Developing a Virtual Modular Synthesizer for Sound Waves and MIDI

Margaret Jagger   Advised by: Dr. Drew Guarnera and Dr. Sofia Visa

Introduction

Within the world of digital signal processing (DSP), digital audio, and physical instruments, the products which result from explorations between computer science and music are dependent on the creator's familiarity with both fields. Current applications for music synthesis are sufficient in usability, with a user able to begin altering an input's sound to their liking, provided they have a MIDI controller on hand to serve as the input device [1]. Otherwise, a MIDI sequencer may be required for modular synthesizers which only accept MIDI.

Figure 1. **The Minimoog Model D. An influential modular synthesizer, this is the first commercial modular synthesizer which contains several fundamental synthesizer concepts found today.**

Objectives

The primary goal is to create a virtual modular synthesizer with two options for inputs: MIDI and pure sound waveforms (pure tones), such as sine waves. Then, this virtual modular synthesizer could be used as a demonstrative tool on how one could combine pure sound waves and MIDI into one synthesizer.

Additional, secondary objectives include:

1. Synthesizing (creating) sound waves within *SuperCollider*
2. Accepting MIDI as an input option
3. Altering both input options, resulting in an obvious sound modification

Modules and Functionality

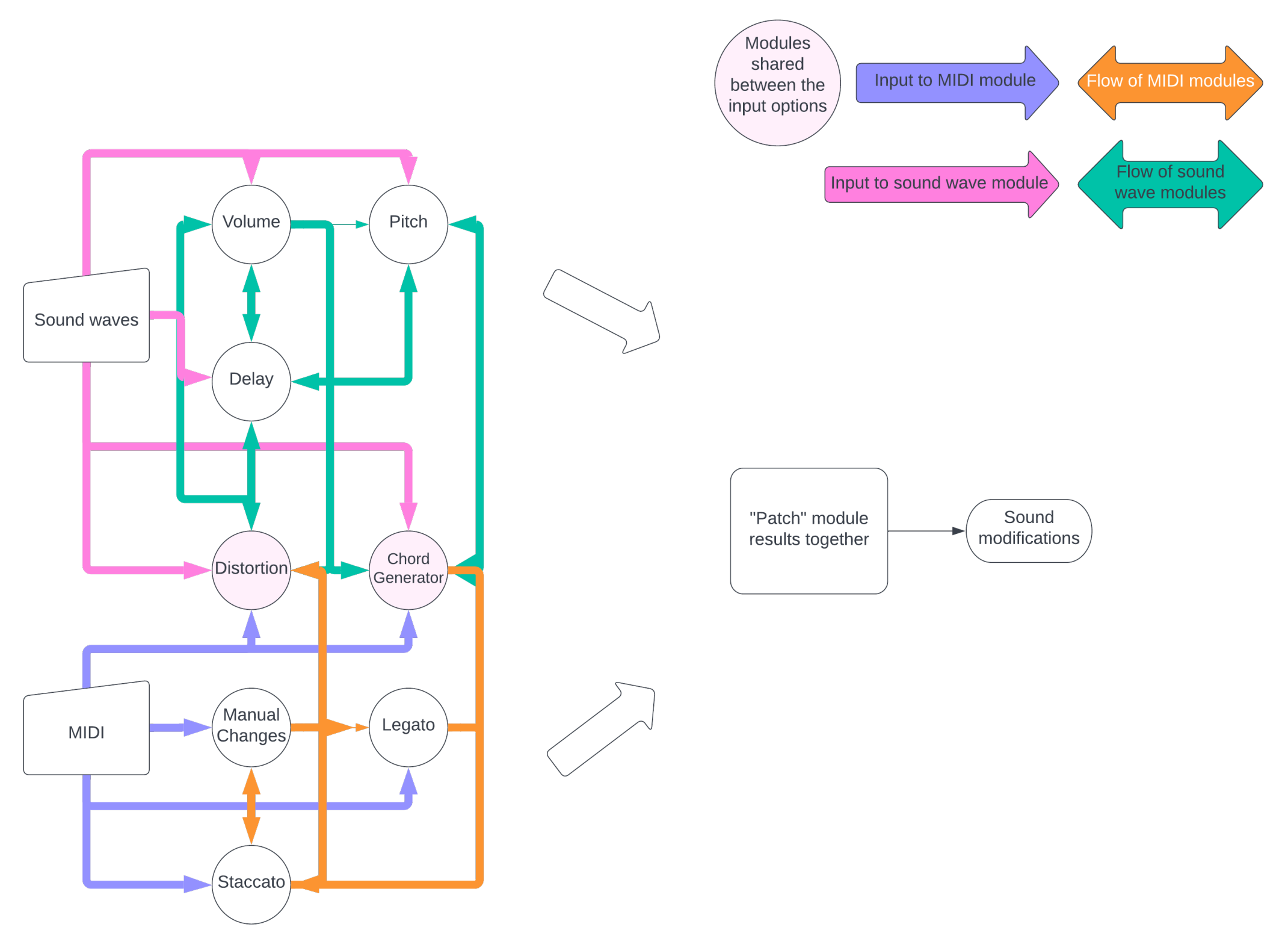

Figure 2. **The order of modules and sound modifications within this virtual modular synthesizer. It depicts how there is no linear flow or set order to sound modifications. Instead, the order is user-dependent.**

Two input options: sound waves and MIDI

Sound wave modules: volume, pitch, delay, distortion, chord generator

MIDI modules: distortion, chord generator, legato, staccato, changes to ADSR envelope

Modular Synthesis

Modular synthesis: the alteration of sound waves, such as the one in Figure 3

Modular synthesizer: an electronic instrument (synthesizer) composed of separate modules for different functions

Modules will be "patched" together to create a resulting modified sound, connected in pseudo-linear way → modules themselves are not programmed in a set order, as "patch" will determine the order modules are applied to digital signal, based on the order user uses the modules.

