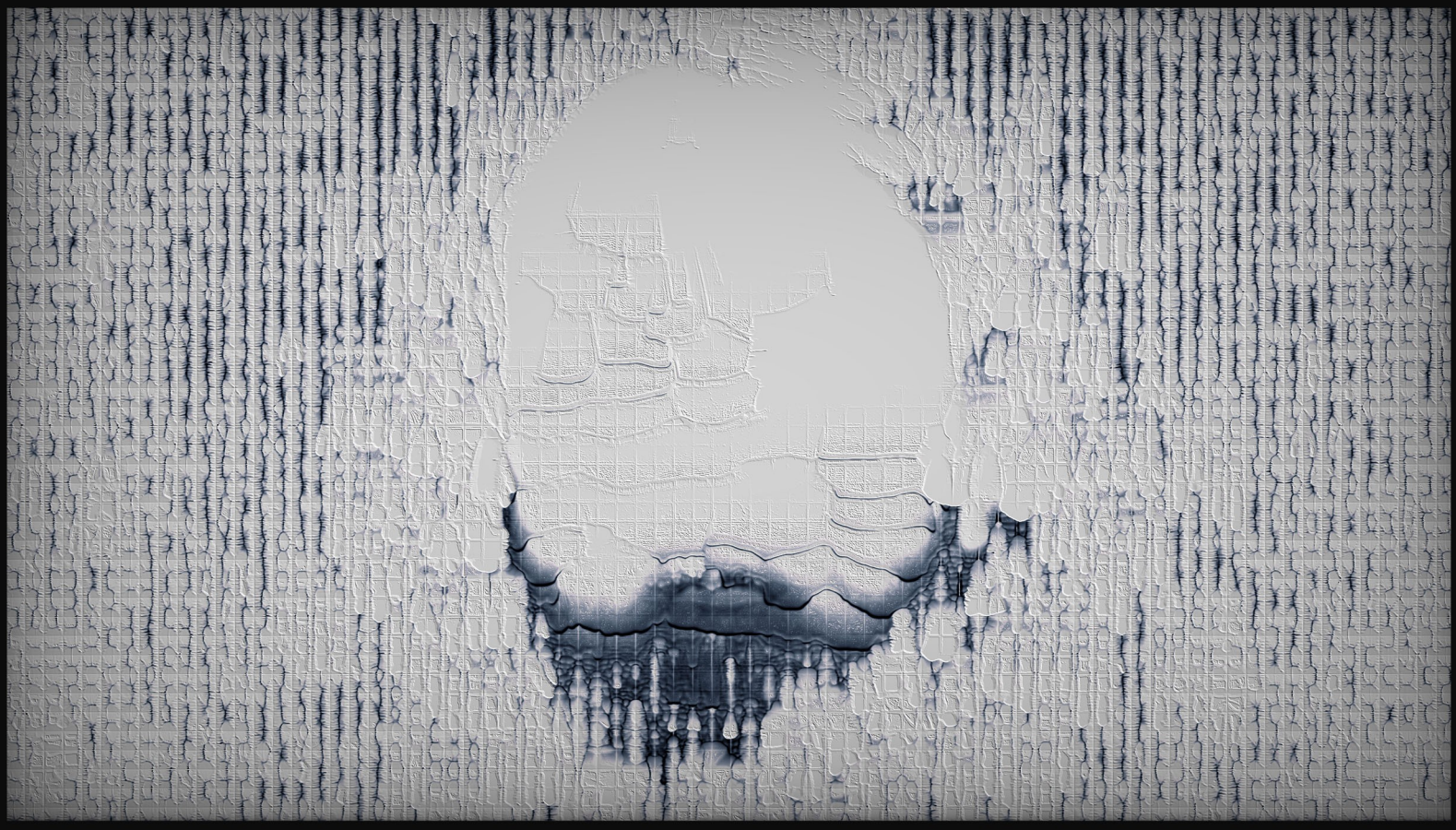


FRAGMENT:FLOW V.1

USER GUIDE

CONTENTS

Introduction, Activation and Launch 3

Image Inputs 6
- Video/Image Input 7
- Camera – Webcam 7
- Spout In 7
- Effects & Image Adjustments 7

Alpha Channels 10

The Core Shader 10

The Noise Seed 11

Feedback Effects 12

Post Shader Effects 13

Display & Resolution Management 14

Audio 15
- Setting up your Audio device... 16
- Audio In & Sound File Playback .....

Modulation Sources 17
- Audio Mods 17
- LFOs 18

Midi & OpenSoundControl 19

Storing Presets & Preset Management 20

- Media Location & Sharing Presets 21

INTRODUCTION

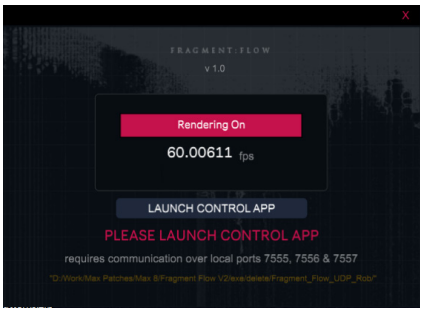
Thank you for choosing Fragment:Flow. Fragment:Flow is a versatile real-time audio-visual platform created using Cycling74's Max 8. At its heart lies a complex and flexible processing chain built around a core shader. Both the Core Shader and the surrounding effect slots can dynamically load GLSL code in the form of jxs shader files exported from Max (or ISF files in the case of FX slots), providing a flexible and growing tool-box for real-time visual design. The comprehensive audio modulation, midi and OpenSoundControl implementation, as well as its robust preset storing system, makes it a valuable tool for both performance and production settings.

After reading this guide, your next step should be to check out the various factory presets provided by Rhizome78, paying careful attention to the combination of effects and the values that are being used. Using Fragment:Flow is best approached as an exploratory process and often the most satisfying results are achieved by making fine, subtle adjustments to the parameters (you'll frequently be working behind the decimal point… and in some cases several points behind!). As such, it is highly recommended that you explore and prepare your visuals using keyboard and mouse to begin with. Once the desired value ranges are established for a given "set", you can then assign your midi/osc controllers and audio modulation settings and scale them appropriately. All of these settings will be stored whenever you save a preset, ready for performance.

ACTIVATION & LAUNCH

When you first launch Fragment:Flow you'll be presented with a small window with a field for entering the email associated with your account and your serial-key. Make sure that you are connected to the internet and that you have allowed Fragment:Flow through your firewall, carefully enter your email and hit return/enter. Do the same for the serial key provided when you purchased the software and finally click ACTIVATE. After a short server check the software should be activated. If you have any issues please do not hesitate to contact me at support@fragmentflow.com.

Once activated, click "start". This is the Fragment:Flow renderer app. The software consists of 2 distinct exes that communicate over local UDP ports (again, please ensure that your firewall isn't causing issues). From here, you'll need to start the Control App using the "launch control app" button:



The control app can take a while to load, so please be patient. Once loaded, the renderer will confirm the connection and you'll have full control over all of Fragment:Flow's features.

Note: There are known issues with certain anti-virus software (Avast), so please allow any scans to finish before attempting to proceed during the initial install/launch (affects both installer and FF's main apps). Fragment:Flow is simply a series of Max patches bundled with the official Max 8 runtime. After the first run/scan is complete, the software should start normally from then on.

The remainder of this document offers a step-by-step guide to each section.

