ARIZONA STATE UNIVERSITY

DR. OLIVER BECKSTEIN

PHY 494: Assignment 7:

Digital Signa1 Processor in Python

Charles E Fortune III

April, 2016

Abstract

This is a proposal written for the final project of PHY 494: Computational Methods in Physics at Arizona State University. The proposal is meant to generate enough interest in Digital Signal Processing that a three-person team may be formed around said proposal. More specifically, the end-goal of the project is to create a code or set of codes in the Python language that are able to generate and manipulate audio signals, known colloquially as a synthesiser.

PROBLEM

Since the dawn of civilisation, music has been an integral part of society on virtually all levels; every culture has created some means of expression through crafting of sound, be it for cultural appeal or even as a defining traditional statement. However, in the modern 21^{st} century world, music has taken on a place on the global stage as an evolving and ever-expanding scene. As technology rapidly swells, so too will the tools necessary to make new and exciting sounds a reality.

The question is, when and where will these new sounds come from? With so many options for musicians, the possibility for expansion is huge, most notably when it comes to digital sound. Unlike analogue sound which relies on tangible interaction that is restricted to often bulky physical apparitions, digital sound is so flexible and broad that a huge array of options are at the disposal of the composer. For the purposes of this proposal, (and indeed to narrow the scope of the assignment), the main problem is to digitally create audible sound by using the Python coding-language to generate waveforms, and further manipulate these waveforms to create unique synthetic sounds.

APPROACH

The idea is basic then: to create a Python code that can generate and manipulate digital waveform(s), known colloquially as a synthesiser, or more technically as a Digital Signal Processor (DSP). The resulting audio may then be broadcasted through some sort of speaker system.

To generate waveforms, a simple periodic function may be generated using a simple algorithm based on the wave-equation; another approach would be to generate sinusoidal, triangular, square, or even non-periodic waves, which is a matter of what function is defined in Python. Otherwise, a manual input through means of .mp3 or .wav audio file may be used and manipulated from there. By using a manual input of an already completed song, it bypasses the need for a function generator completely.

As with any DSP system, the foremost task is to manipulate waveforms in a meaningful way. The Python Module *ThinkDSP* will be invaluable for this as both a reference and an outright tool. Types of manipulations include but aren't limited to: Fourier transforms of all kinds, function scaling, shear mapping, partial transforms, etc.

OBJECTIVES

1. Running code able to generate numerous varying continuous functions
2. Running code can take an input function $[x]$, apply a response $[h]$, and generate output $[y]$
3. Running code able to plot and display both the input function $[x]$ AND resultant output $[y]$ as a visual waveform
4. Running code that is able to write to a playable audio file
5. **[Stretch Goal]** GUI utilising Pygame for real-time manipulation
6. **[Stretch Goal]** Analogue interface that can produce audio-signals as an independent source (hopefully using Rasberry Pi/Arduino hardware).