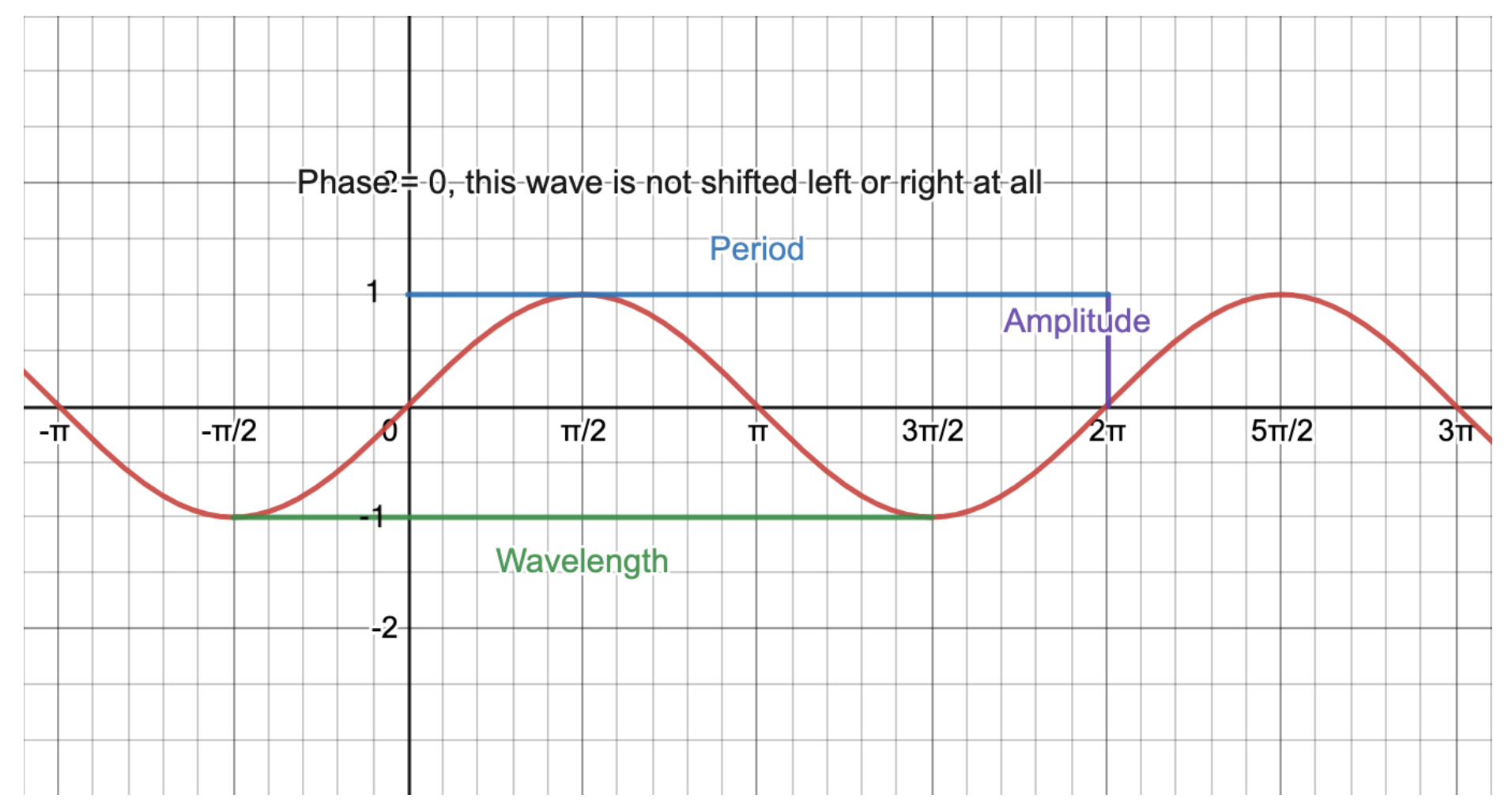

Figure 3. **Parameters of a unit sine wave explained. In this figure, the amplitude is 1, as this is the maximum vertical displacement of the wave from the x-axis. The frequency is determined by the wavelength, and has a value of 1. The phase is unchanged and has value 0.**

Modifications to a periodic sinusoidal wave are done through a combination of 3 parameters or adding/subtracting waves from an overall composite sound

- Amplitude: determines the volume of sound. From Figure 3, amplitude is the maximum vertical displacement of the wave's crest (either the peak, the highest point, or the trough, the lowest point) from its equilibrium point of 0. The larger this distance from 0 on the x-axis, the larger the volume.
- Frequency: determines the wave's pitch, or how high or low we perceive sound to be. This is determined by the number of vibrations (or the value of the wavelength) of the wave within a second. A smaller wavelength = higher pitch. Examples of various frequencies are seen in Figure 4 [2].
  - Wavelength: distance between repetitions of the wave's crests
- Phase: determines the timing of the wave, if it's perceived to be early, delayed, or on-time. This depends on the wave's period, a delay module typically affects this.
  - Period: the time *T* of a wave to move through one full oscillation (one peak and one trough)