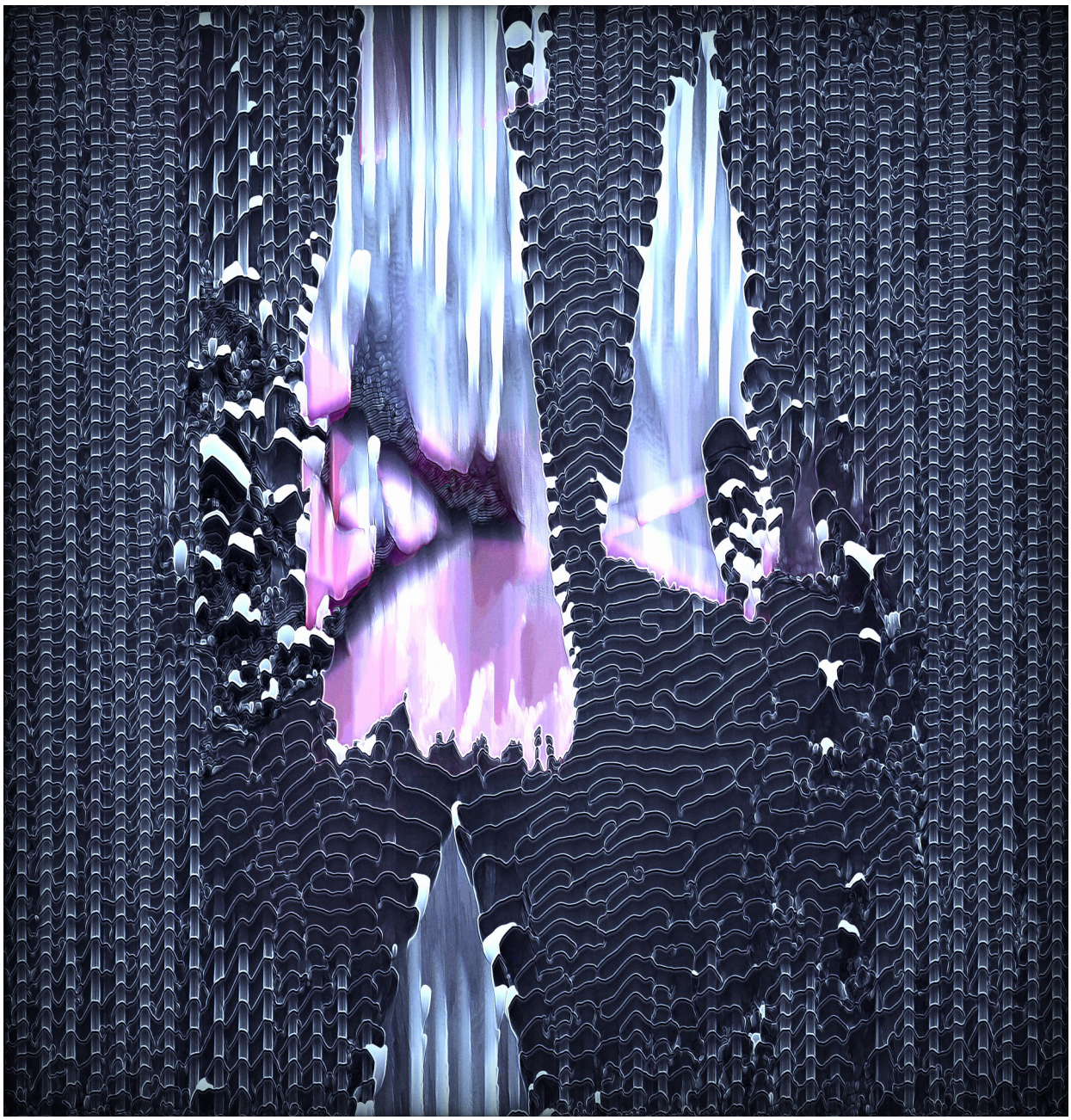
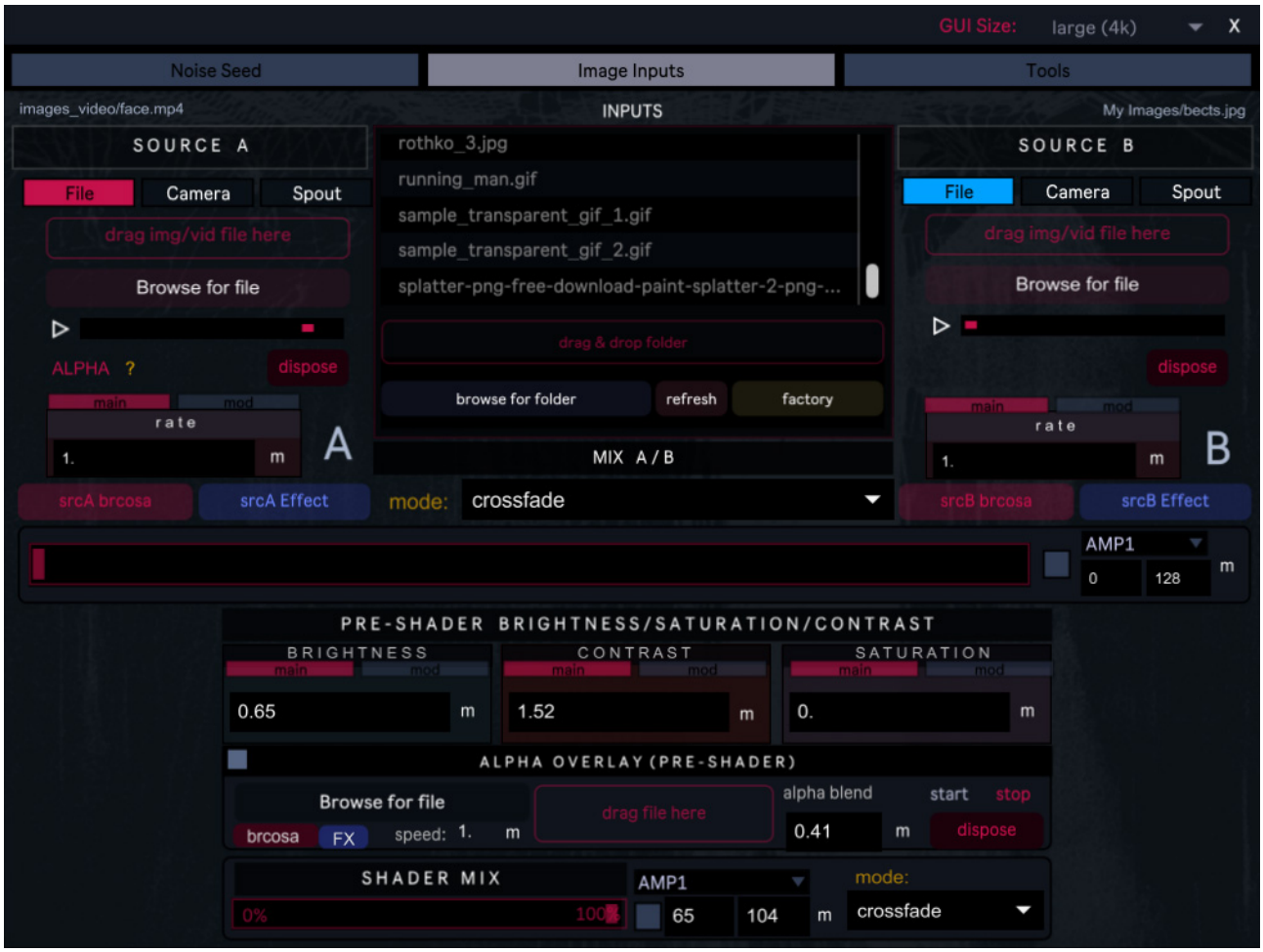


IMAGE INPUT



The **Image Inputs** section provides two slots which can independently host a media file (video/image), stream footage from digitized video (camera/webcam) or receive a texture via a Spout enabled app. Choosing the desired input type is a simple matter of clicking the corresponding tabs for Source A and B. The content of these two slots can be cross-faded or blended together using a range of Photoshop style blend modes. Blending methods are selected using the **“mode”** drop-down menu and a horizontal slider is provided for hands-on mixing of the two sources. The default blend-mode is “cross-fade”, which is a simple linear mix. As with all of the essential elements in Fragment:Flow, this slider is exposed to audio modulation, the included LFOs, midi and OSC mapping (details can be found in the *Modulation Sources* and *Midi & OpenSoundControl* sections below). At the very bottom you’ll find a slider to mix the pre and post shader outputs and a drop-down menu to select your blending mode (more details below).

As with other sections, the availability of the input section is determined by the nature of the Core Shader that is currently loaded. Aside from image processing shaders, Fragment:Flow is also capable of running purely generative code such as this example from Shadertoy (once ported using Max), in which case the input section is disabled. At release all of the factory shaders will be of the image processing type, but varied shader packs are planned for the future.

Video/Image Input

Media files can be loaded into either slot by clicking “browse for file” or by dragging and dropping files into the red delineated sections directly above. In the middle section users can select a folder and generate a list of its contents for drag-and-drop functionality. Accepted image and video file types are: **mp4**, **avi, mov**, **jpg**, **bmp**, **png**, **gif**, **pict.** In cases where your source material has alpha data (transparency, such as gifs or pngs), you may load that file into Source A and select the **alpha_blend** mode. In this case, Source B will provide the background and Source A the foreground. Moving the mixing slider to the right will gradually fade in Source A. Interesting results can be achieved with this method by applying effects to Source A while the background remains unaltered, or vice versa (see *Input Section > Effects & Image Adjustments* below).

Note: The final output resolution of Fragment:Flow is decoupled from the source material resolution for the sake of flexibility, and in the case of Reaction Diffusion shaders the resolution of the **Noise Seed** has a much greater impact on the perceived level of detail than the source material does. So, for example, the visual difference between a 720p and 1080p video running with a high resolution Noise Seed and an internal texture of 4K will be minimal. Users should experiment to achieve the optimal performance when using such shaders.

Camera Input

Accepts input from any installed video digitizer (such as a webcam). Populate the drop-down menu by clicking “get device list”, select the desired device and press “start” to begin streaming. Note that only one slot can use a given device at a time.

