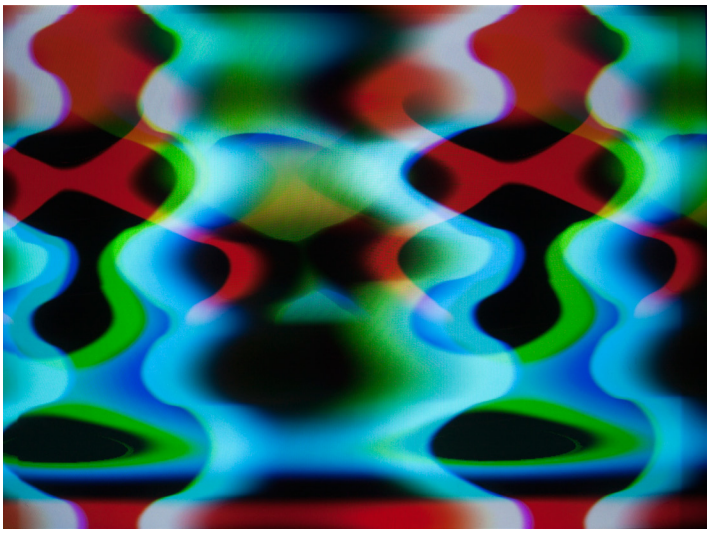
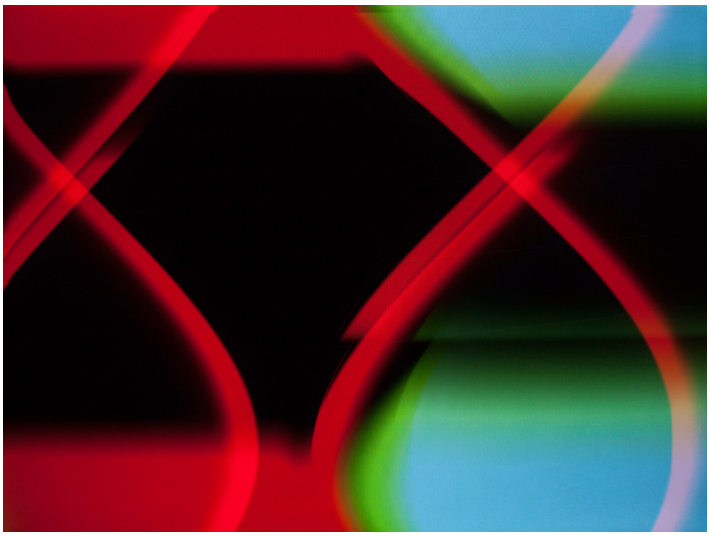
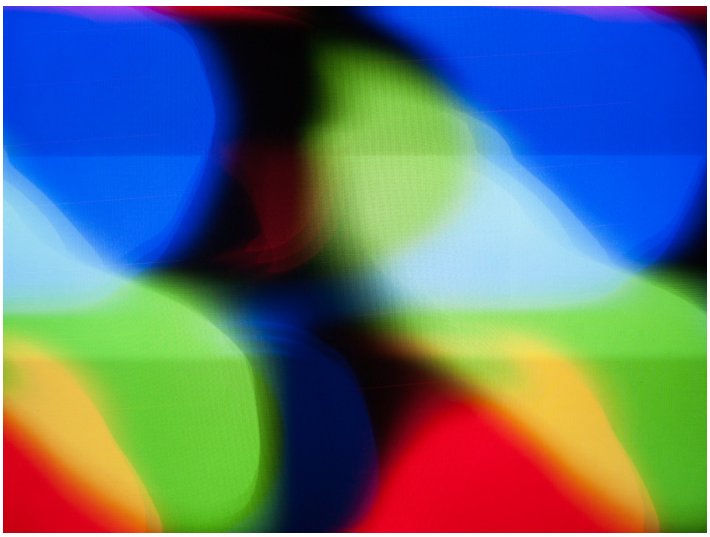
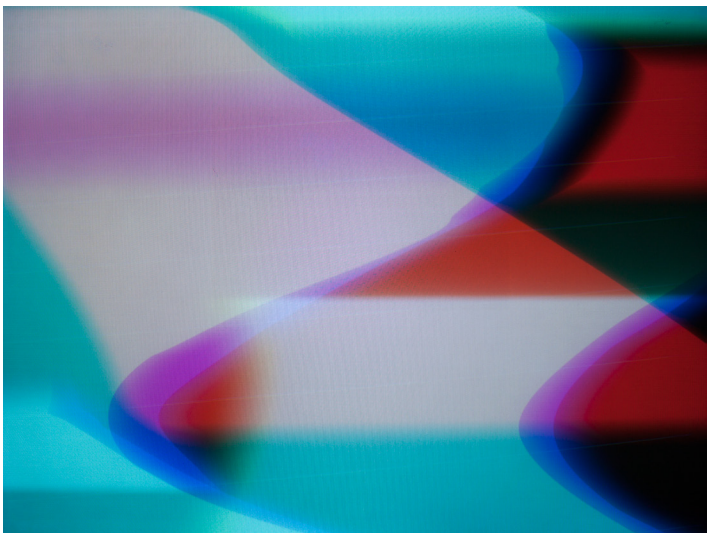

RGB.VGA.VOLT

CONTENTS









RGB.VGA.VOLT is an audio/video synthesizer that enables realtime exploration of the rich materiality concealed beneath the consumer interfaces of cathode ray tube computer monitors. By hacking and improperly rewiring the cables of these obsolete devices to short their video signals, their black-boxed analog infrastructure is liberated and driven by digitally synthesized high-frequency complex waveforms, audio playback, and feedback loops that fully exploit their latent visual spectrum. Inspired by early video tools such as the Sandin Image Processor and the Paik-Abe Synthesizer as well as contemporary programming and error-based approaches to sound and moving image, RGB.VGA.VOLT revives an analog aesthetic that has acquired a renewed power and potency in the present era of digital immateriality.