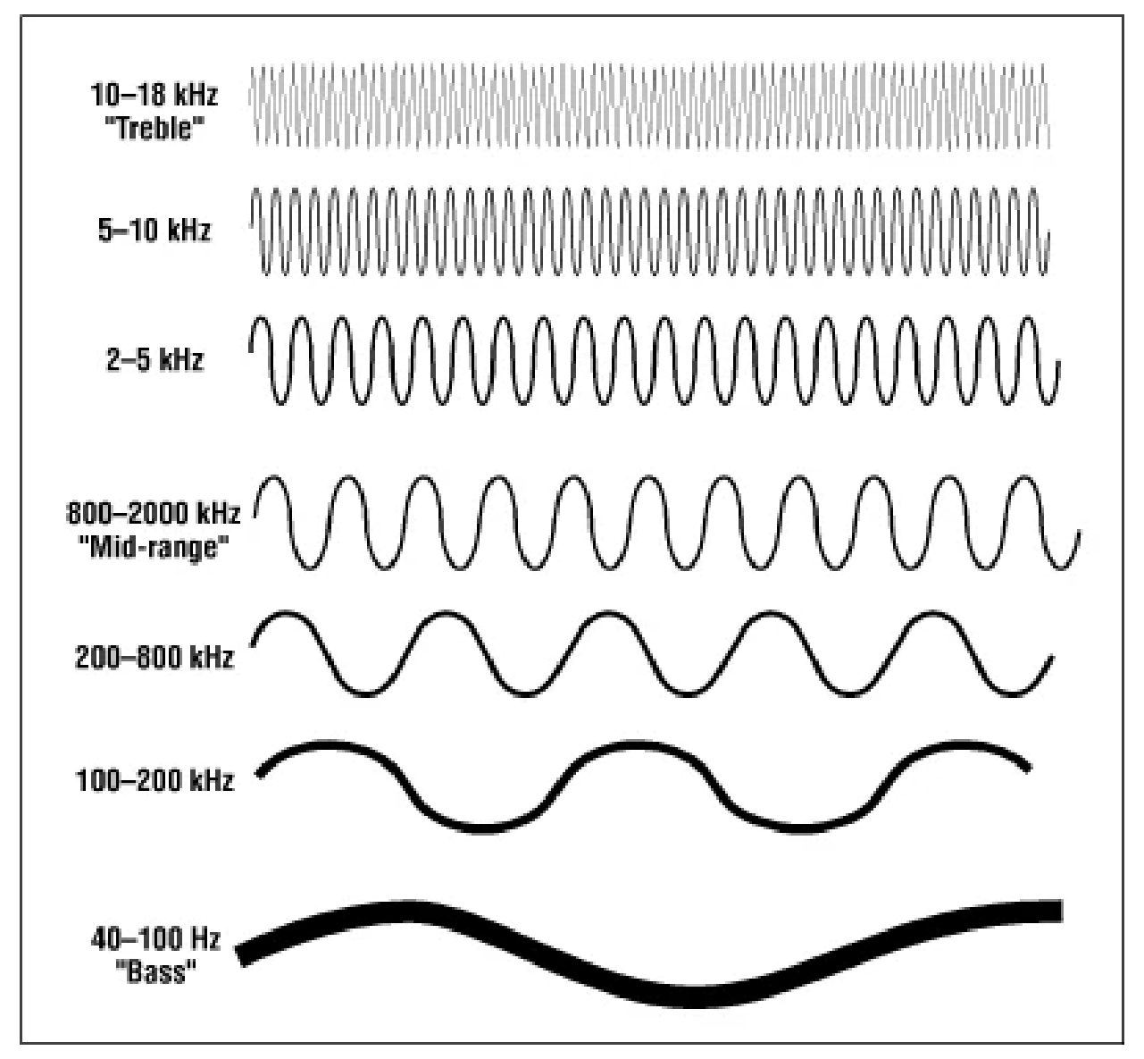

Figure 4. **The audible range of frequencies within average human hearing, visualized as periodic sinusoidal waves.**

MIDI and its Limitations

MIDI (Musical Instrument Digital Interface): digital communications protocol allowing for multiple hardware, software devices to communicate

The primary role of MIDI is to translate the performance "events" of a MIDI controller into its equivalent digital message, then transmit the messages to other MIDI-compatible devices

Two MIDI messages we focus on: Note on and Note off, which define the beginning and ending of MIDI notes

Several limitations to MIDI include:

- No sound support: the MIDI protocol cannot communicate audio itself, instead it will use audio connections to instruct a device compatible with audio to manage sound (refer to Figure)
- MIDI is built around the keyboard/keyboard notes: for other types of MIDI instruments, there may be restrictions to how notes may sound and how individual characteristics are portrayed (e.g. Clarinet vs Keyboard; Acoustic Guitar vs Keyboard)

Final Design

Due to the continuous nature of pure sound waves, such as the sine wave, modules built for pure tones will be different from modules built for MIDI.