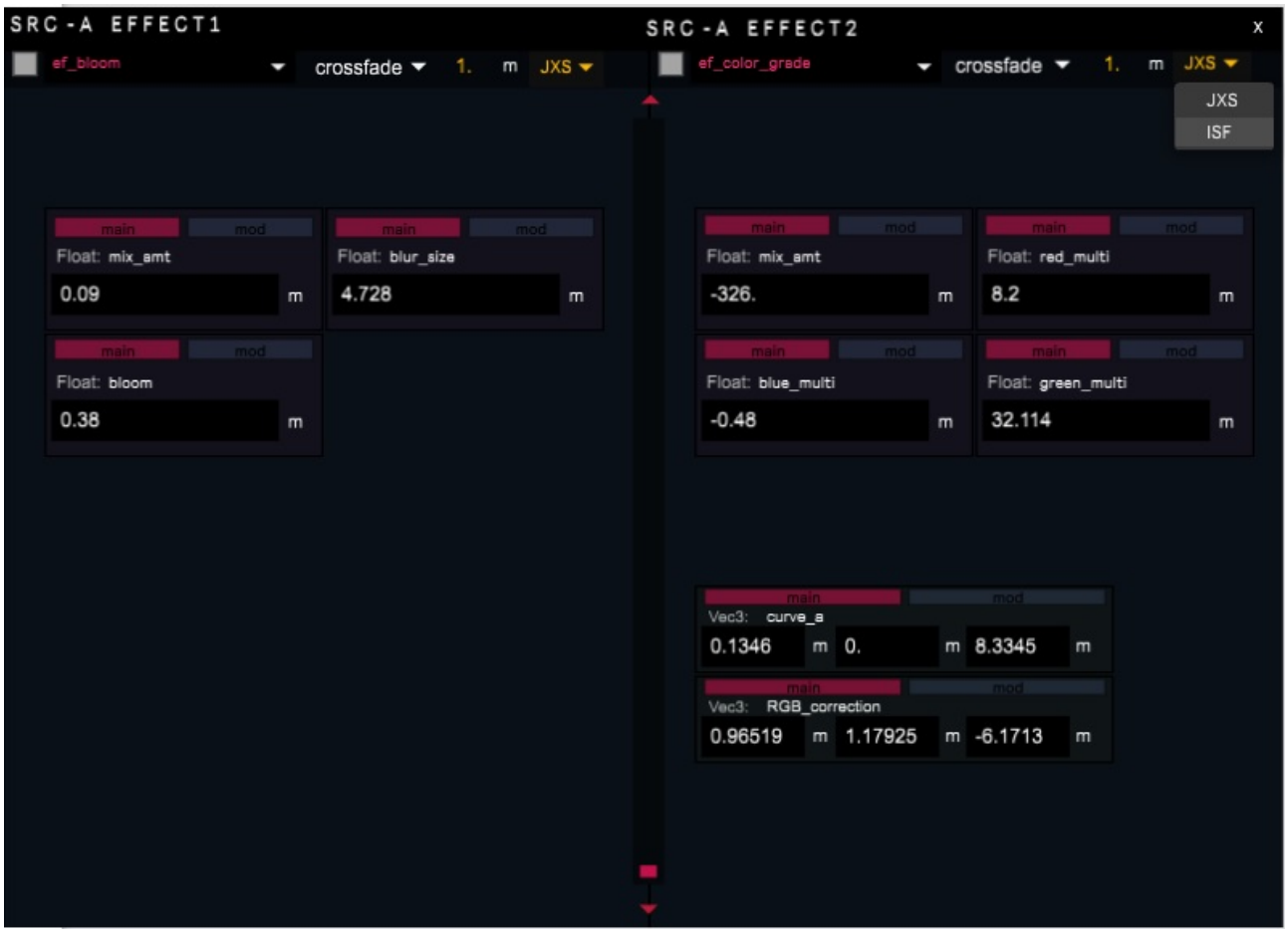
Spout In

Accepts a texture broadcast from any Spout enabled app. Launch and initiate your sender app, click “getavailablesenders” and select the desired texture from the drop-down menu. Each slot can receive from both the same or different Spout senders (the former is useful when you want to add different effects to the same source and cross-fade between them).

Input Section: Effects & Image Adjustments

The input section provides a number of effect slots and image adjustment utilities. Both source A and B are equipped with their own **brightness, contrast, saturation** and **hue** sections (**brcosa**), as well as two effects slots each. The number of available effects in these slots is determined by the shader files installed in the app’s **effects folder** (the effects installed here will be available at all effects sections throughout the app) and any installed ISF files (see below). These sections are accessed by clicking the corresponding red and blue buttons in section A or B, launching a floating window where you can access all of the controls and set up modulation, midi and osc mapping:

The brcosa sections should be self-explanatory, but the **effect slots** warrant closer inspection as they provide an introduction to basic functionality found throughout the app:



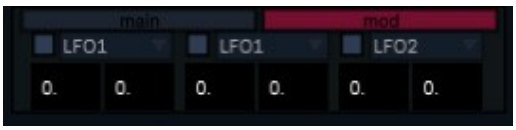
Input Section Effects, Mapping & Modulation

In the image above you can see a grey toggle (square, top left) which enables/disables the effect and next to it a drop down menu in red which allows you to select the desired shader. To the right you'll find a second drop-down menu used to specify the **blend-mode** for that effect (white), and a number box that allows you to define the **mix amount** (yellow). Finally, a further menu switches between JXS and ISF effect types (see below for details regarding ISF shader support).

Upon selection the section below will be populated by a series of "control boxes" which correspond to the parameters accepted by the shader. In the Src-A Effect1 example above, the chosen effect is a *bloom* which accepts the user-defined parameters "mix_amt", "blur_size" and "bloom" (floats).

Each box has 2 sections, "**main**" and "**mod**", which can be toggled using the tab along the top. The **main** section allows for direct user input with mouse or keyboard. Clicking the small "**m**" next to a number box launches a floating Window where you can set up **Midi/OSC** mapping for that value (refer to the relevant section in this document for more details).

The **mod** section gives you access to the modulation options for each value (the number of values depends on whether the parameter is a float, vec2, vec3 etc).



In the image above you can see the modulation options for a vec3 parameter. Each value (of 3) has a toggle to enable/disable modulation, a grey drop-down menu to select the modulation source and two number boxes which allow you to define the modulation range (min/max). When a value is being modulated, user input and the midi/osc mapping sections are disabled. As noted previously, interesting behaviour often lies within a very narrow range, so it's advisable to determine the appropriate values beforehand. Given the number of possible variables and states, it's quite easy for beautiful emergent behaviour to be lost without due care.

As well as providing image adjustments for Src A and B individually, this section also offers brightness, contrast and saturation controls at the stage after A + B have been combined, and employs the same essential mapping/modulation methods as above:

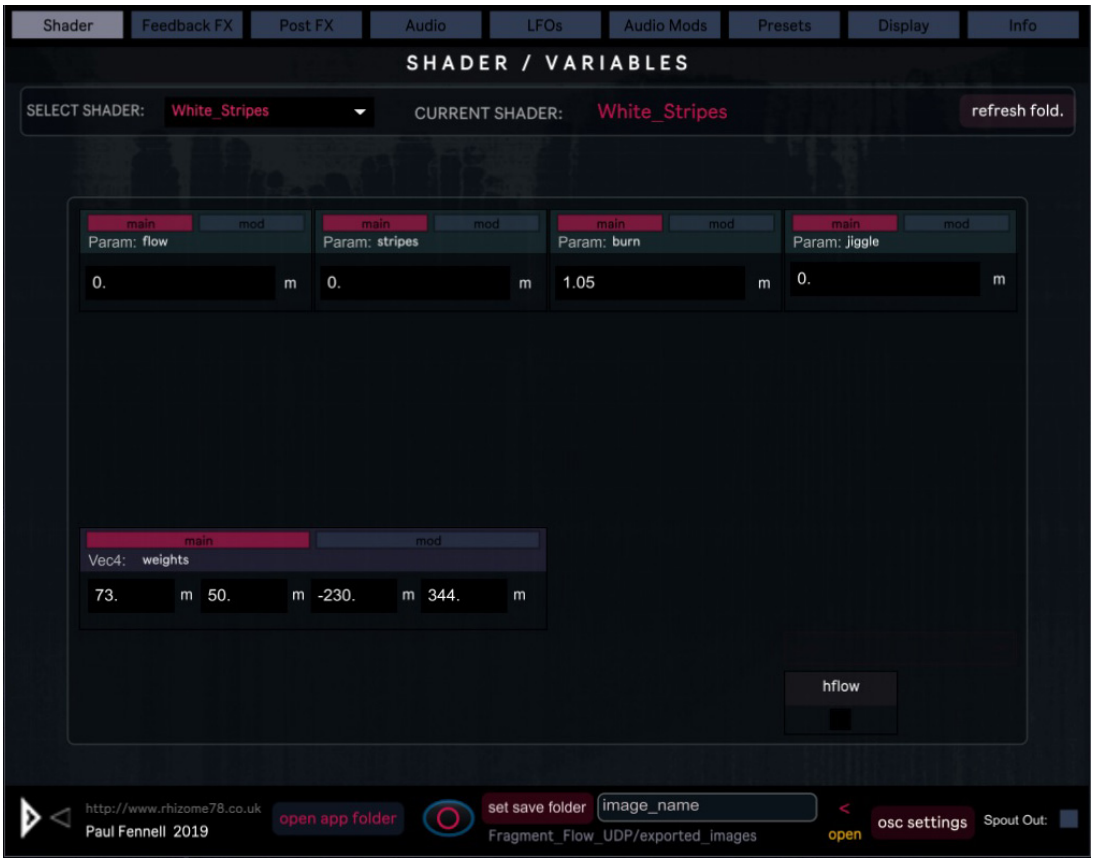
HINT: *When using reaction diffusion or feedback based shaders the brcosa controls are often key. Excessive brightness can cause a "white out" effect and the shaders themselves often respond dramatically to varying values. Modulating these with audio can be very effective, provided that you get the modulation ranges right. During development and as part of my personal work, I find myself flittering constantly between these values and the corresponding controls for Source A and B to find the perfect balance between the sources and the combined output going into the shader.*