The instrument combines custom built software, hardware modification, and DIY instrumentation. This PDF includes full documentation of the RGB.VGA.VOLT software as well as the instrumentation used in interfacing the software with VGA computer monitors for realtime synthesis. It is free to be copied, repurposed, remixed, etc. “Copying is as good (I think better from this vector-view) as any way of getting ‘there.’” [1] Please cite and link to the tool in your work that utilizes it, and send your outputs my way.

James Connolly
jconno@saic.edu | jameshconnolly.com

1 Phil Morton, *Distribution Religion*.

SYSTEM COMPONENTS AND REQUIREMENTS

RGB.VGA.VOLT is a system of components rather than a singular instrument. In order to run, it *requires*:

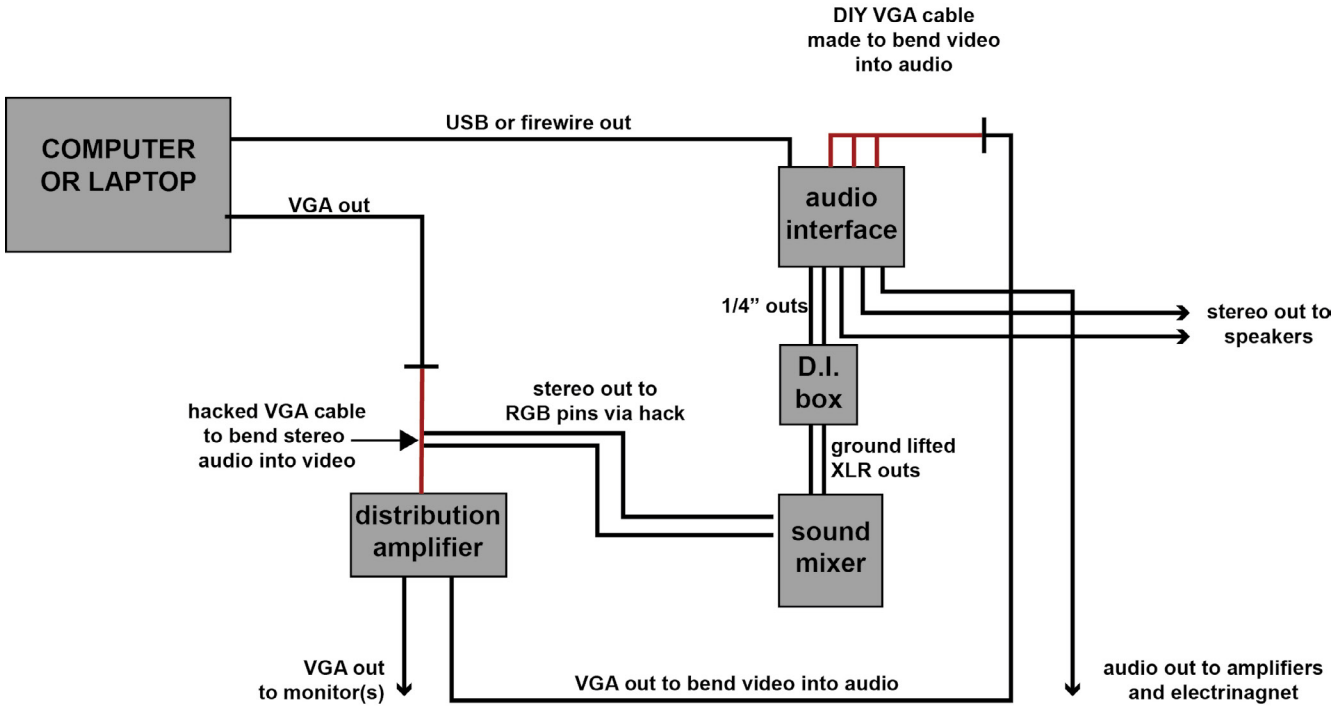
1. A mac or windows computer with a vga output and audio output (I'm hoping to support linux/ubuntu soon).
2. A DIY soldered VGA cable exension/hack to bend audio into video with a purposeful short/ground error. I've created a tutorial on building this device here.
3. A sound mixer with two XLR inputs to control the level of audio signal before it enters the red, green, and blue pins of the VGA cable.
4. A direct input (DI) box with a ground lift to enable the shorted video to display three discrete (red, green, and blue) colors. This receives the sound signal from the comuter before it enters the sound mixer and then the DIY VGA cable.

In addition to the above, the following components are optional:

5. an audio interface with at least five outputs and three inputs
6. a distribution amplifier to send the VGA signal to multiple monitors.
7. A DIY soldered VGA cable that enables video to be bent back into audio. I've created a tutorial on building this tutorial here.
8. DIY electromagnets (see Kyle Evan's tutorial at crackedraytube.com)
9. An amplifier for electromagnets (see Kyle Evan's tutorial at crackedraytube.com)

