Building Digital Sound Healing Tools in Pure Data

Scott Yeager
yeagersm@gmail.com
June 18, 2019

There is no question that acoustic instruments, and especially the human voice, are the most powerful tools for healing with sound. Computers and synthesizers, on the other hand, offer unparalleled flexibility and precision in generating and analyzing sounds for the purpose of supporting sound healing practice. For example, a tuner app can help to identify the name of a note that was intuitively discovered and expressed vocally, or calculations for shifting the frequency of planetary movements into an audible octave may be performed in a spreadsheet. Despite the great potential for building powerful custom digital tools to aid in the exploration and understanding of acoustic phenomenon for healing and the expansion of consciousness, few such applications currently exist.

Developing custom audio software is no simple task, especially when support for different operating systems and devices is a priority. An ideal solution for prototyping digital sound healing tools would offer the flexibility of writing code from scratch, while also offering the convenience of graphical interfaces and easy cross platform deployment. Pure Data provides a happy medium in these regards, being a visual environment where objects are patched together to specify a flow of audio and other data. It provides a collection of building blocks for synthesizing, analyzing, and processing audio, which can be freely supplemented with extensions written in code. Patches created in Pure Data can be run on all major operating systems and even mobile devices, just by installing the appropriate free, open source software package or app.

The first experiment that a sound healing student might want to try in Pure Data is generating sine tones at specific frequencies. This can be accomplished with just two boxes: one that generates the tone and another that represents the computer's audio output. As seen in diagram 1, the first box reads `osc~ 432`, meaning a sine wave oscillator at 432 hertz, and the second box reads `dac~`, which stands for "digital to analog converter" (the computer's soundcard or audio interface). There are two lines connecting the boxes, specifying that the signal flowing out of the oscillator should be connected to both the left and right channels of the audio output of the computer. If everything is properly configured, the tone will be heard on speakers or headphones as soon as the connection is made. Without any interactive controls, this patch is a bit boring and not terribly useful, but it serves to demonstrate how simple it can be to get started with Pure Data.

Diagram 1

To expand our tone generator, we might like to add a way to set the frequency and volume, or include a second oscillator for playing with harmonies and binaural beats. It's easy to patch in additional boxes for more functionality, or copy and paste entire sections, which can also be made into new boxes called "abstractions" that hide the details inside. Diagram 2 shows a tone generator with slider controls and a display of the closest note to the chosen frequency (in equal tempered tuning with a concert pitch of 440hz, though this can be changed). The graphical interface is separated from the computational details of the patch shown in the other window, and it is scaled to fit on a smartphone screen, for use with an app that runs Pure Data patches on mobile devices. This is now a complete, if still basic, single tone generator with a functional interface.