ALPHA CHANNELS

Aside from the method outlined above, two dedicated alpha slots are provided for loading material with transparency data (such as pngs or gifs). The first resides directly beneath the pre-shader **brcosa** section and is applied just before the combined result of Source A and B enters the shader. A subsequent alpha slot is provided at the end of the processing chain (**Post FX > Misc Post Processing**) and is designed chiefly for the use of logos and other overlays. Both sections offer speed control (if animated), mix amount (alpha blend), drag-and-drop functionality and "start" and "stop" controls. The "dispose" button will discard the loaded file. As previously described, clicking the small "m" next to a parameter will launch a floating window where you can access the midi/osc mapping options, but in this case you'll also find the modulation options there (a space-saving measure). There is also a dedicated brcosa section and two effect slots per alpha slot, accessed by clicking the corresponding red and blue buttons. Alpha channel functionality has been tested with PNGs, gifs and videos encoded with the **HAP Alpha codec** (be sure to export at 32-bit depth).

THE CORE SHADER

At the heart of Fragment:Flow resides the **core shader**. Much like the effects sections, selecting the desired shader will automatically assign a series of "control boxes" that correspond to the parameters accepted by the code. Unlike other sections, the core shader determines which sections of Fragment:Flow are available and alters the underlying structure of the software according to requirements. So, for example, most reaction diffusion based shaders utilise the **Noise Seed** section, and any shader that has a feedback loop will utilise the "**Feedback Effects**" section. Some shaders may be purely generative and not support the **Image Inputs** section at all. The shader section is accessed by clicking the "Shader" tab in the bottom half of the GUI:

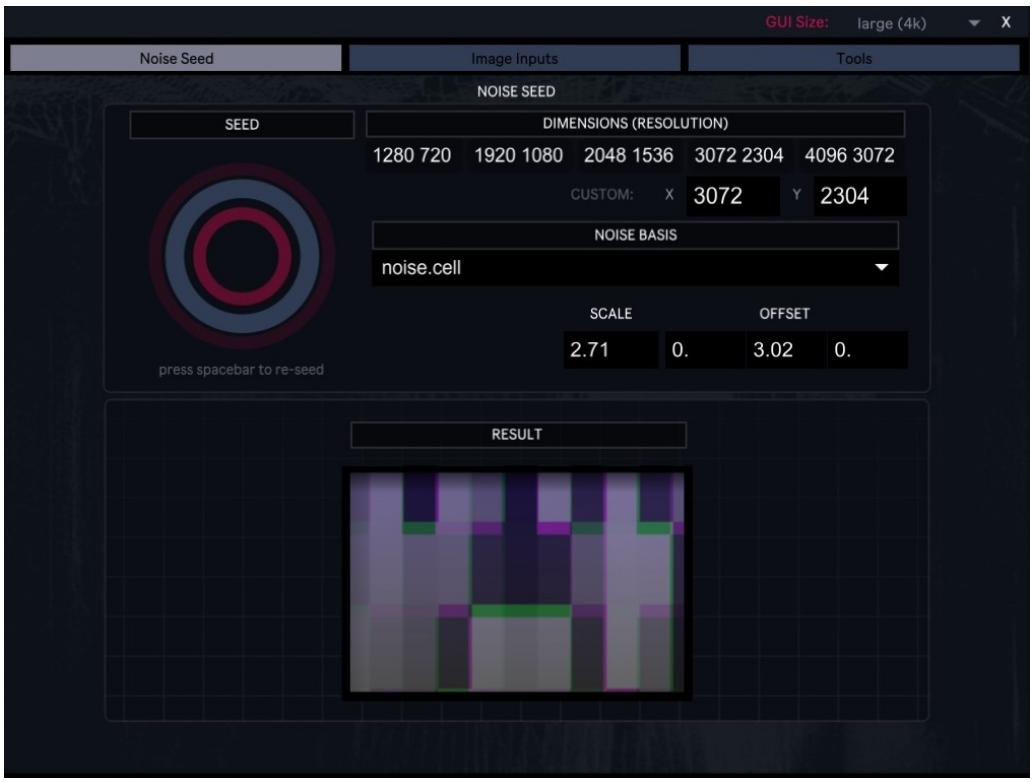


You can select the desired shader from the drop-down menu, *but the recommended way to start is to select one of the related factory presets* (it can take a while to set-up a shader with suitable values). The modulation options are accessed in the same way as previously described, by clicking the "**mod**" tab in each control box. Similarly, clicking the "**m**" allows you to set up midi and OSC mapping for that value.

When using a shader that processes media from Source A+B you can mix between the pre and post shader output using the "**Shader Mix**" slider provided (dry/wet) and define the **blend-mode** using the drop-down menu. The blend-mode determines how the processed image from the Core Shader is applied over the original input. **Cross-fade** is the default and represents a simple linear mix (0-100%). Since this is only relevant when the shader actually processes images and video, these controls are located at the bottom of the **Image Inputs** section. Again, the mix amount can be modulated or mapped to midi/OSC.

THE NOISE SEED

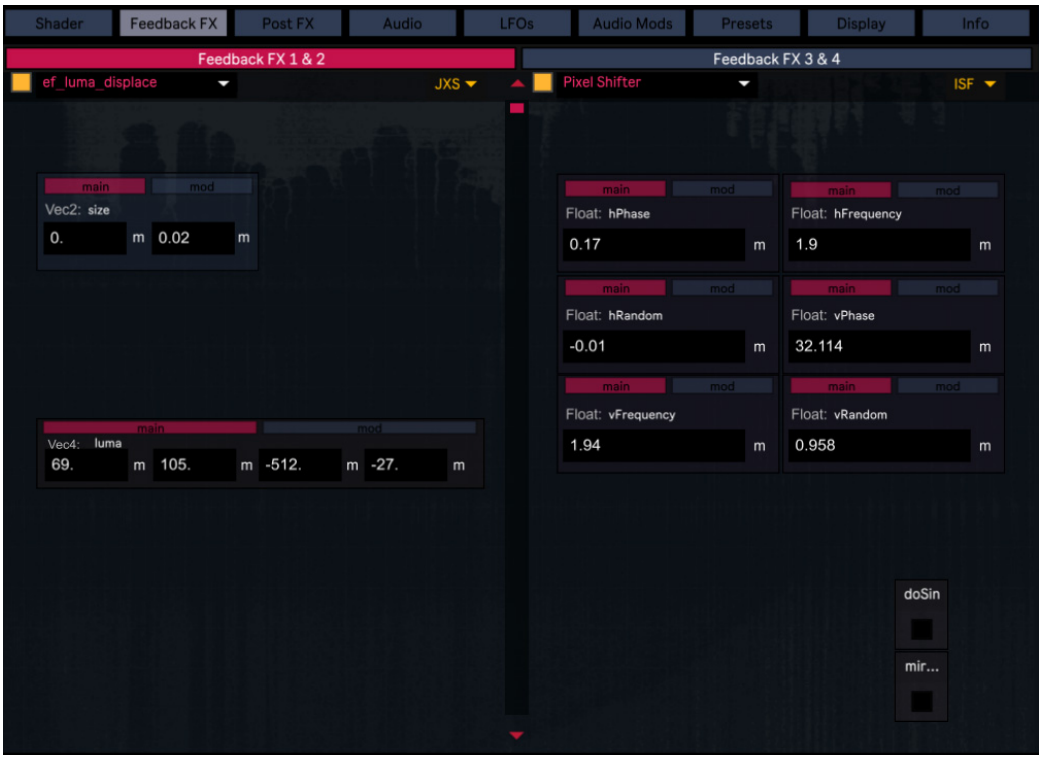
The **Noise Seed** section can be accessed by clicking the corresponding tab in the upper half of the GUI. Many shaders, particularly reaction diffusion based ones, utilise the Noise Seed to provide a set of values that initiate the simulation. The noise seed can be re-initiated at any time by pressing your **spacebar** (re-seed). Indeed, you should get into the habit of doing this quite frequently, especially when saving presets. Since many shaders are organic systems whose state evolves directly from their previous state (which often results from complex, "unrepeatable" sequences of user interactions), re-seeding will show you exactly how the preset will look when recalled.