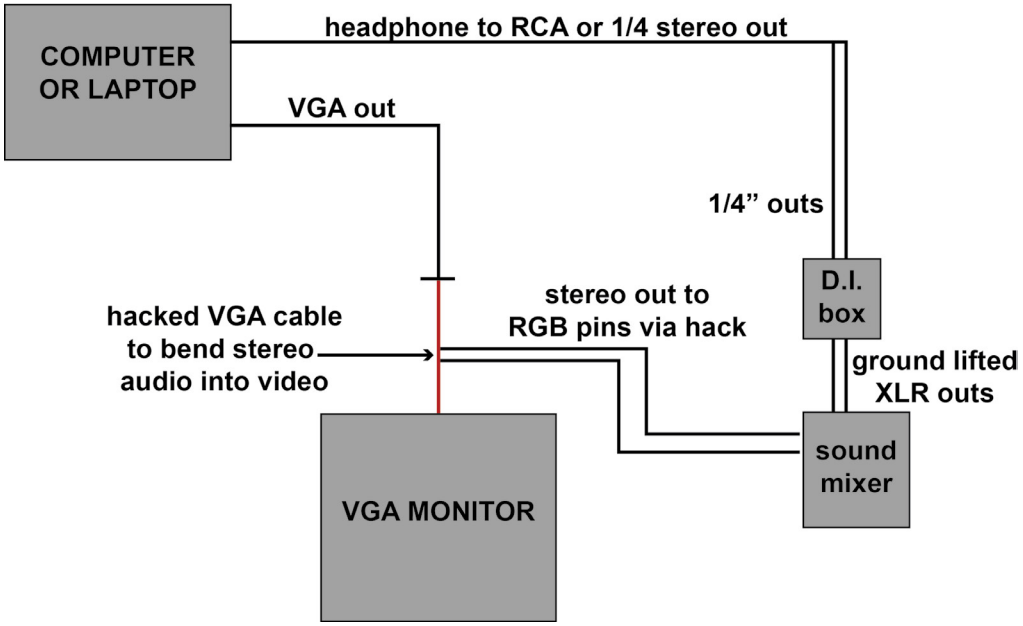
The schematic of the full system can be seen below, and a basic system schematic, running only the essential parts of the system is on the following page.

RGB.VGA.VOLT FULL SYSTEM SCHEMATIC



The above schematic is the complete RGB.VGA.VOLT system: a computer with a VGA out that connects to a distribution amplifier that then sends the signal to multiple monitors *after* the DIY VGA cable hacked to bend video into audio; the same computer is connected to an audio interface with a resolution of 192kHz that sends a stereo audio signal to the VGA hack to bend audio to video via a DI box and the sound mixer, a stereo signal to the speakers, and a mono signal to the magnets via an amplifier; the audio interface is receiving three inputs: one from each vga video channel via a DIY VGA cable that bends video into audio.

RGB.VGA.VOLT BASIC SYSTEM SCHEMATIC



The above schematic is the basic RGB.VGA.VOLT system: a computer with a VGA output that connects to one single VGA computer monitor *after* the DIY VGA cable hacked to bend video into audio; the same computer is connected to a DI box via a 1/8” stereo to double 1/4” mono cable that then connects to a sound mixer; the sound mixer outputs its signal to the 1/4” jacks of the VGA cable hacked to bend audio into video.

WIRING THE SYSTEM

Using the above schematics as a guide, set up your hardware to the specifications below.

COMPUTER/VGA OUTPUT SETUP

VGA SIGNAL SETUP: 800 x 600 resolution and 60 Hz refresh rate

This device uses your computer's VGA output in order to stabilize the video signal and enable easy video playback. The settings of this outputted signal directly impact the synthesized visuals. The presets loaded into RGB.VGA.VOLT enable high-frequency synthesis to occur in moments where the visualized waveforms are most interested and can generate the widest range of outputs. **The presets function properly only when being bent into a VGA signal with a refresh rate of 60 Hz and resolution of 800 x 600**, and need to be properly tuned from one computer to another using the internal tuner of the software.

In your settings, unmirror your displays so that the VGA output acts as a secondary display. On a mac, this is done in the Displays section of your System Settings. This will enable you to control the patch while the monitor displays black or a video/image file.