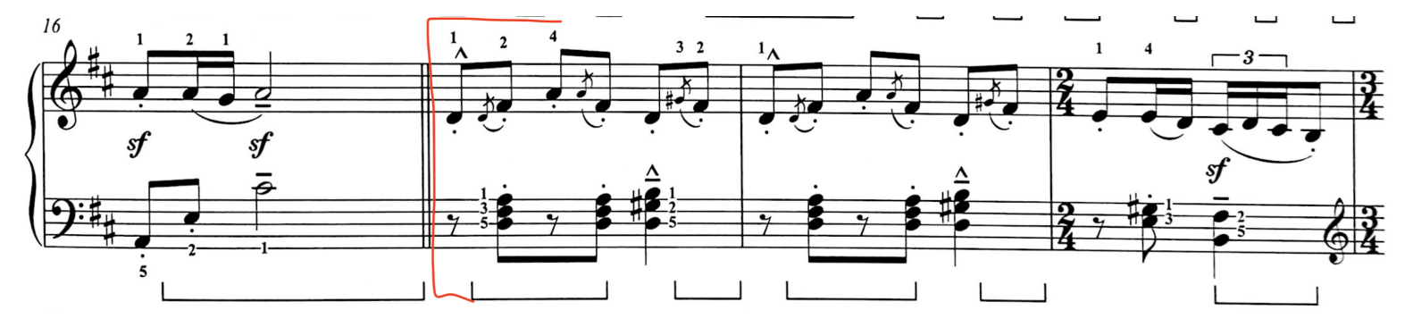

Figure 5. **An example of staccato in classical music, Béla Bartók, Romanian Folk Dances, Poarga Românească, mm. 16-19**

Figure 6. **An example of legato in classical music, Béla Bartók, Six Romanian Folk Dances, Buciumeana, mm. 13-15**

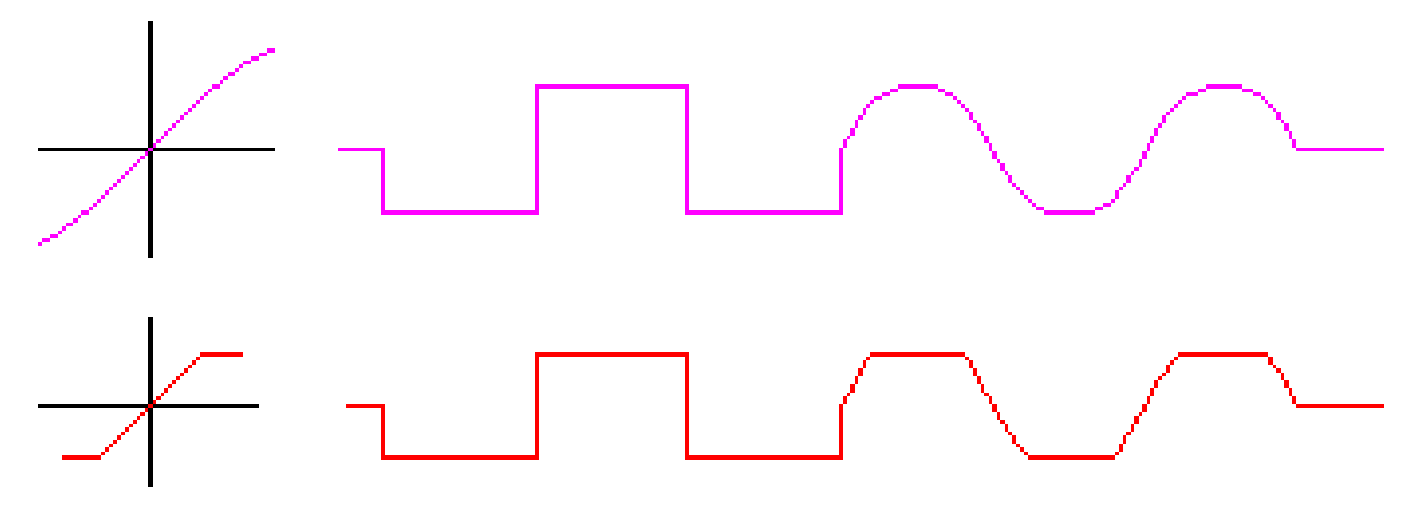

Figure 7. **An example of distortion. Normal waves are shown on the left, and an example of distortion applied to each wave on the right.**

Conclusions and Challenges

The primary goal of creating a virtual modular synthesizer which accepted both pure tones and MIDI as valid input options was achieved. The secondary objectives were also met, resulting in a fully functional prototype virtual modular synthesizer.

The most noticeable challenge: an inability to always invoke principles of good software development. For variables to be accessible outside its immediate scope of where it is first initialized, it must be environmental → we lose the ability to separate functionality as each module must maintain its modularity and ability to function in any order and affect the output of another module.

