DSP JOCKEY

Roles:
Brian Bourn (bab2177): Product Manager
Vanshil Shah (vs2409): Language Guru
Abhinav Mishra (anm2147): System Architect
Addisu Petros (aep2157): Test/Validation

Description:
Our proposed language will be a signal processing language that engineers and designers can use to build signal-processing algorithms. This has many use cases in Digital Signal Processing, DJ/Audio Mixing, Telecommunications, and more. The main primitive types would essentially be int, float and signal. These parameters can be used to specify key values such as time, frequency, amplitude, phase, etc. which are fundamental in representing a given signal. Using these primitive types, one would be able to programmatically add, manipulate, and create new waveforms from some given base wave.
The main idea behind our language is to provide a simple framework that would enable interested parties to write programs that can conveniently manipulate signals. The language will provide simple data types that would make this task manageable. As such, programmers will be able to import and create signals with relative ease. At the same time, we will provide operations that will be used to support the essential modifications a programmer will want to make to sound files. Overall, by the end of our project, it is our belief that we can create an environment, which will allow for straightforward representation and manipulation of signals.

Proposed Uses:
Since this language will be able to manipulate signals it opens up a variety of different possibilities. With the ability to modify signals, the language will be able to support writing any DSP function such as Fourier Transforms, basic phase shifting, amplitude/frequency modulation, etc. One possible end user application could be an Electronic music generator. Digital Signal processing is even relevant when it comes to the financial stock market where there are many DSP applications used in essential data modeling and market analysis. Therefore, as signal processing is ubiquitous, this language has many applications that can be used to create programs for a myriad of industries.

Syntax:

1. Comments:
For comments we will be using C-style comments. Thus, whitespace and comments are ignored. // means a comment for the whole one line and /* test */ means all of the text is treated as one overall comment.

2. Primitives + Operations:

Integer	32 bit integer value
Float	Modeled after floating point numbers in C based on IEEE 754
Boolean	True/False values
Signal	Represent a vector of integers or floating point values
=	Assignment operator
+	Integer/float addition as well as signal concatenation operator
-	Integer/float subtraction
*	Integer/float multiplication
/	Integer/float division
+=	Add then assign to the left value. In the case of signals, concatenate to the signal on the LHS
^	Power operator to raise an integer to a power. Only integers accepted as RHS values
>, <, <=, >=, ==, !=	Comparator operators for the primitive types
signal[ ]	Signal access operator indexed by time

3. Variable Declaration:

Examples:

```
int i = 0; //same syntax as C
float f = 0; //same syntax as C
Signal signal = createWave(sine, 0, 0, 60, 10);
```

4. Control Flow:

a. **if-else:** Same behavior as C

Example:

```
if(boolean){
//execute this code block
}
else {
//else execute this code block
}
```

b. **For-loop:** Same behavior as in C

Example:

```
int i;
for(i = 0; i<10; i++){
```

```
//execute this statement ten times
}
```

c. While-loop: Same behavior as in C

Example:

```
while(boolean){
//execute this statement
}
```

Functions:

Create Wave	Takes a type and parameters and return a wave that represents that signal. These types will include sine, cosine, impulse, unit step, and ramp.
Phase Shift	Phase shifts a signal by an amount specified by an input parameter
Time Shift	Time shifts a signal by an amount specified by an input parameter
Amplitude Multiplication	Multiplies the amplitude of a signal by an amount specified by an input parameter
Time Scaling	speeds up or slows down a signal
Time Reversal	reverses a signal

I/O:

Import - Allows a User to import a premade signal into the program from a .wav file

Export - Writes a signal from a program to a .wav file

--

Example Program:

Write a Frequency step Program

```
amplerate = 44100;          // Every program requires a sample rate
freq = 100;
Signal final_signal;        // Declaration of the signal to return

for(int i = 0; i < 8; i++) {
        Signal signal = create_wave(sine, 0, 0, freq, 10, 2);
        /*createWave(signal type, Float time delay, Float phase shift, Float frequency,
Float Amplitude, Float length in seconds)*/

        final_signal += signal;     //Append the newly created signal to the final signal
        freq += 100;                //Increase the Frequency
}
export (final_signal,"My_signal.wav");      // Creates the final .wav file
```

--