Assigr Design Document

m Rowland

09/04/2020

1. Overview

For part 2 of assignment 1, I decided to write a program that simulates the sound of a human male “oo” vowel. It uses lookup tables to generate sine waves and combines them into one waveform using additive synthesis. I have also outlined some short-falls of my approach and possible improvements.

2. Motivation

As a singer, I ha[illegible] the physical properties of the human voice. In particular, the physics b[illegible] interests me. Vowels are characterised by a set of three strengthened harmonics [illegible] goal was to combine these formants to create a convincing human sound.

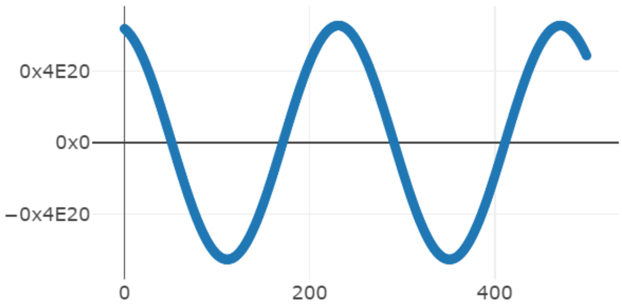
3. Wave Generation

3.1. Lookup Tables

The lookup tables are stored in SRAM and contain values for one period of four sine waves with frequencies 200Hz (fundamental), 300Hz (formant 1), 870Hz (formant 2) and 2240Hz (formant 3). They can be found in the file `lookup_tables.S`. Because the audio module has a 48kHz sample rate, the lookup tables have `round(48000/f)` entries where `f` is the frequency of the corresponding wave. I chose to use full period lookup tables to simplify computation. It is possible to use smaller quarter-period tables and extrapolate using symmetry of the sine function, but the memory space saved is insignificant compared to the available memory.

3.2. Sine Wave Generation

Sine waves are perfect for voice simulation since they have no harmonics themselves. The formants can be generated cleanly. There are three generated waves with frequencies 200Hz, 300Hz, 870Hz and 2240Hz, each with an amplitude of `0x7fff` (32767). The three higher frequencies are the first, second and third formants for a male “oo” vowel I chose the fundamental pitch to be lower than the first formant but still easily audible. The waves are combined using additive synthesis.



Segment of a 200 Hz sine wave with range `0x8000-0x7fff`

3.3. Additive Synthesis

The sine waves are combined using additive synthesis. I chose this method for its simplicity. The formula used to combine them in each sample is

```
69/100*fund + 1/4*f1 + 1/10*f2 + 1/100*f3
```

where `fund` is the funda[illegible]t 1, f2 is formant 2 and f3 is formant 3. Since all the waves have amplitude [illegible]g factors add up to 1, the resulting waveform is guaranteed to lie in the [illegible]`f`. The weights were chosen through trial and error and may not agree with the [illegible]voice. I believe a more accurate vocal simulation could be achieved by altering the scaling factors.

4. Improvements

The end result is not a convincing vowel sound. The pitches do not meld together as I would have liked. A more suitable synthesis method would yield better results, or perhaps better scaling for additive synthesis would also improve the sound. Adding more frequencies in the harmonic series of the fundamental could also help.

The current register-heavy implementation does not allow synthesis of large numbers of waves since the lookup table offset for each wave is stored in its own register. This is not a problem for this project since there are only four tables, however adding further waveforms would be impossible. A more memory-dependent implementation would be needed for large numbers of waves.

In situations when space efficiency is more important that time efficiency, the lookup tables should be reduced to quarter-periods. The other three quarter-periods can be extrapolated using the symmetry of the sine function. This is not an issue with the current program since there is a lot of available memory space and computation time, but it could become a problem for more complicated extensions of the idea.