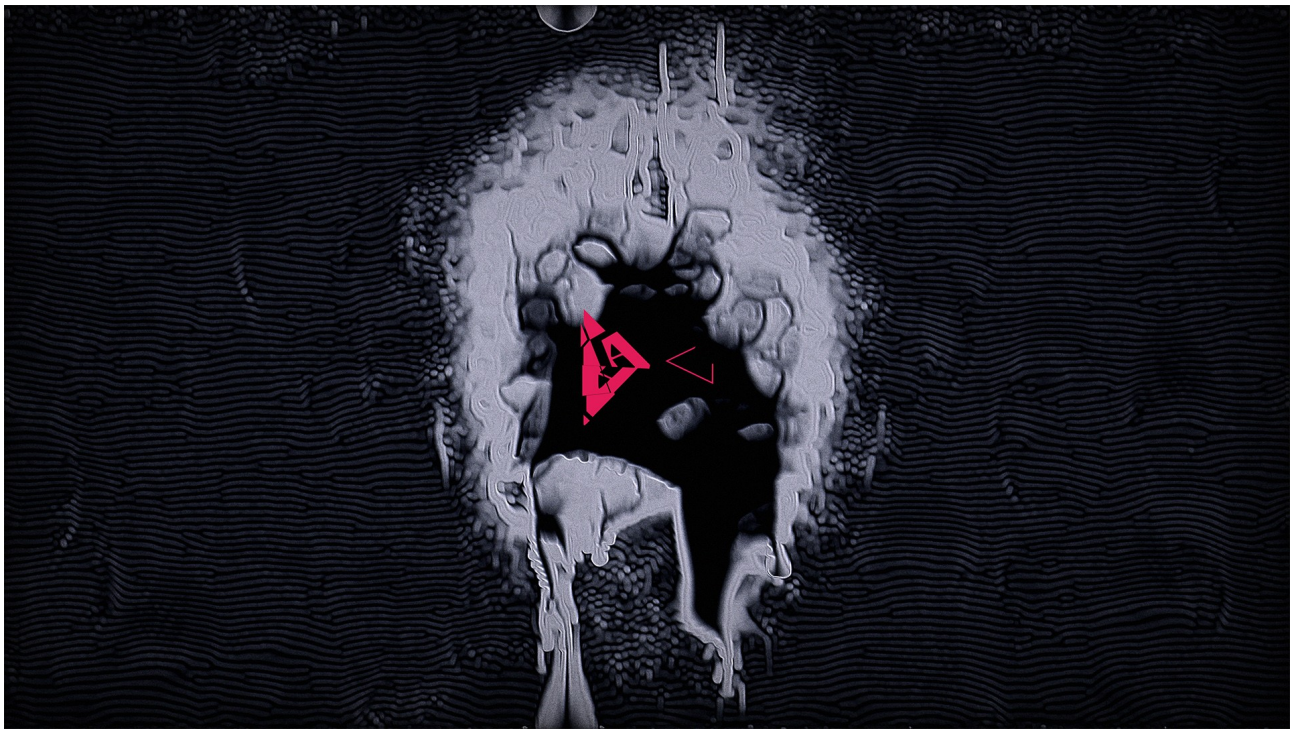


The Noise Seed section offers numerous options to control the resolution, “noise basis” (the type of noise) and scale/offset of the seed. The extent to which these variables impact the behaviour of the shader varies between shaders. The most important thing to note is that the resolution of the Noise Seed has a significant impact on the perceived level of detail with such shaders, and that **higher values put greater strain on your system**. The noise resolution can also directly alter the behaviour of certain shaders, and indeed lower resolutions can sometimes yield very desirable results. As ever, free experimentation is the best approach. You can set the resolution by clicking the pre-defined values or by manually entering the desired values into the X/Y number boxes, but *please be aware that very high values may cause severe slow-downs and stability issues*. If you’re experiencing performance issues and the shader utilises the Noise Seed, reducing the resolution here will often help.

FEEDBACK EFFECTS

Found in the lower half of the GUI, this is arguably the most interesting section of Fragment:Flow and is applicable to any Core Shader that employs a feedback loop. As the name implies, the Feedback Effects section consists of a series of effect slots placed between the Core Shader’s output and its feedback input. Since the effects are applied recursively, extremely interesting, organic and sometimes chaotic results can be achieved. There are four slots available and they function in a similar manner to the effect slots previously described, although they lack the blend-mode functionality. Like other sections all of the effects installed in Fragment:Flow’s effects folder will be available, but displacement effects are particularly useful here. Since pixels are being displaced at every iteration the result is a wide array of behaviours akin to video feedback and other recursive phenomena. More than anywhere else, the Feedback section repays a *lightness of touch*, with very small sub-decimal changes frequently yielding dramatic results. The possible behaviours are too broad to cover here, so the best way to start is to load a relevant factory preset and examine the combination of effects that are being used. From here, make subtle adjustments to the values, behind the decimal point to begin with, to get a feel for how it responds. Like other sections, the parameter control boxes are initiated when the shader is chosen and provide the full range of modulation and midi/OSC mapping options. Again, with shaders that do not possess a feedback loop this section will be disabled entirely.





* example of reaction diffusion shader (Black Ink) with displacements applied at the Feedback Effects stage (luma displacement)

POST SHADER EFFECTS

The **Post FX** section, found next to the Feedback Section in the lower half of the GUI, offers a series of effect slots and general image utilities that are applied after the Core Shader, en route to the final output window. There are 4 effect slots which function in the same way as those previously described, whereby the available effects are determined by the files installed in Fragment:Flow's effects folder or ISF files installed on the system (see below). Unlike the Feedback Effects, the Post Shader effects allow you to specify a **blend-mode** in the same manner as the effects slots dedicated to Source A+B. So, for example, applying an emboss effect with the **overlay** blend-mode yields the illusion of texture and relief whilst maintaining the original colour of the input (Photoshop users will be at home here).

In addition to these 4 user defined slots, the **Misc Post Processing** section provides a set of hard-wired general tools which are applied just before the final output. These include:

- Brightness, Contrast, Saturation and Hue adjustment
- X/Y mirroring
- A noise overlay
- A final Alpha channel (as described above).
- A Vignette Effect

The Brightness, Contrast and Saturation controls here are particularly useful. The brcosa controls earlier in the processing chain often have a significant impact on how the core shader behaves, whereas here they are applied globally at the final stages of the processing chain. As such, they should be your go-to for general image adjustments.

HINT: Taken together, the combined Source A+B and Post Shader effect slots represent a versatile framework in their own right, and there may be occasions when you wish to bypass the Core Shader entirely

and just utilise these. This is achievable with any given Core Shader simply by moving the Shader Mix slider to the left (0% shader), but a much more efficient method is to use the dedicated Pass-Thru-No-Effect shader. As the name suggests, this simply passes the input directly to the output without further processing, whereas in the case of the Shader Mix method the active Core Shader continues processing unseen in the background.