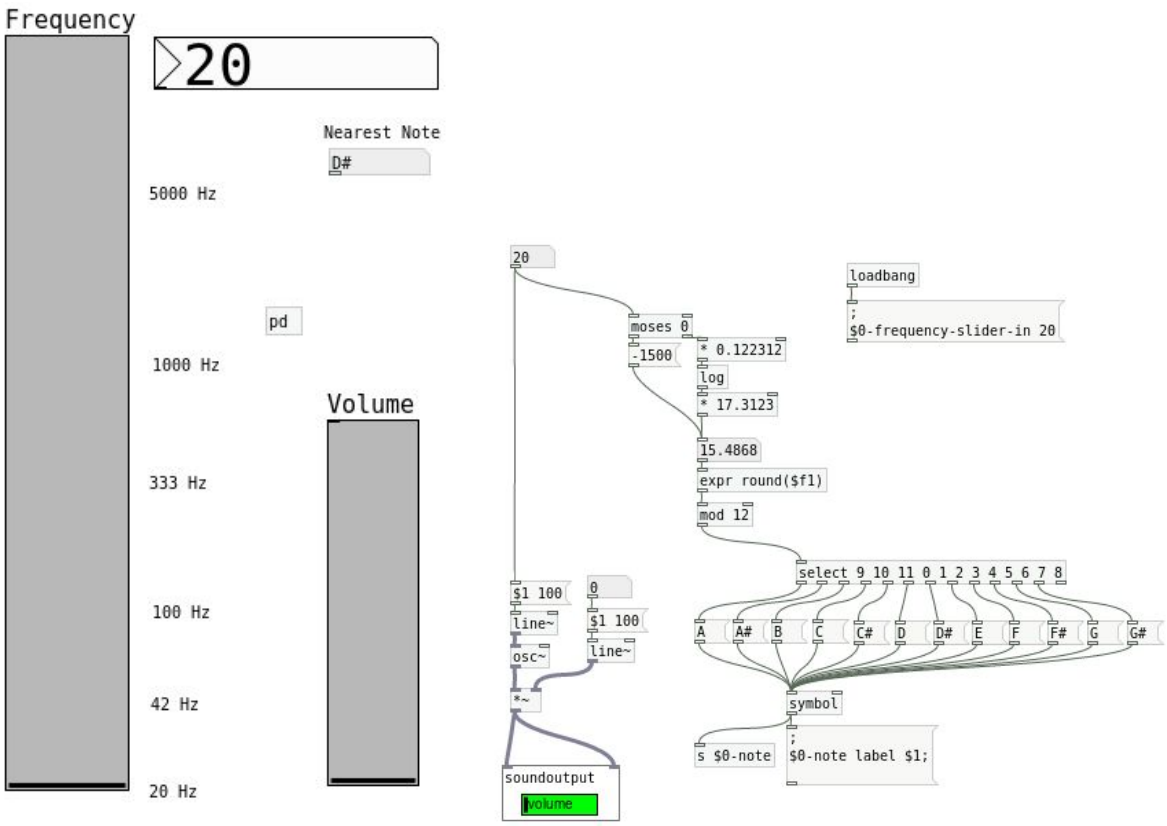

Diagram 2 - Tone Generator Interface and Patch

So far, we haven't accomplished much that's outside the capability of widely available and free software applications. The power of Pure Data really shines when its mathematical capabilities are coupled with its audio engine. For example, we might be interested in synthesizing binaural beats with both the carrier and beat frequencies tuned to octaves of a particular note. By encoding the necessary calculations within the patch, it is possible to determine the appropriate frequencies and set the oscillators to produce them, in real time according to user input. Diagram 3 shows a patch that does this. Likewise, someone interested in experimenting with microtonality and alternative tuning systems, for example, can create their mathematical specifications and hear the resulting pitches all in one place.

