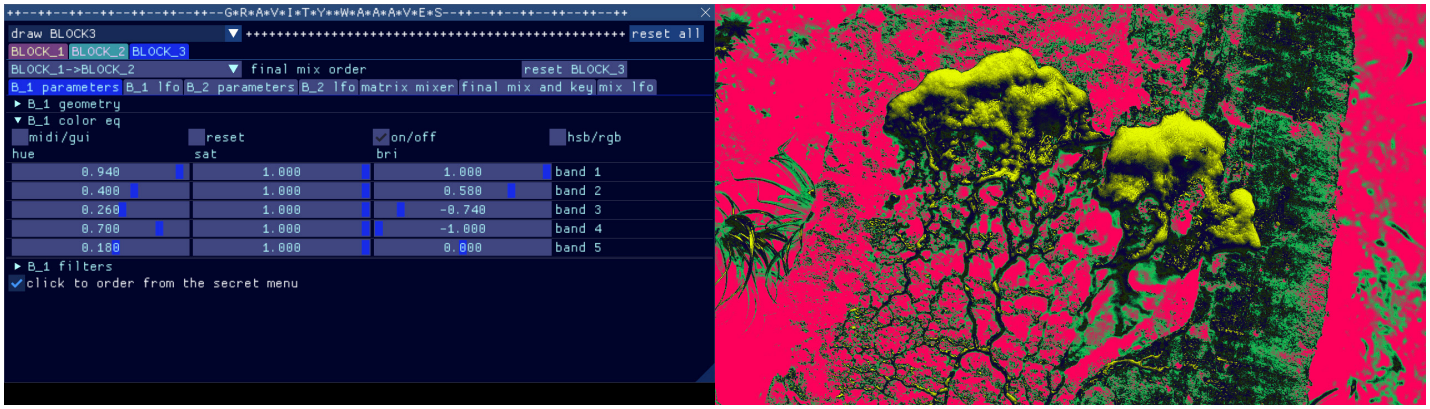
COLOR EQ

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.

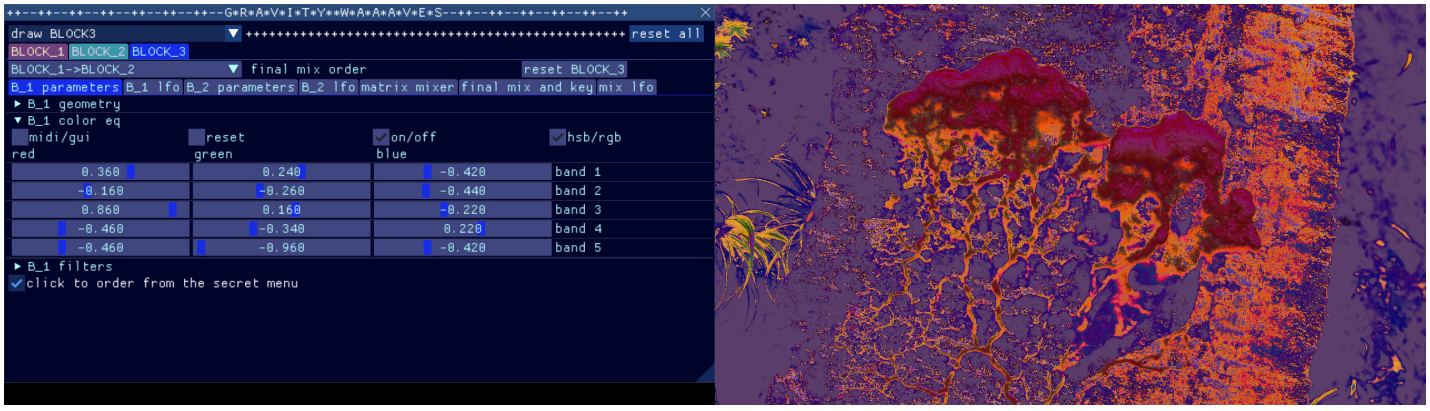
BANDS:

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)



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 prexisting 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.

Geometrical distortions work fundamentally different for inputs and delays. If you displace a media player input by 1 pixel, all you see is the same input but displaced 1 pixel. If you displace a delay line by one pixel, because of the recursive feedback happening, you see a continuous movement in the delay. What is happening, at a rate too fast for your visual perception, is that the video delay frame gets shifted by 1 pixel, then on the next frame the delay is shifted 1 more

pixel over, and so on and so on so that, assuming we are running at 30fps, in 1 second the frame has been shifted 30 pixels. In feedback situations t makes more sense to think of these geometry parameters of controlling velocity of the apparent motion, i.e both the rate and direction of movement.

x <->
for any inputs (uvc, media player, BLOCK_X) x displace is the full range of whatever resolution the input is set to. I.e if your video clip is 640x480, x displace set to 1 is 640 pixels. If your uvc input is 1920x1080, a value of 1 maps to 1920 pixels. If internal processing is set to 1280x720, then BLOCK_1 input for BLOCK_3 x displace is 1280 pixels.

for video delay lines, x displace is .0625*processing width.

y <->
for any inputs (uvc, media player, BLOCK_X) y displace is the full range of whatever resolution the input is set to. I.e if your video clip is 640x480, y displace set to 1 is 480 pixels. If your uvc input is 1920x1080, a value of 1 maps to 1080 pixels. If internal processing is set to 1280x720, then BLOCK_1 input for BLOCK_3 y displace is 720 pixels.

for video delay lines, y displace is .0625*processing height.