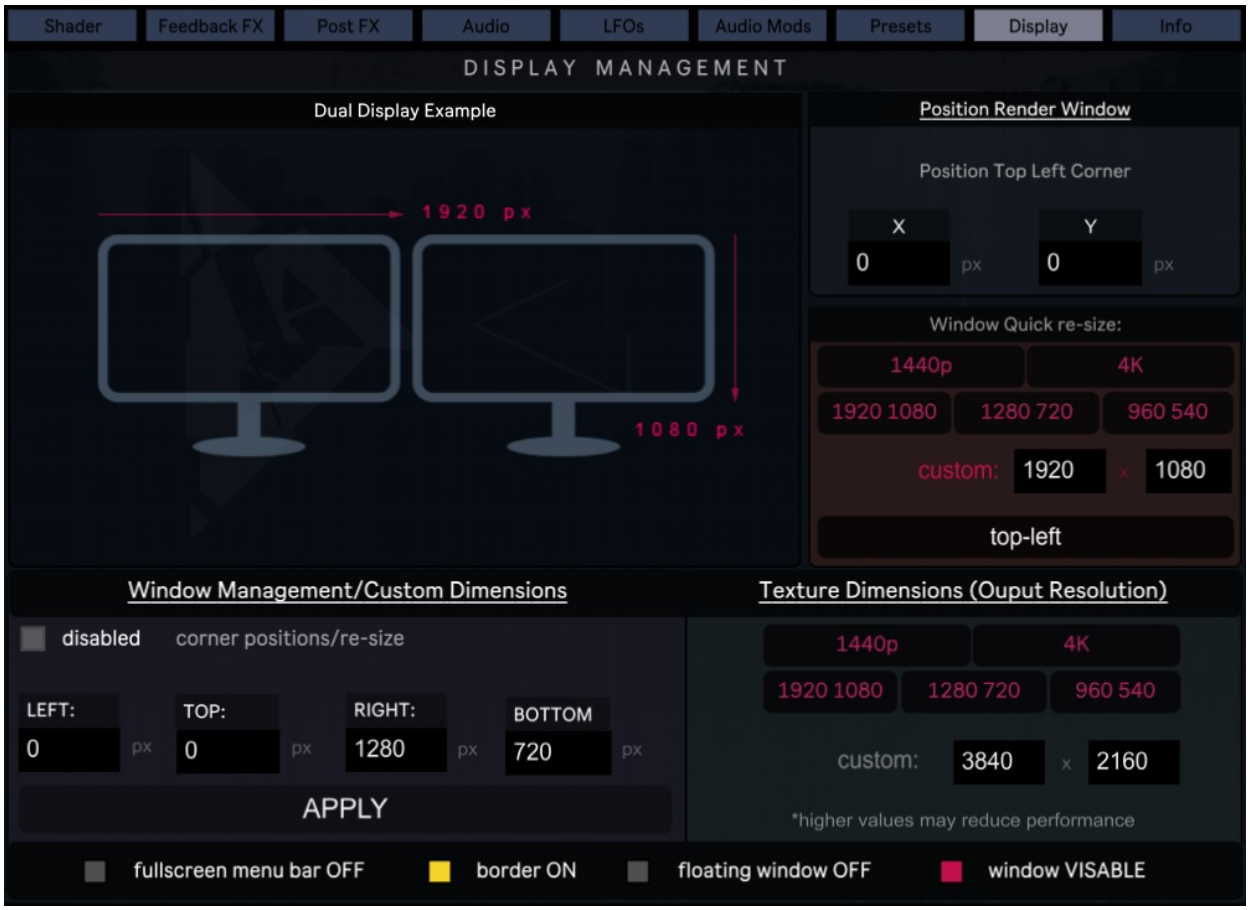
ISF SUPPORT

Fragment:Flow supports ISF filter effects thanks to the Max ISF package from Vidvox. The best way to get started with ISF is to install the free ISF Editor from here which ships with a large library of shaders. Fragment:Flow expects ISF files to be installed in the default *C:/ProgramData/ISF* folder. Once installed to this location, users can select “ISF” from the JXS/ISF drop-down menu and select the desire filter effect.

Note: At present only filter/processing ISF effects are supported for use in Fragment:Flow’s various FX slots, but a full implementation that will allow for the use of generative core shaders is planned for the future.

DISPLAY & RESOLUTION MANAGEMENT

The Display & Resolution Management section allows for precise positioning and scaling of the display window and control of the internal texture resolution.



Window size and internal texture resolution has been decoupled to allow for greater flexibility in terms of optimising performance (needless to say, higher internal resolutions are more demanding), so when specifying custom window sizes be sure to also set the texture resolution as desired.

Note: when **exporting images** (described below) the dimensions of the resulting file are determined by the Texture Dimension settings rather than window size, so be sure to set it appropriately beforehand.

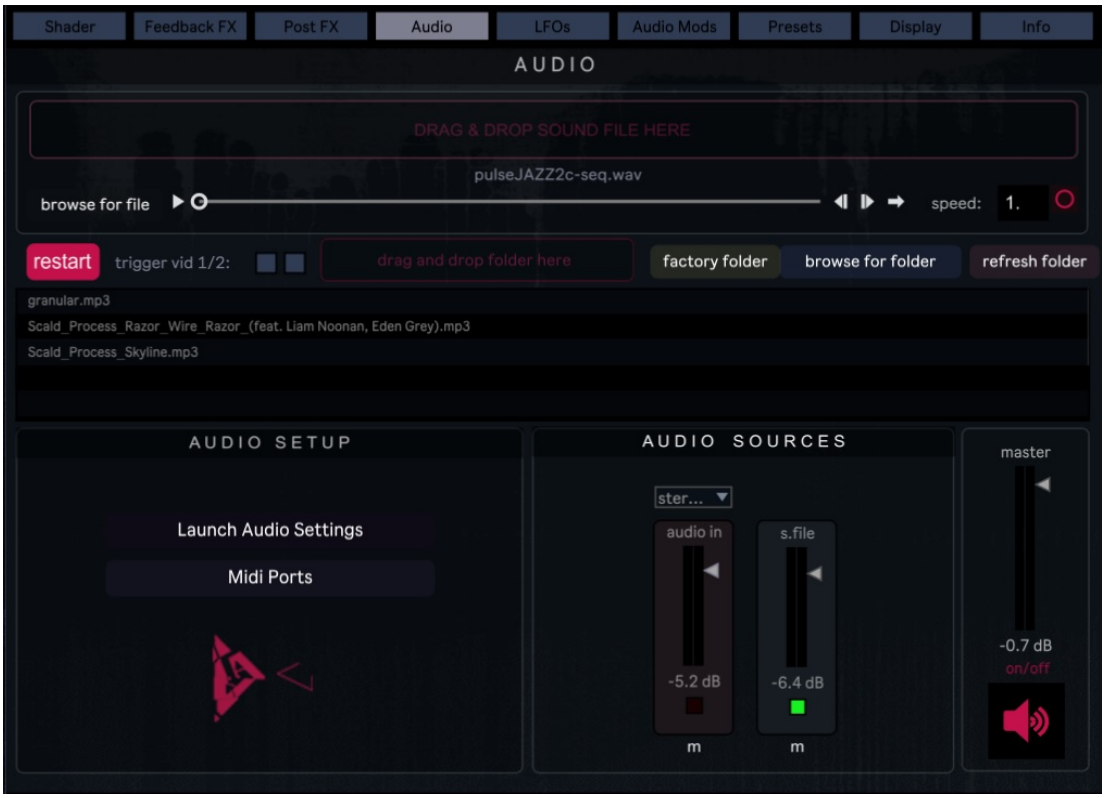
The display window can be positioned and scaled using a mouse, but the various controls found here are useful for more precise arrangement, such as setting up dual displays and projections. The **Window Quick Re-size** section allows you to quickly assign a set of common dimensions without specifying the precise position of the window, whereas the **Window Management/Custom Dimensions** section allows you to precisely place each corner of the output window on your display (thereby altering both its position and scale). The **Position Render Window** section simply shifts the window along the x and y axis without altering its size. Along the bottom there are a series of toggles to enable/disable the window bar and border, set the window to “floating” (always on top) and hide the window entirely. The latter may be desirable when you’re streaming the output to another app via **Spout** and do not require a separate display. Please note that when setting the window to “floating”, “always on top” means exactly that and it is possible to obstruct Fragment:Flows GUI. In this case, you can press “f” on your keyboard to toggle the floating option.

Note: You can toggle full-screen mode by pressing ESC on your keyboard. However, the manual positioning tools are often the better method.

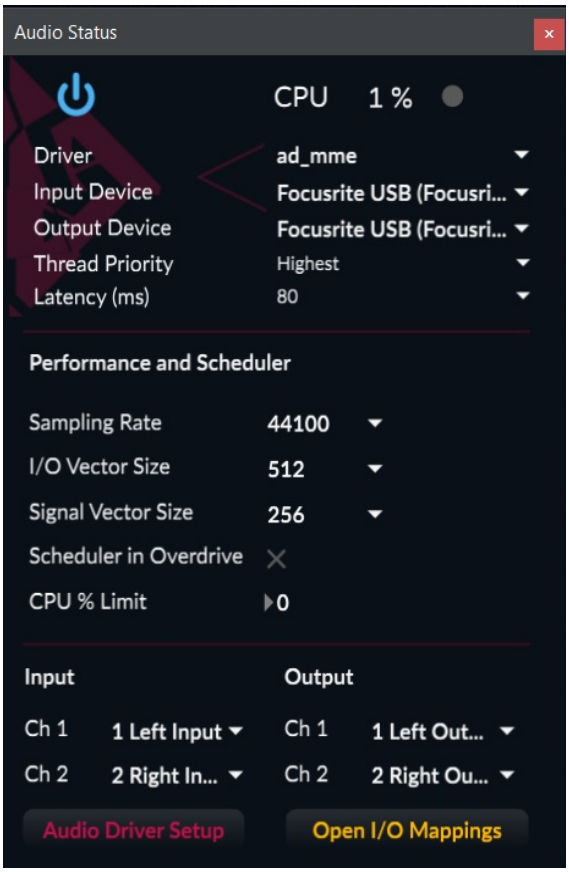
AUDIO

The audio section provides access to your audio device settings, audio routing and sound file playback. Here you will also find volume adjustment sliders and a global enable/disable audio toggle (the speaker icon, bottom left).