Future Work

Future development on this type of project could include, but is not limited to, the following changes: additional modules and functionality for both pure tones and MIDI, adding compatibility for MIDI and MP3 files (such as the MIDI file in Figure 8), integrating the two input options more fluidly, and altering the GUI for easier use outside the *SuperCollider environment*.

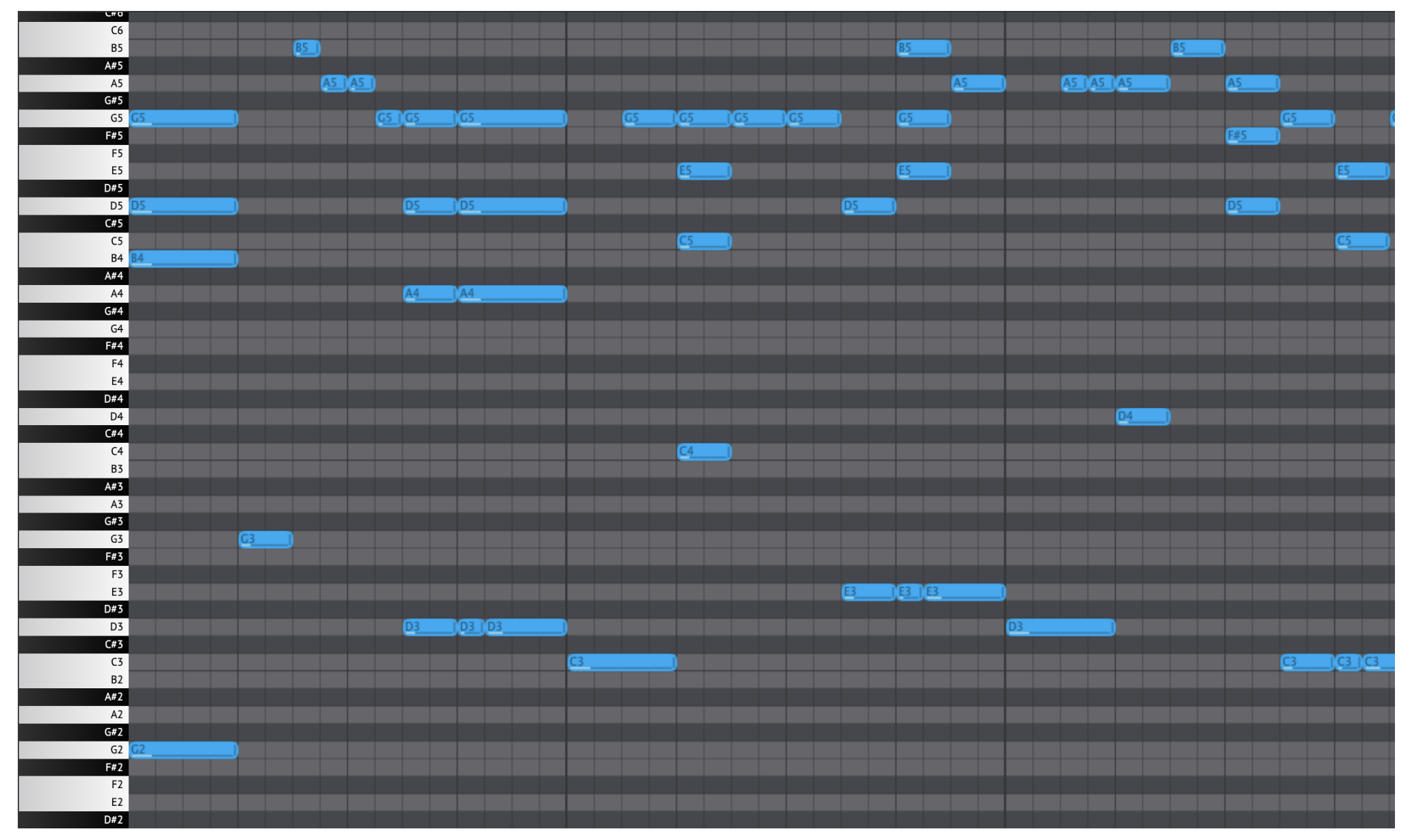

Figure 8. **An example MIDI file**

References

[1] Krash. *Minimoog synthesizer*. Dec 2005.

[2] Digital Audio Wiki. Sine wave.