AUDIO INTERFACE SETUP

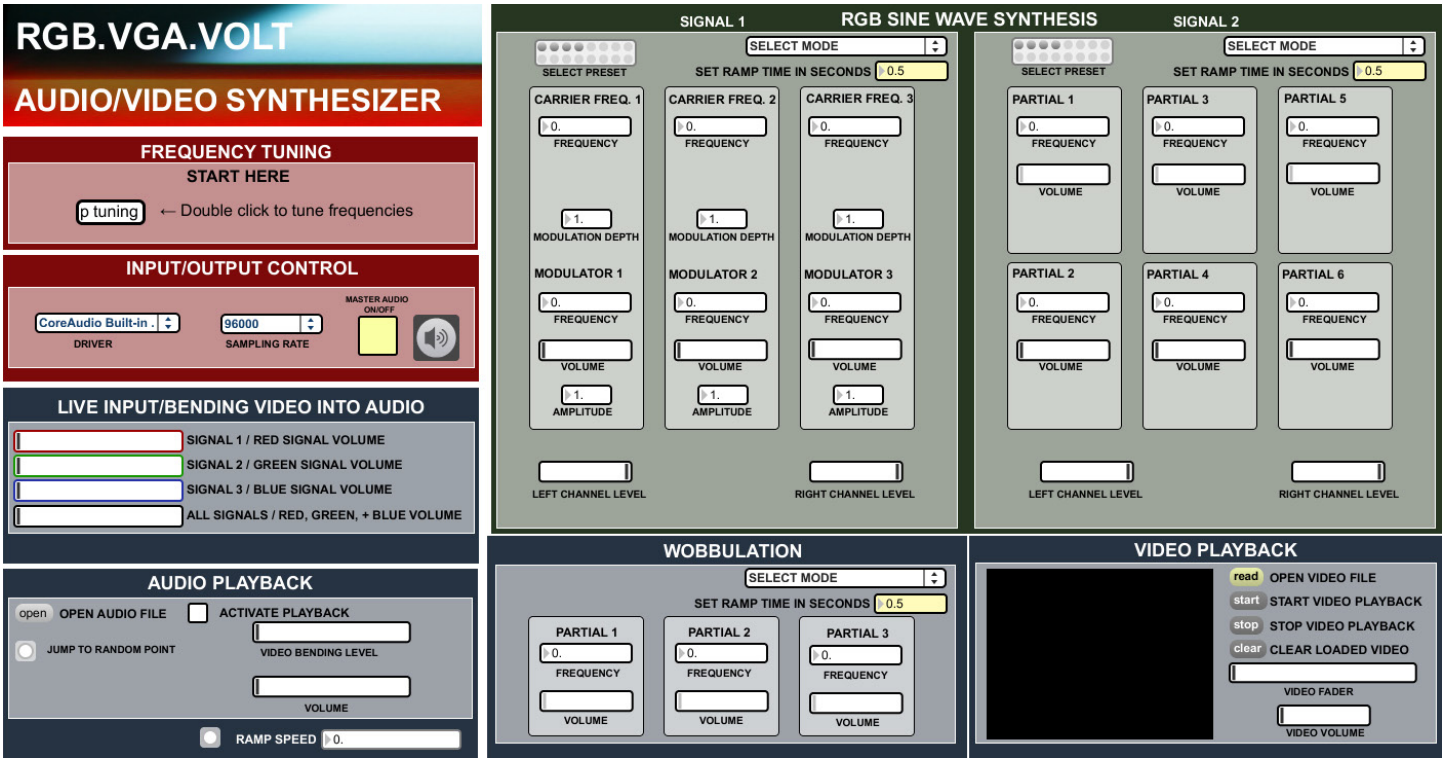
USE HIGHEST AUDIO RESOLUTION POSSIBLE

The hack requires high-frequency

HACKED VGA CABLE SETUP: LIFTING THE GROUND, PANNING AUDIO, CONTROLLING SIGNALLEVEL

The hacked VGA cable uses an improper wiring method that I stumbled upon while experimenting with VGA monitors in 2010 (see an example of those early experiments here). The wiring enables discrete red, green, and blue signals from one single stereo output (left and right channels) by grounding one of the three video pins and attaching the other two to the positive signals (e.g. attaching the red pin of the VGA cable to the ground of one of the two stereo signals, the blue pin to the positive of the left signal, and the green pin to the positive of the right signal). This improper wiring—grounding a signal—means that the red visualizations are lost if the hacked VGA cable makes contact with a properly grounded device such as a sound mixer, an audio interface, and a number of other things. To get around this issue, and enable the

RUNNING THE SOFTWARE



The RGB.VGA.VOLT software is an application built in the Max environment that is intended to be used with an audio interface that has at least 5 outputs and 3 inputs, but it can be used in a less powerful way with your computer's standard stereo input and output. The software requires DIY instrumentation to properly function (a VGA cable hacked to bend audio into video, another to bend video into audio, and optional DIY electromagnets to wobbulate the imagery), as outlined in the tutorials below. It generates complex waveforms through additive synthesis, frequency modulation, amplitude modulation, or any mixture of the three using specific high frequencies that, when bent into video, generate particularly powerful visualizations. The software also enables the user to display a video file on their VGA monitor, bend any AIFF or WAV formatted audio file into video, sonify video by bending it back into audio, and generate low-frequencies to "wobbulate" the CRT imagery if interfaced with a DIY electromagnet built using Kyle Evans' tutorial.

THE INTERFACE

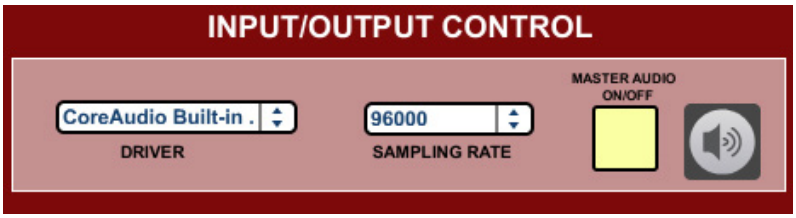
The RGB.VGA.VOLT interface is divided into seven sections, organized into three color categories:
Red sections at the top left set up control of your VGA output including refresh rate and screen size, and the audio input and output control including your driver and sampling rate.
The green section at the top right controls the sine wave synthesis that is driven into the red, green, and blue pins of the VGA cable.
Blue sections consist of additional audio and video tools that give the user the ability to bend video back into audio, to playback audio files and output them into the RGB pins of the VGA cable, to playback video fiiles, and a "wobbulator" section to send low-frequency sine waves to electromagnets (if you built them and used them to modify your CRTs).

These sections are outlined on the next page.

RED: TUNING AND SETUP

SECTION 1: INPUT/OUTPUT CONTROL

SELECTING THE DRIVER, SAMPLING RATE, AND TURNING THE MASTER AUDIO ON AND OFF



DRIVER

The driver drop-down menu, automatically populated by available audio input and output hardware, selects the device through which the software will send outputted and receive inputted signals. Open the drop-down menu and select your audio driver. If you don't have an audio interface use your computer's internal output. If your audio interface doesn't show up, ensure that your interface is on, close RGB.VGA.VOLT and restart it.

