Assignment 2: Sonifying Pong!

I. Matlab

A helpful tutorial:
http://www.math.utah.edu/lab/ms/matlab/matlab.html

Additive Synthesis: Synthesizing sounds from a bunch of sinusoids added together.

Here is a fundamental, important truth about audio signals (or any signal, for that matter): Any waveform can be broken down into sinusoids (specifically, *complex* sinusoids- but we can talk about complex sinusoids at another point). Sinusoids are the most rudimentary building block of a signal. There are three parameters to a sinusoid: *Amplitude, Frequency, and Phase.* We're going to use our sineTone function from lab 1 to build a bunch of different, crazy sounds (ok, I'll admit that most of these sounds are pretty boring, but it'll still be cool to see how they are made).

A.
First off, we are going to add an *initial phase* parameter to our sineTone function:

```
output = sineTone(frequency, phase, duration, fs)
```

Check out this wikipedia page for phase, or ask me if you have any question about what it is. The definition of a sinusoid that has an *initial phase* component is this:

$$y(t) = A\sin(2\pi f t + \varphi) = A\sin(\omega t + \varphi)$$

Source: https://en.wikipedia.org/wiki/Sine_wave
Where A is amplitude, f is frequency in Hertz (cycles per second), and φ is *initial phase*. Little omega (ω) is defined as frequency in *radians*.

B.

Using your new sineTone function, add together a sinusoid with a sinusoid with the same frequency, duration, and sampling rate, but with a phase of π:

```
y = sineTones(440,1,0,48000) + sineTones(440,1,pi,48000);
```

Try listening to the resulting signal, and plotting it. Write a brief description of what you see/hear in a comment in your Matlab script. *Hint:* your plot may look strange, but look at the actual scale of the y axis. How big are the actual numbers in the signal? Look up the term *roundoff error*.

Look up *constructive* and *destructive* interference, and let me know if you have any questions about what these terms actually mean.

C.

Create three functions- sawtoothTone, squareTone, and triangleTone:

```
output = sawtoothTone(frequency, phase, duration,
harmonics, fs)

output = squareTone(frequency, phase, duration, harmonics,
fs)

output = triangleTone(frequency, phase, duration,
harmonics, fs)
```

Frequency is going to be the *fundamental frequency* of each tone, and harmonics is how many sineTones we will add together. Each of these functions will return a signal that is the sum of multiple sineTones with specific frequencies that are multiples of the *fundamental frequency* given.

The equation for a sawtoothTone is the following:

$$y(t) = \frac{1}{2} - \frac{1}{\pi}\sum_{k=1}^{K}\frac{sin(2\pi kft+\phi)}{k}$$

Where *f* is your fundamental frequency, *k* is the number of the harmonic, and *K* is the number of harmonics we are rendering. Φ still represents *initial phase*.

The equation for a squareTone is the following:

$$y(t) = \frac{4}{\pi}\sum_{k=1}^{K}\frac{sin(2\pi(2k-1)ft+\phi)}{(2k-1)}$$

And the equation for the trangleTone is the following:

$$y(t) = \frac{8}{\pi^2}\sum_{k=0}^{K}(-1)^k\frac{sin(2\pi(2k+1)ft)}{(2k+1)^2}$$

Try plotting these three different tone types with, different numbers in the `harmonics` parameter. When you increase the number of harmonics, what does the waveform start to look like for each of these three tones? Answer in a comment in your script.

D.

Make a function called fmTone:

`Output = fmTone(fc, index, rate, duration, phase, fs)`
Where fc is the *center frequency*, index is the *modulation index*, rate is the *modulation rate*, and duration, phase, and fs are the same parameters we've passed into our other tone functions.

This function will produce a signal that uses *frequency modulation*: Inside of our sine wave, the frequency is being controlled by *another sine wave.* This is where I would make a joke about the movie *Inception* if it wasn't 2016.

The function for an FM tone is as follows:

$y(t) = sin(2\pi f_c t + (A_m sin(2\pi f_m t)))$
Where f_c is the center frequency, f_m is the modulation rate, and A_m is the modulation index.

Try changing all these numbers with wanton abandon. It's worth noting that the *center frequency* will be the closest thing you have to a fundamental- so putting 440 there will result in a tone that is around 440 Hz.

That's it! If you've done all of these, you now have a really robust additive synthesis library. Keep in mind that you can stack any of these signals together by adding their samples together (+), or make a signal by playing multiple signals in sequence by using `[signalA, signalB]`. You also still have those rampOn and rampOff functions from the first lab, which you can use on any signal. Also remember that you can use phase to create interesting constructive and destructive interference patterns in a sound!

E-mail me the following:

- A copy of all the functions you write here
- A plot of 1000 samples of each of these functions (`output(1:1000)`). In the title of your plot, please write the function name, as well as the parameters you passed to that function to make the resulting signal.

III. Sonifying Pong

Take Pong as an inspiration for your first video game. In a group of 3-4 we'd like you to design, program, and test a new version of pong with a focus on sound!

You can do anything you want with the idea of Pong! You could add more players, add more mechanics, add new artwork, make sound spatialized, use the frequency spectrum as a game mechanic, or create something you could play with your eyes closed! Go crazy! **Please remember to focus on sound design and think about the how the sounds you are using can give cues to the player.**

You should create your main sounds in matlab. As we've been doing, you can write your tones to a wav file using `wavwrite(signal, fs, filename);`.

We have provided a base Pong program in GameSalad, but if your group would like to use a different framework or programming language feel free. Some other frameworks are Unity(2d walkthrough) or Processing.

I posted a short tutorial on starting out with GameSalad. But also check out the GameSalad Cookbook.

YouTube GameSalad Tutorial: https://www.youtube.com/watch?v=lYEUHuNds4I

Afterwards please playtest your game with people that are not in the class.

Deliverables:

Please write up a short design document (100-250 words) discussing how you used sound and how you think that can add to the game experiences (see if you can tie in with some discussions we've seen in lectures about loudness, masking, expectation, or anything about the cochlea. A google doc is great!). In your design document please also talk about what you learned through the playtest.

Create 2 short videos. One of a pitch and one of a playtest.

Send me the game program, the videos and the write up and have it ready to play at the beginning of lab this Tuesday (we'll be rotating and playing each other's games for the first 15-20 minutes).