SETTING UP YOUR AUDIO DEVICE



Launching the **Audio Settings** window allows select your desired audio device and input/output channels, as well as access advanced audio settings such as sample rate, signal vector size.



AUDIO IN & SOUND FILE PLAYBACK

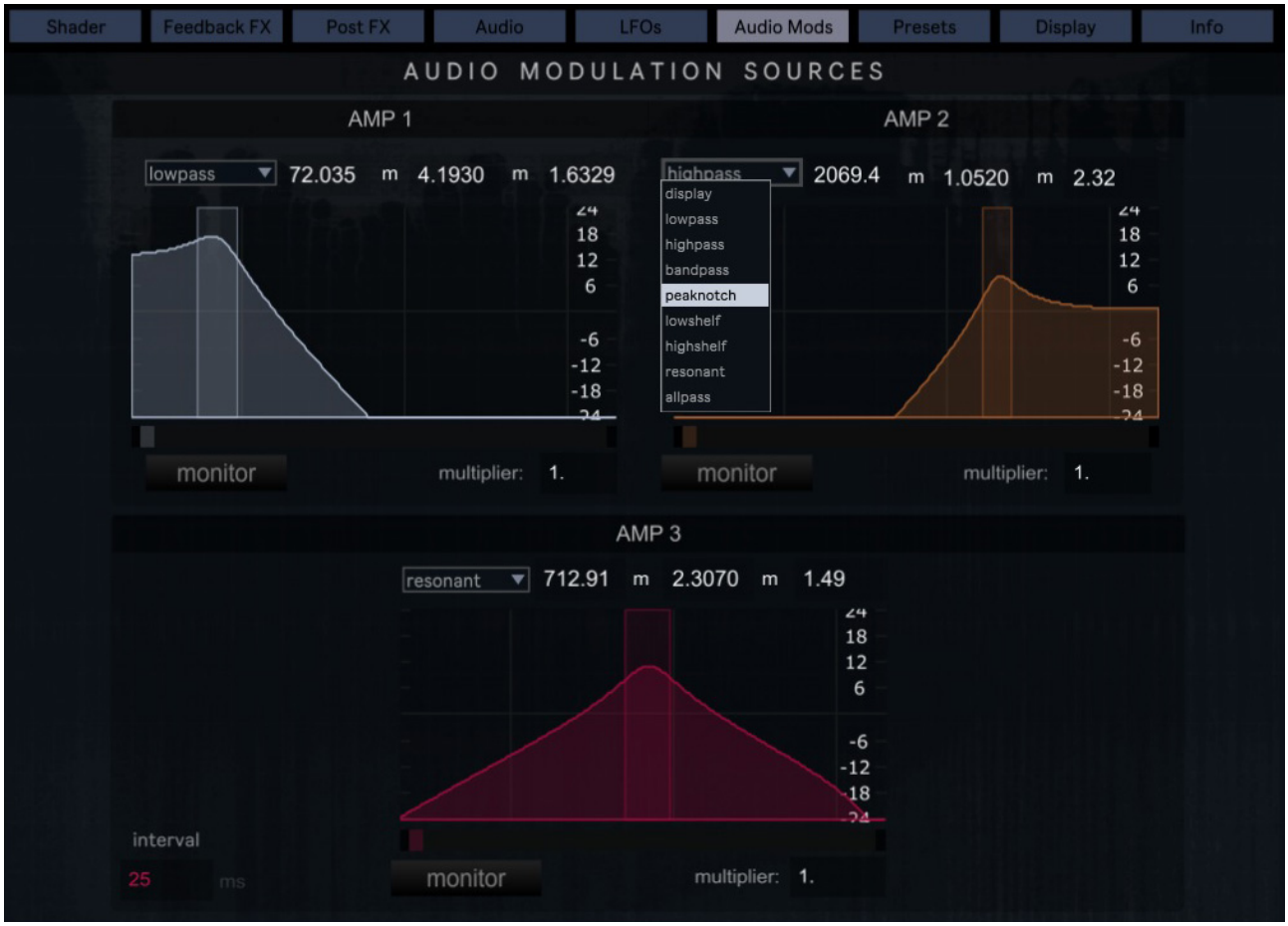
The **Audio Sources** section allows you to enable/disable audio-in and sound-file playback as well as adjust their respective volume. By default the audio input expects a stereo signal, but the drop-down menu also provides a mono option (for use with monophonic instruments etc).

The **Sound File Playback** section provides a simple interface for loading and playing sound files. Files can be loaded manually using the “browse for file” button or dragged into the red drag-and-drop section. Similarly, the contents of a folder can be loaded into the section below. The white playbar offers a simple set of controls including play/pause and looping functionality. Control of playback speed is also provided. The combined output of the audio-in and sound-file signal will be sent to the audio-modulation section for filtering and used to generate modulation sources which can be applied throughout the app.

Due to the fact that the app consists of two distinct executables it is not possible to stream the audio track directly from a video source. To circumvent this limitation, users should extract the audio track using suitable software such as ffmpeg (ensuring that the play length exactly matches the length of the video), and then load that into the sound-file section. From here, enabling the desired “trigger 1 / 2” toggle will restart the appropriate video when the sound-file is triggered, thereby syncing the two files (assuming identical play-back speeds). It’s not ideal, so more robust syncing in this outlier case may be implemented in the future.

MODULATION SOURCES

Audio Mods



Fragment:Flow is equipped with three filters which can be used to isolate frequency ranges from the signal routed to them from the audio section. The amplitude of the signal from each filter is sampled and data streams generated to be used as modulation sources throughout the app (labelled AMP1, AMP2 and AMP3). Filters can be adjusted directly using a mouse or via the number boxes directly above each filter. A drop-down menu allows users to set the appropriate filter type (low-pass for bass, high-pass for treble etc) and the

multiplier value scales the data output by the specified amount. The **monitor** button allows users to listen to the isolated audio from that filter, but *users should be careful that the amplitude isn't set too high before doing so*. The vertical height of the filter corresponds to amplitude but as a rule users should be using the data multiplier to increase the output instead. The interval value in the bottom left specifies the rate at which the signals are sampled in milliseconds. Lower values give better responsiveness, but are more computationally expensive. It's best to leave this at the default value of 25ms unless you want to increase the value for efficiency/performance reasons.

LFOS

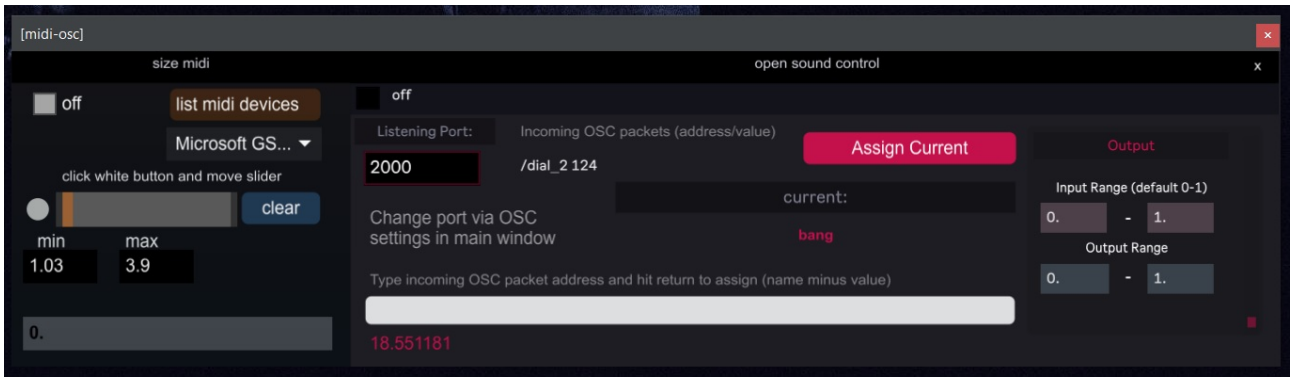
Fragment:Flow is equipped with 4 independent LFOs featuring the following wave shapes: *sine, up, down, triangle, rect, random, bin noise, random smooth*. The LFO rates can be synced to the master clock or operated in free frequency mode using the toggle button provided. The interval value specifies the rate at which the LFOs is sampled (again, lower values are more computationally expensive). This section also offers quick access to basic global transport settings.

The LFOs are available as modulation sources throughout the app. In most cases the modulation options for a given parameter are accessible via a tabbed interface, but on occasion you will find them placed directly next to the element or nested in the midi/OSC pop-up menu associated with the parameter (in cases where space is at a premium).