SAMPLING RATE

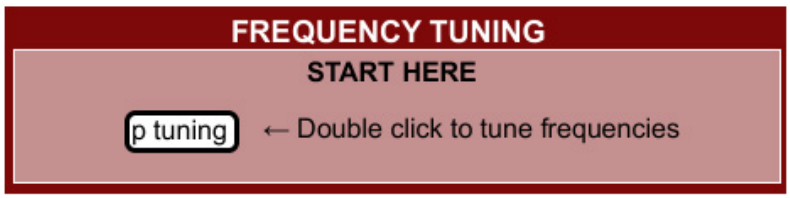
The sampling rate drop-down menu, automatically populated with your driver's available sampling rates, automatically selects the highest resolution possible. RGB.VGA.VOLT uses presets An audio interface is only able to generate frequencies that are HALF its sampling rate or below, so if your sampling rate is 44kHz, you'll only be able to generate a frequency of 22kHz, lower than all three RGB.VGA.VOLT presets at roughtly 38,000 Hz, 57,000 Hz, and 76,000 Hz, meaning that a sampling rate of at least 152,000 Hz would be ideal. Work within your system to enable this software to function proplery. If your highest sampling rate is only 88,000 Hz, ony use the 38,000 level preset. This still enables plenty of complex visuals.

MASTER ON/OFF

The on/off toggle enables and disables the master volume for all of RGB.VGA.VOLT's functions. When the toggle is turned on and the box contains an X, audio is on; when the X is not present, audio is turned off. The main function of this button is to enable the audio to fade in and out, ensuring that a large pop doesn't occur if sounds are suddenly turned off

SECTION 2: TUNING

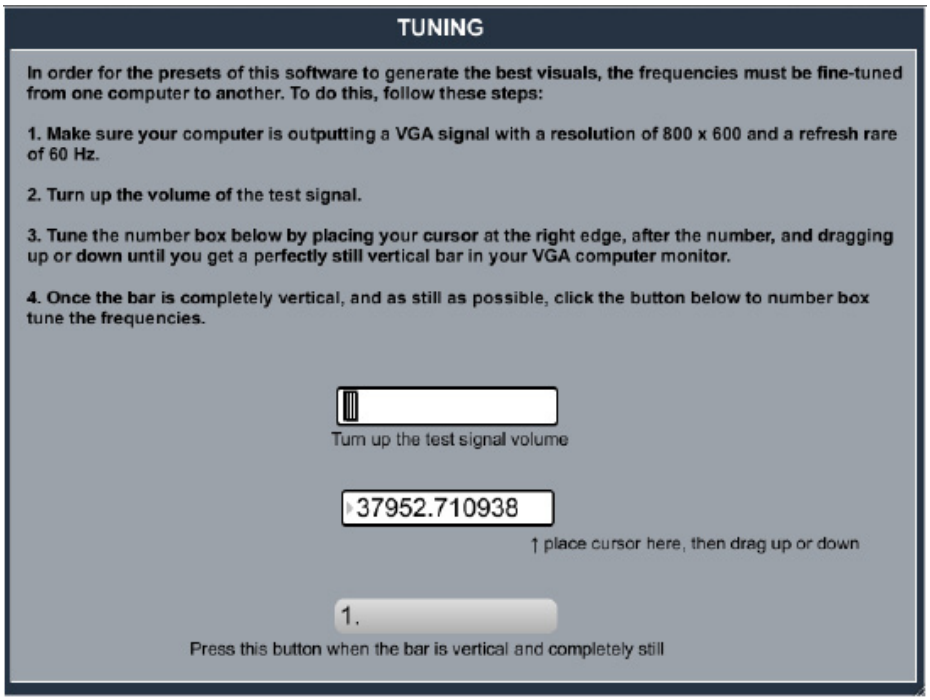
FREQUENCY TUNING



The high-frequency complex waveforms generated in RGB.VGA.VOLT have presets that generate specific visuals that are particularly compelling--moments where the spontaneous visualizations of the signal turn from thin horizontal lines to vibrant and wide vertical bars. For whatever reason, when run on different computers these ideal frequencies shift and don't generate the proper visuals as the system I

built this software on. Because of this, the frequencies being generated by RGB.VGA.VOLT need to be tuned by a multiplier that corrects the difference.

To tune your system, double click the rounded rectangle labelled “p tuning.” This opens a new window:



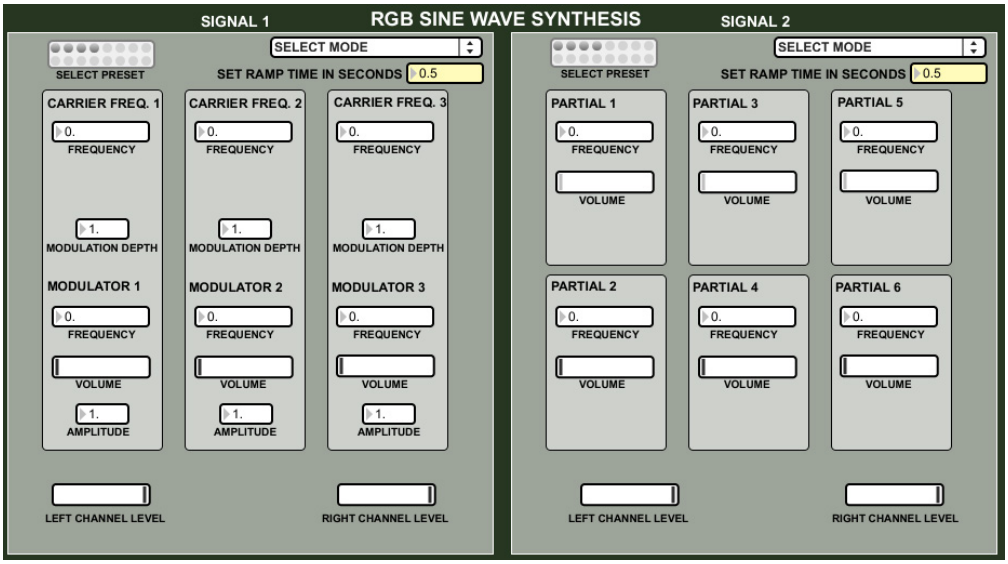
The tuning subpatch has an oscillator generating a high-frequency sine wave that, in order for the system to function properly, will generate a perfectly still vertical bar as it’s bent into a VGA computer monitor. Turn your volume up Most likely when you turn set your system up this will not be the case. In order to tune your system, place your cursor to the right of all of the integers, where the up-pointing arrow is, and drag it up or down. Keep doing this until the visualizations turn into a vertical bar, as perfectly still as possible.