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.

Section 5 Erratum (for DSK)

i hope that most all if this stuff has or will very soon be merged into the main manual. Still keeping this here until im like 101% positive that this is the case.

- number one thing to keep in mind is that this is, first and foremost, a Video Mixer built around the UVC standard. If you have no video devices (either hardware usb or virtualcam softwares) working on your system, you shouldn't expect to see anything happen right away. if you have a lot of video devices working you'll want to use the settings page to choose which ones in particular you want to run. you should have at least 2 hardware or virtualcam video inputs working and ready to use before starting GWDSK

- otherwise if you insist on not using video inputs for some reason, try searching through the presets and you'll find stuff pretty quickly that works fine with zero inputs.

- there are really kind of a lot of free 'virtualcam' softwares out there that you can use to get video from your desktop into . try using a search engine that isn't google to find one that works best for you. OBS virtual cam actually isn't really the best one unless you really, seriously, know what you are doing.

- mouse pointer: the mouse pointer dissapears when its over the output window. this is normal and expected behavior

PASSWORDS AND REGISTRATION

- you must enter your
    - username (email address with which you bought GWDSK)
    - the machine you are registering on (i.e windows11AcerLaptop, osx14macbookPro2021, enough information to identify this specific version you are registering)
    - and your password to verify your version of gravity waaaves
- no whitespaces anywhere
- you can only run your purchase of GWDSK on up to 3 different machines. if you absolutely need to run things all the time on a larger number of computers for some reason, contact andrei directly via email.
- if you are updating GWDSK and want to keep your registration and save states, simply install to a different folder, and copy the reg.json file and your entire presets folder. i recommend backing up any presets you are saving for now somewhere separate too b/c each new update will come with more presets

SETTINGS PAGE:
- most of these things are fairly self explanatory and will result in obvious visible changes if you mess around with them. The most important thing is that you can Save the majority of things you tweak on this page using init settings. you do have to manually save them yourself, and choose whether or not any particular one is default. nothing anywhere in GW gets auto-saved.

INPUTS
you cannot assign the same UVC source to both input1 and input2. you will be prevented from changing inputs or saving init settings if you've selected the same source for both input1 and input 2.
- you should be able to change the PROCESSING resolution on the fly. note that this won't change anything about the output resolution, that is controlled by the window size. it will make a difference tho if your processing resolution and the output window are different aspect ratios, either stuff will get cropped width wise or there'll be pillarboxing.
- if input video and processing resolutions aren't the same aspect ratio then the default is to center input video and fill the processing buffers with video. you will notice that if you move the input video using 'x <->' controls in the relevant adjust menu that nothing is actually cropped, and if you prefer letterboxing to cropping thats just a matter of tweaking 'z <->'

OUTPUT VIDEO
- drag and resize the screen, the output will resize to automatically maximize height. it should only pillarbox, never letterbox. stuff will get cropped width wise but never height wise
- you can also manually change the position and size of output screen (and gui) from the settings page and save these in init files. multiple displays are treated as one continuous virtual display here, so if you've got a laptop and an external monitor, both with 1920x1080 resolutions, and you want to put the output video on the
-fullscreening: make sure the output window is on the correct screen you want it to full screen on. you can either press the fullscreenToggle button on the gui or click anywhere on the output window and press the 'f' key. If you went and fullscreened your output window over the gui (try not to do this), click back on it (potentially difficult b/c you can't see yr mouse pointer on the output window) and press the 'f' key again.

PRESETS
- rename files when you save them every time for now, unless you are just saving the exact same preset in the exact same slot with the exact same name. just do it! in fact you are forced to by default and cannot turn it off.
- you cannot save over any of the presets in 01-basics
- the preset queue is set up so you can have a bunch of presets ready to roll at any point. say if you wanted to have a 'fade to black' preset made you don't need to search thru everything to load it up right away. the main thing to keep in mind is that the preset queue points to specific slots in directories, so if you a working on a preset thats currently in the queue and save it in a different slot, the queue won't point to the new slot you saved in.

KNOWN THINGS
- changing anything about inputs takes like 2 seconds to complete and involves said input freezing, then blanking. this is why the inputs only will actively change if you hit the button, not just from any random messing around with resolutions or sources.
- changing processing resolution takes about 1 second or less but is noticeable
- saving/loading input2 in initFiles doesn't work at the moment
- if you move or resize either window, the output video freezes.
- macro controls: either you can control them via nk2 set to the usual VSERPI/VSEJET scene setting (visit website for instructions on that) with the 'macros on' checkbox on or via mouse/keypad/keyboard with 'macros on' checkbox off.
- gui text size loads really small for the first couple seconds then gets larger.

Please do feel free to report bugs using the support forms here
https://videosynthesisecosphere.com/support
tho!