MIDI & OPEN SOUND CONTROL

A small “m” next to a GUI element or value indicates that it can be controlled via midi or OSC.

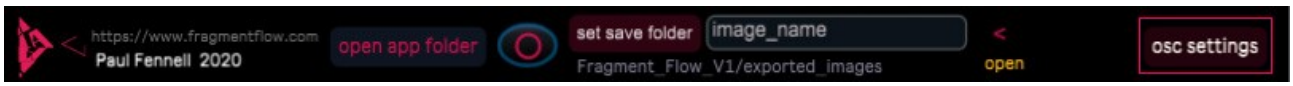
Clicking the “m” will launch a pop-up window with which to assign, manage and scale your control sources for that parameter. **NOTE:** Midi and OSC settings are stored on a per preset basis.

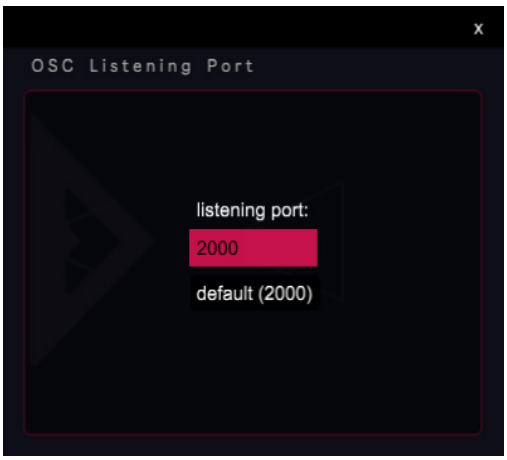


The smaller left-hand section is dedicated to **midi**. The grey toggle button enables or disables midi control and the drop-down menu allows you to select your controller from a list of installed devices. To assign your midi slider or knob, simply click the circular button to initiate midi-learn and move the desired element on your device. The min/max value boxes allow users to define the desired output range. Clicking “clear” will discard the currently assigned controller.

Located in the Audio section, the **Midi Ports** button will allow you to enable and disable midi devices globally. This is useful if you find that FF is hogging your midi devices when running other apps simultaneously (DAWs etc).

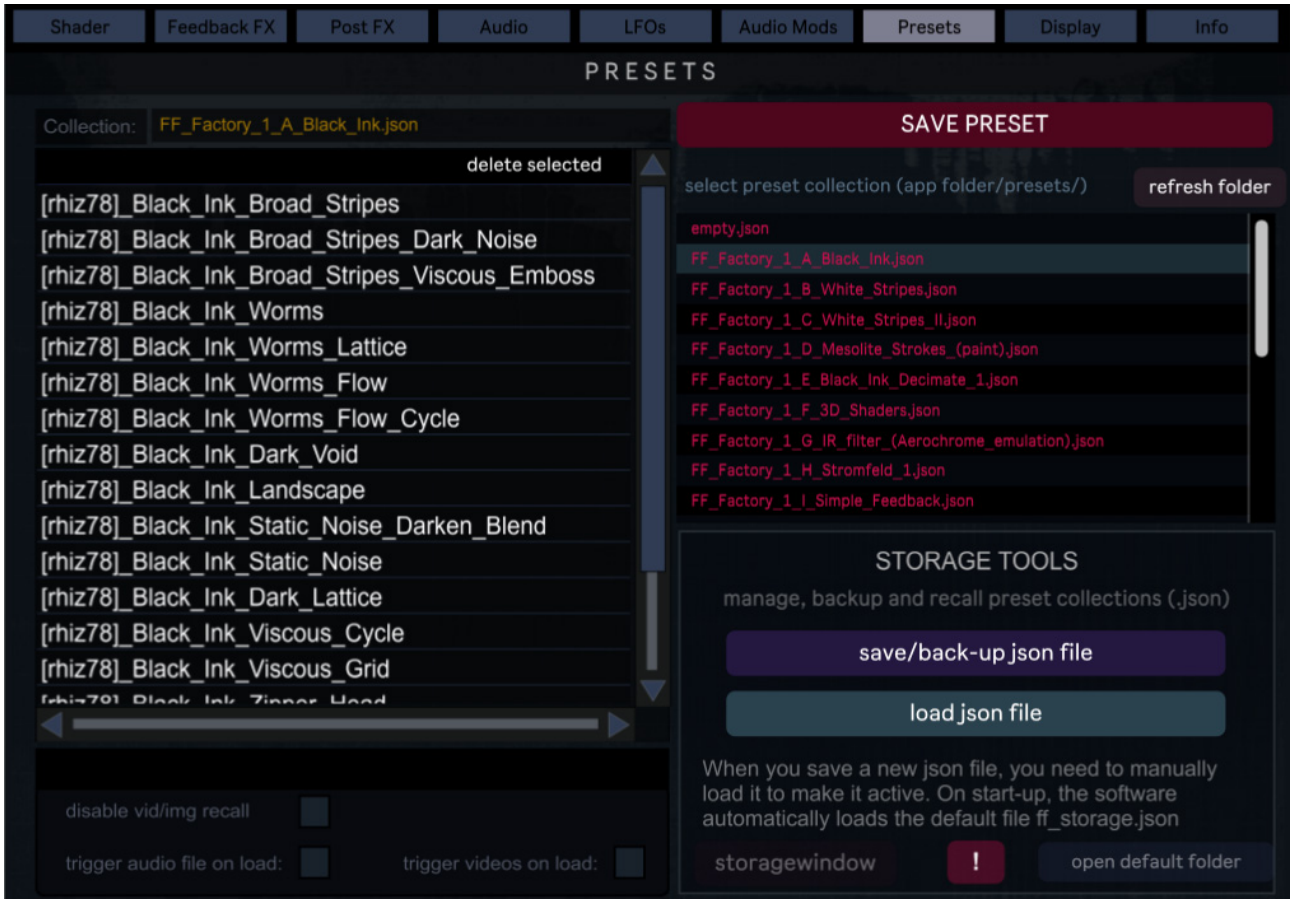
The larger left-hand section assigns OSC messages sent over UDP. Incoming OSC addresses can be assigned via the “Assign Current” button or typed manually into the white text-field shown above (eg */dial_2).* The input range allows you to specify the value range being broadcast from your OSC sender, while the output range scales the value range being sent to the parameter. The **Global Listening Port** is set by clicking the “osc settings” button at the very bottom of the *main GUI window*, as shown below.





STORING PRESETS AND PRESET MANAGEMENT

The **Preset** section allows users to create and manage “preset collections” in the form of json files. Each json file can store any number of individual presets, but it’s best to spread your work out over numerous files as loading large files can become sluggish. As a rule of thumb, 20-30 presets per json is a sensible number.



The red list on the right shows all of the preset collections stored in the apps default preset folder. The white list on the left shows the individual presets stored within the currently selected json file. Simply click on an entry to load a json file or an individual preset. Clicking "**Save Preset**" stores Fragment:Flow's present state as a new entry in the currently loaded json. Loading a json without selecting a preset will not alter the current state, so it is safe to load json files in search of an appropriate saving location without loosing the current settings.

JSON files are automatically saved when a new entry is added or the app closed, but you can back-up and re-name an active json file using the **save/backup** button. If you wish to create a new empty collection, load the provided empty.json file and save/rename to the default folder. Clicking "refresh folder" will add the newly created file to the list. Please note that when you save json files in this way, you'll need to then manually load them using the "**load json file**" button to them to render them active and start saving entries. Users can also manually copy and rename json files in windows explorer, pressing "refresh folder" to list any additions or filename changes. A button is provided to open the apps default for those who prefer this method.

MEDIA LOCATION & SHARING PRESETS

The location of loaded media files are saved with the preset but the files themselves are not. That is, if the file is deleted, renamed or its location changed the media will be unavailable to Fragment:Flow. The way in which media paths are saved varies depending upon their location. If the file resides outside of Fragment:Flow's app folder then the path will be absolute and system specific (for example *C:/My_Files/Video/movie.mp4*), whereas the path of a file loaded from Fragment:Flow's own folders will be relative (../media/images_video/movie.mp4) and therefore transferable between installations of Fragment:Flow. A dedicated media folder is provided in the main app folder for this purpose, so it's a good idea to use this to store and load media files at the start of each project. Presets saved in this way can be shared along with the required media files and, provided that the files are placed in the corresponding folders in the recipients installation, will recall perfectly. This is also useful for transferring work between installations generally.

That covers the main functionality of the software, but please feel free to get in touch if you have any specific questions. Fragment:Flow remains a work-in-progress and I have lots of ideas for updates and future expansions, so I sincerely appreciate your enthusiasm and support. Enjoy!

Best Wishes

Paul (Rhizome78)

www.fragmentflow.com

For general questions or enquiries:

contact@fragmentflow.com

Technical support:

support@fragmentflow.com