Once the visuals are a perfectly still (or as close to being perfectly still as possible), press the gray button at the bottom of the patch. This is the ratio of the original frequency to your system’s frequency, and it sets the system up to multuply all generated frequencies by this amount. Once you’ve generated a vertical bar and pressed the gray button the system is tuned.

You’ll need to tune your system every time you restart the software.

GREEN: SINE WAVE SYNTHESIS

The most important function of RGB.VGA.VOLT is its sine wave synthesis, which generates complex waveforms through additive synthesis, amplitude modulation, frequency modulation, or a combination of two of these methods. These synthesized signals are then connected to the RGB pins of the VGA cable, where they generate three discrete video outputs.



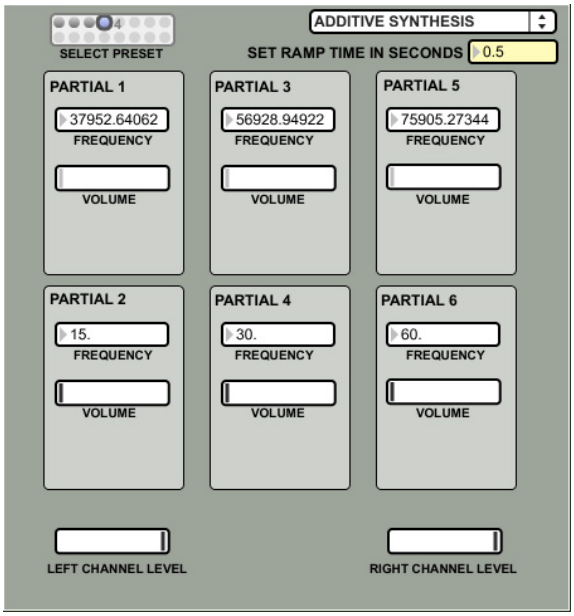
The drop-down menu at the top right selects which synthesis mode you're using. No signal will be generated if nothing is selected.

The sine wave synthesis section is built so that two separately functioning signals are produced and then mixed into one stereo signal that gets bent into video via the DIY VGA cabled hacked to bend audio into video. Signal 1 and signal 2, as they're labeled, *both* go to the left and right output to the VGA cable. At the bottom are sliders that set the level of audio for the left channel and the right channel for both signals. Moving these allows the user to change the color pins that the signal is being sent to. If your VGA cable has red grounded, blue attached to the left output, and green attached to the right output, turning both channels all the way up creates two discrete colors: red, and teal (blue and green mixed). If, for example, the left channel is turned down in that scenario, you will have two discrete colors: red and green (with blue absent). Using both Signal 1 and Signal 2 allows you to generate three discrete colors. With the same scenario above, Signal 1 with the Left Channel turned all the way down, and Signal 2 withe the right channel turned all the way down enables three discrete colors: red, green, and blue (so long as different waveforms are being generated

Ramp time, at the top right of both signal sections, changes the amount of time (in secconds) that it takes for a change to occur within that section. For example, if ramp time was 5, and you changed a frequency from one number to another, it will create a smooth transition from the first number to the second over 5 seconds. Ramp time controls the time over which transitions occur between all elements of each signal: volume, channel volume, frequencies, and all elements of both frequency modulation and amplitude modulation.

The different modes of synthesis are described below.

ADDITIVE SYNTHESIS



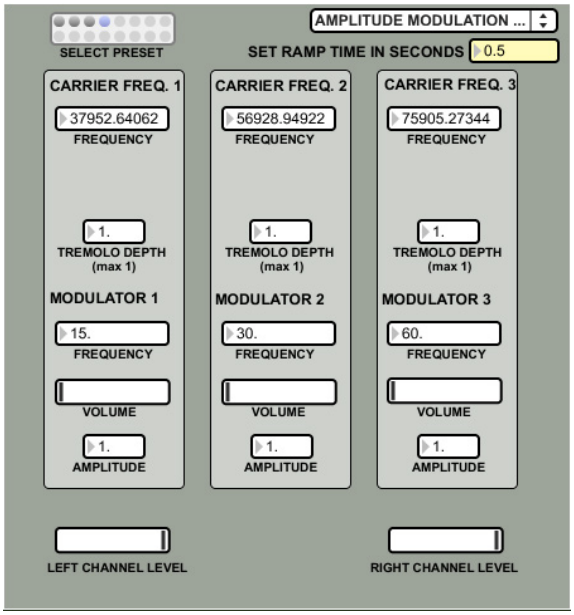
The first mode of sine wave synthesis is Additive Synthesis: synthesis of multiple basic waveforms (6 if using one signal, or 12 if using both signals). Additive synthesis generates a *composite* tone through multiple *partials*,

The composite tone generated when multiple sine waves of different frequencies are added together audibly (and visually, of course, when that signal is bent into video) perceivably contains the *sum* and *difference* frequencies of those added signals. When two sine waves that are close in frequency are added together, such as 57.5 and 62.5, we perceive their difference frequency (the frequency between the two partials, in this case 60Hz) rhythmically beating or pulsing as well as their sum frequency. This is why we can hear a complex waveform generated by sine waves oscillating at a rate both below and above our hearing range. The addition of even more sine waves greatly adds to the complexity of the waveform, resulting in a wide range of possible visual outputs within VGA computer monitors.