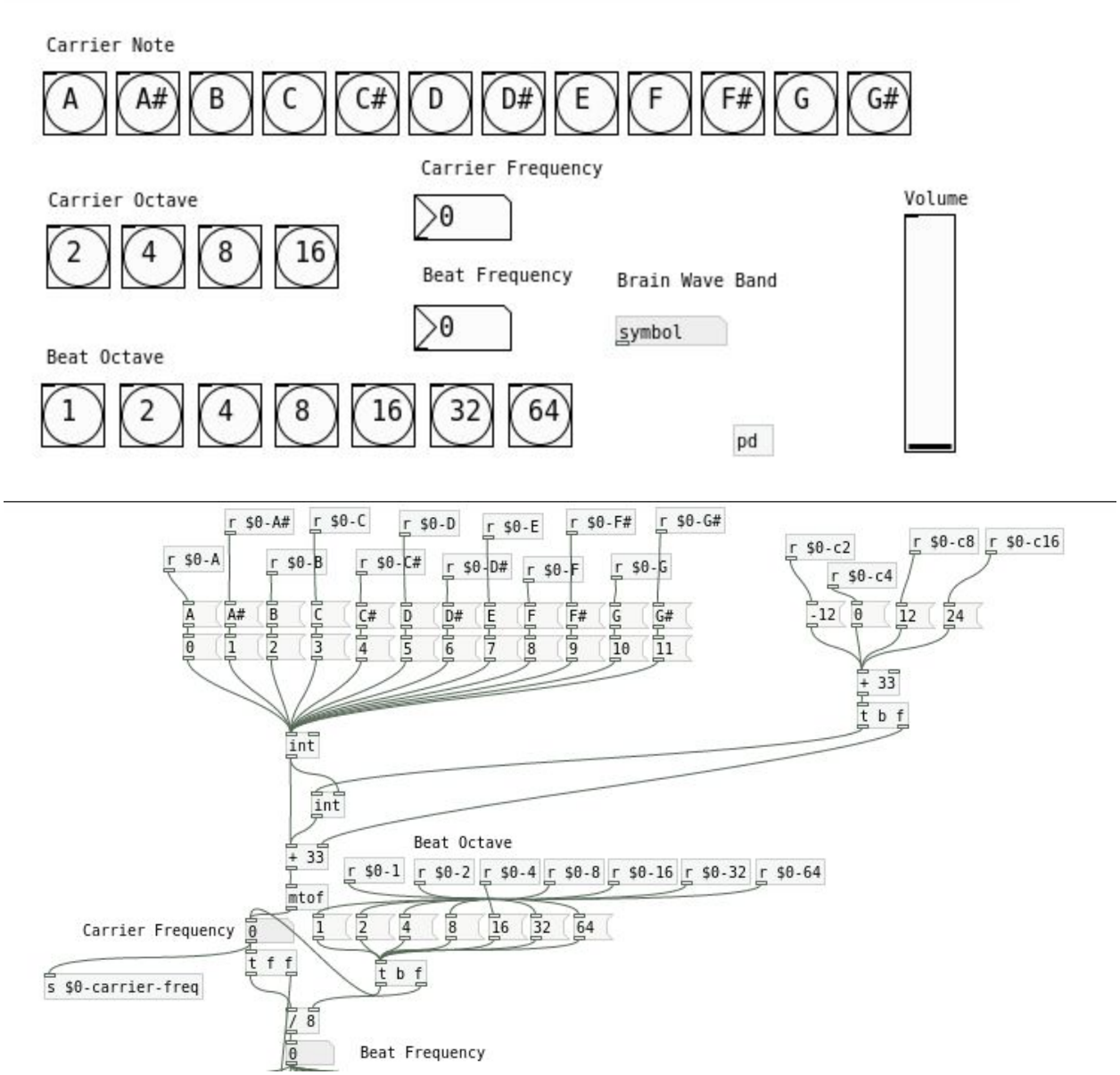

Diagram 3 - Binaural Beats (partial patch shown)

Synthesizing sine waves is just scratching the surface of what Pure Data is capable of. Beyond more complex synthesis, it also has facilities for analyzing input in real time and working with recorded audio. It is relatively simple to build a tuner that identifies the frequency of a tone picked up by a microphone. With a bit more work, a patch can separate individual notes and keep track of them, for the purposes of analyzing music or spoken word. In diagram 4, such a patch is shown, with sliders graphing out the density of each note logged. While this prototype is fully functional, it could still be much better tuned to isolate individual tones, especially in speaking voices. However, the relative ease with which such applications can be created and deployed across speaks greatly to the potential of the platform.

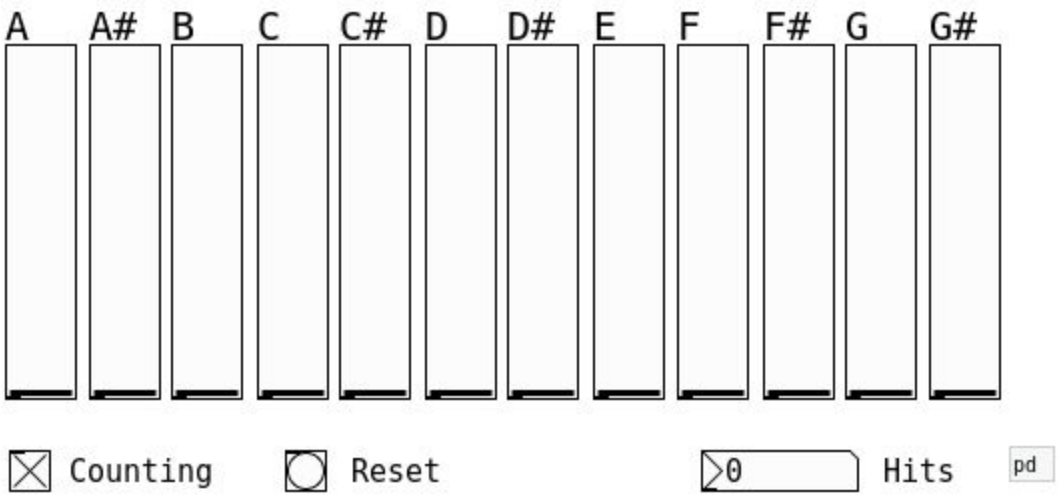

Diagram 4 - Note Counter Interface

Pure Data is not without limitations, of course. While the patching environment is more convenient than coding in some ways, it can also become tedious when creating and patching many objects together. The graphical controls are easy to include but can't be customized much beyond size and color. Running patches requires installing software and manipulating files, which can be a hurdle for some users. In fact, many patches are deployed inside of other applications just for handling audio, like the sound effects in a game. This is made possible and relatively simple by additional software contributed from the community to the Pure Data ecosystem. That means prototypes created in Pure Data can be fleshed out into native applications for computers and phones that have nice looking interfaces and install with a click, without recreating the inner workings of the patch.

One might imagine an app that includes a library of frequencies and their associated uses in sound healing, perhaps with an interactive component to rate effectiveness and add new entries, optionally shared online. A user could input their home note, then hear binaural beats (with synchronized visual entrainment), drones, and even generative music in that key. Voluntary data submissions could constitute a valuable addition to the study of sound healing. Learning tools might interactively demonstrate the relationships between harmony and sacred geometry with responsive audio. Writing the remaining code to implement these ideas on top of Pure Data prototypes is a much more manageable task than starting from scratch. In the meantime, patches can always be run on computers, or smartphones using the PdDroidParty and PdParty apps.

This has been a brief introduction to the potential for building digital tools for sound healing in Pure Data. For anyone curious to learn more and begin experimenting for themselves, many excellent tutorials and resources are available online. Visit the Pure Data website to get started.