Music 171: Computer Music I
Assignment #4,
Due: Friday, November 1, 2019

This assignment is to make a Pd patch that implements additive synthesis and an ADSR envelope while synthesizing 2 clarinet-like sounds:

1. use **analysis** plot available in the lecture slides to obtain actual frequency and amplitude values of sinusoidal components in the sound;
2. use **theoretical** harmonics (every odd harmonic) each having an amplitude of one over its harmonic number (like a square wave);

Consider the following steps when making your patch:

1. Your additive synthesizer should have at least 5 (but no more than 10) sinusoidal oscillators created using the `osc~` object. The frequencies may be set using a single message that is unpacked (using the `unpack` object).
2. For **theoretical** synthesis, calculate the frequency of each oscillator according to some fundamental (sounding) frequency and its harmonic number. Changing the fundamental will change the pitch.
3. The output of each oscillator should be multiplied by an ASR envelope, defined with a single message having 3 values: the duration (ms) of the attack (A), sustain (S) and release (R) segments. The attack goes to level 1 and the whole envelope can be scaled at the output. The ASR should be created as an *abstraction*[1]:
   - it should take a single message as its input (through an inlet) with 3 values for A, S, and R;
   - unpack (using the `unpack` object), and send values where needed;
   - use a single `line~` object with a message implementing the attack (A) or "fade in", and the release (R) or "fade out".
   - use a single `delay` object to trigger the release after the sum of attack and sustain times;
   - hint: you will likely need a `float` object to hold the R value, so that it doesn't trigger the release before the "bang" from the `delay`.
4. Apply an ASR (with different parameters) to each oscillator before summing them to a final output. A general rule of thumb is that higher frequencies take longer to reach their steady state and are the first to decay (i.e. longer fade in and out for higher frequencies).

[1] to be discussed further in class, an abstraction is a way to avoid duplicating code/objects