The presets within RGB.VGA.VOLT are organized so that the top row focuses on high-frequency vertical bars, while the bottom row focuses on low-frequencies harmonized with the computer monitor's refresh rate. Additive synthesis is most effective in RGB.VGA.VOLT when combining these two frequency types, and exploring the areas surrounding the exact preset values. Mixing all three vertical-bar frequencies generate engaging visuals, as does exploring the visuals generated from very close frequencies, generate rhythmic beats by making small changes to two of the three pre-set frequencies on the top row.

The final way to control these signals is through their volume: mixing levels and enabling one partial to overpower others.

AMPLITUDE MODULATION

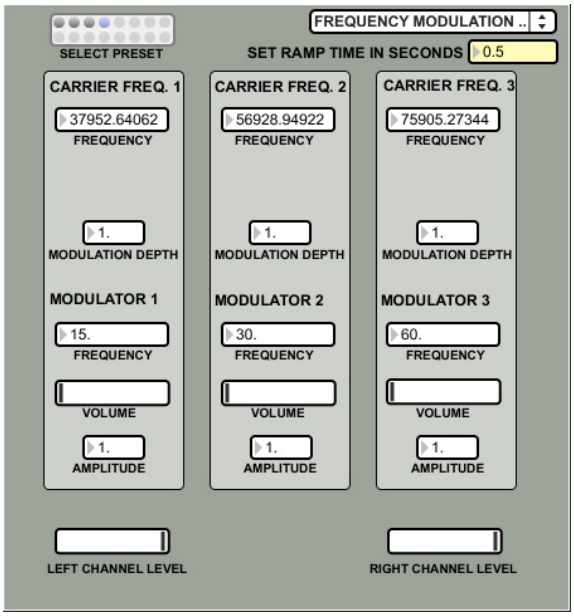


The second mode of sine wave synthesis is Amplitude Modulation: varying the amplitude (or strength) of a *carrier* frequency with a *modulating* frequency that is altered by a *tremolo depth*.

Amplitude Modulation mode organizes the sine wave synthesis sections into three rows of discrete signals generated through amplitude modulation. These three signals then interact with one another through additive synthesis, just as the simple sine waves do in the Additive Synthesis mode described above.

Each amplitude-modulated signal is organized into 5 components:
1. A *carrier frequency*, the base frequency that will then have its amplitude modulated. These are tuned to be high-frequency vertical bars by the presets.
2. A *modulator frequency*, the frequency that fluctuates the carrier frequency's amplitude. These are tuned to low frequencies that are harmonics of the monitor's refresh rate. Sub-audio modulating frequencies are particularly good at generating tremolo (sonic wavering) effects.
3. The *tremolo depth*, a value between 0 and 1 that further controls the modulated amplitude and further enables the wavering of the audio signal.
4. The amplitude of the signal post-amplitude modulation.
5. The volume of the signal, added to enable the user to further control each signal's balance when additively synthesized with other generated signals.

FREQUENCY MODULATION



The third mode of sine wave synthesis is Frequency Modulation: modulating the frequency of a *carrier* frequency with a *modulating* frequency that is altered by a *modulation depth*. Frequency modulation produces tones equal to the carrier frequency plus and minus the modulating frequency.

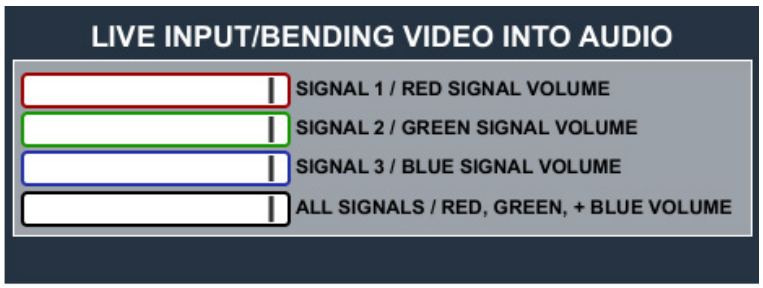
Frequency Modulation mode organizes the sine wave synthesis sections into three rows of discrete signals generated through amplitude modulation. These three signals then interact with one another through additive synthesis, just as the simple sine waves do in the Additive Synthesis mode and the three signals do in Amplitude Modulation, described above.

Each amplitude-modulated signal is organized into 5 components:
1. A *carrier frequency*, the base frequency that willbe modulated by another frequency.These are tuned to be high-frequency vertical bars by the prests.
2. A *modulator frequency*, the frequency that modulates the carrier frequency. These are tuned to be harmonics of the monitors' refresh rate by the presets.
3. The *modulation depth*, a value that changes the amplitude of the modulating frequency. This is preset to 1, but can be raised much higher for the best effect (the 250-500 range is particularly good)
4. The amplitude of the signal post-frequency-modulation.
5. The volume of the signal, added to enable the user to further control each signal's balance when additively synthesized with other generated signals.

Experimenting with RGB.VGA.VOLT using both signals and mixing a range of sine wave synthesis techniques can generate endless outputs. Try methods on their own and synthesized together. Use the presets and explore frequencies on your own.

BLUE: ADDITIONAL A/V TOOLS

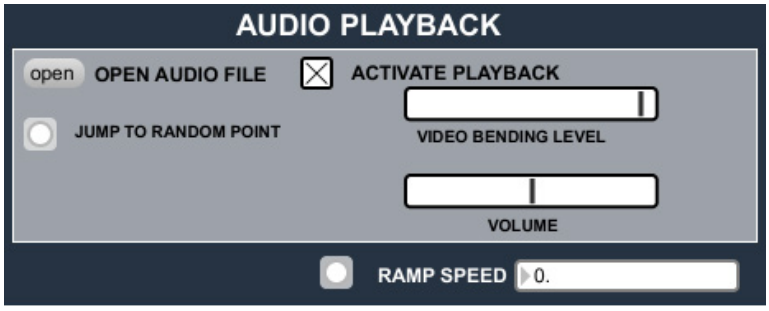
LIVE INPUT/BENDING AUDIO INTO VIDEO



The LIVE INPUT/BENDING VIDEO INTO AUDIO secion of RGB.VGA.VOLT enables the red, green, and blue video signals of a DIY VGA cable built to bend video into audio to be sent to the speakers of your system via outputs 3 and 4. This enables synchronized audio and video including video or audio iles played through the video playback section (described below).

The four volume sliders control the red, green, and blue audio levels discretely via the labelled top 3, and all three together via the slider at the bottom.

AUDIO PLAYBACK

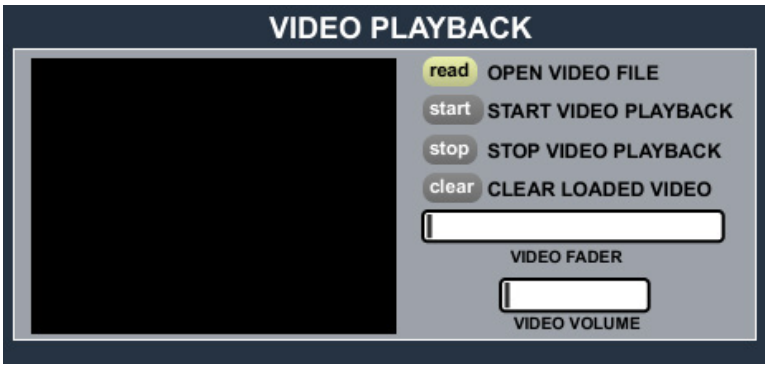


The audio playback section enables an audio file to be bent into video via outputs 1 and 2 (and the VGA cable modified to bend audio into video). It also enables the same audio file to be sent to the speakers via outputs 3 and 4.

Press the “open” button to open a finder window and select your audio file (use AIFF or WAVE file formats only). Press the “ACTIVATE PLAYBACK” toggle to enable playback, and press the “JUMP TO A RANDOM POINT” button to randomly jump within the timeline of the audio file. The random function is mainly for abstract audio files that are sent into the monitors to generate chaotic visualizations (such as feedback signals).

Audio playback levels to both the VGA video and the speakers can be controlled through their respective toggles. Ramp speed ensables both of those toggles changes to occur over a number of specified seconds.

VIDEO PLAYBACK



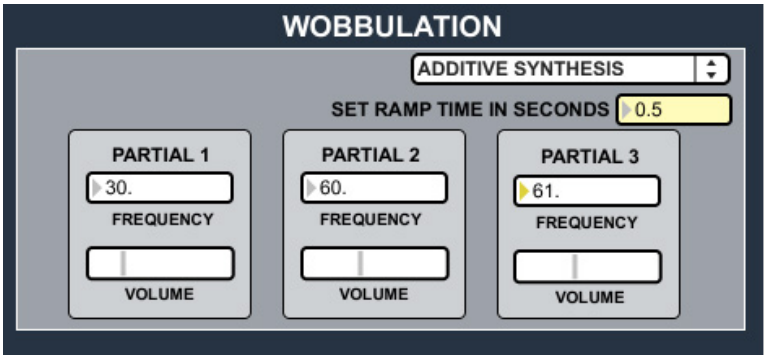
When you open RGB.VGA.VOLT, a black video window automatically appears. The video playback section enables video or image files to be displayed on the VGA computer monitor(s). Unmirror your displays to enable the VGA video to be displayed while your main monitor still displayes the RGB.VGA. VOLT software. Drag the video window into the screen area that is displayed on your VGA computer monitors and pres ESC to fullscreen it.

The video window can display .mov or .jpg files. Press the “read” button to open a finder window and select your file. Press the “start” button to enable playback, the “stop” button to stop playback, and the “clear” button to clear the loaded video and return to a black screen.

The slider bars at in the VIDEO PLAYBACK section enable you to fade the video in and out (all the way to the left is faded to black, and control the video’s volume (which is sent to outputs 3 and 4/the speakers). Usually you don’t have to turn the volume up much for it to be at a reasonable level.

Note that video files directly impact the bending of video back into audio.

WOBBULATION



The wobbulation section is meant to be interfaced with a DIY electromagnet through an audio amplifier, using Kyle Evans’ tutorial on hacking a cathode ray tube with a DIY electromagnet at crackedraytube.com. Kyle’s instrumentation is a based on the Paik-Abe synthesizer, also known as the wobbulator.

RGB.VGA.VOLT has only one magnet output, but that can be multiplied through a hard splice (and intensified in the amplifier), or through a distribution amplifier or sound mixer.

The wobbulation section functions through additive synthesis alone, as electromagnets wobbulate imagery best through simple oscillations from one polarity to another.

Note: Electromagnets work best in the low frequency range (30 Hz - 120 Hz), particularly around 60 Hz. Because the VGA signal is set up at a refresh rate of 60 Hz, a 60 Hz signal sent into the magnet will mean that the wobbulation will be synchronized with the refresh rate, creating a jagged still image. Exploring the space around 60 Hz and harmonics of 60 Hz can be particularly interesting.

Cite myself and RGB.VGA.VOLT in work deriving from this software.

Email questions, suggestions, thoughts, and comments to jconno@saic.edu.

jameshconnolly.com
crackedraytube.com

James Connolly