

# Music 171: Audio Synthesis in Pd

Instructor: William Brent; [w@williambrent.com](mailto:w@williambrent.com); <http://www.williambrent.com>.

Textbook: *The Theory and Technique of Electronic Music* by Miller Puckette. Available online at <http://crca.ucsd.edu/~msp/techniques.htm>

Software: *Pure Data*. Pd is free and multi-platform for Windows, Linux, and Mac OSX. Download it at <http://www.crca.ucsd.edu/~msp/software.html>

This course introduces audio synthesis using the Pure Data (Pd) programming environment, with a focus on understanding exactly *why* the sounds we'll be making behave the way they do. We will work through many standard synthesis techniques, learning reliable Pd strategies for managing control information and sequencing as we go. There are 8 assignments to complete, which fairly consistently ramp up in difficulty. It is essential that you keep up with these tasks—if you're stuck on any aspect of an assignment, don't hesitate! See me for help as soon as possible.

Your final project will be to create an instrument in Pd that incorporates at least three of the synthesis techniques we cover. You will also perform or play back a very short demonstration piece that highlights the features (or quirks) of your instrument.

## Primary areas of investigation

1. Making & mixing sinusoids.
  - Reading: Chapter 1.
  - Objects to familiarize yourself with: `osc~`, `phasor~`, `cos~`, `+~`, `*~`, `dac~`.
2. Enveloping.
  - Reading: Chapter 1 (especially the example in section 1.9).
  - Objects to familiarize yourself with: `line~`, `metro`, `snapshot~`, `delay`.
  - Sub-patching via `[pd ...]`
3. Control information, sequencing and scheduling.
  - Reading: Chapter 3.
  - Objects to familiarize yourself with: `arrays`, `tabread`, `list`, `list length`, `list split`, `select`, `pack`
4. Wavetable synthesis.
  - Reading: Chapter 2 (pp. 27-46).
  - Objects to familiarize yourself with: `mtof`, `ftom`, `rmstodb`, `dbtorms`; `arrays`; `tabwrite`, `tabread~`, `tabwrite~`, `tabread4~`.
5. Sample manipulation.
  - Reading: Chapter 2 (pp. 47-57).
  - Objects to familiarize yourself with: `tabplay~`, `soundfiler`.

6. Making noise & subtractive synthesis.
  - Reading: Chapter 8.
  - Objects to familiarize yourself with: noise~, bp~, vcf~, lop~, hip~.
7. Amplitude and frequency modulation.
  - Reading: Chapter 5.
  - The difference between sub-patches and abstractions.
  - polyphonic voice management in Pd.
8. Delays & Doppler pitch shifting.
  - Reading: Chapter 7.
  - Objects to familiarize yourself with: delwrite~, delread~, vd~.

## Assignments

There are eight assignments spread throughout the course, as described below. A final presentation is due during the final exam time.

### Assignment 1

Make a Pd patch that produces a bank of 7 sine waves that are all tuned within the same octave. Use separate low-frequency oscillators (LFOs) to control their amplitudes so that they enter and leave the texture in unpredictable sequences. Be sure to limit their individual amplitudes so that the sum never clips.

### Assignment 2

Compose a short, simple monophonic melody, and create an additive synthesis instrument to play it with. Your instrument's sound should be the sum of 7 sine waves with frequencies that are integer multiples of the fundamental pitch. For instance, if the first note in your melody is A 440, the upper 6 oscillators should be tuned to 880, 1320, 1760, 2200, 2640, and 3080. These harmonics should all have amplitudes that are proportionally much lower than the fundamental so that the sum fuses into one tone.

The interface to your instrument should be a row of 8 message boxes with fundamental frequency values, and you should be able to play your melody by clicking on the messages in sequence. When triggered, each note should be appropriately enveloped so that it rises to full volume then dies away after a short amount of time.

Extra credit: figure out how to use the [key] or [keyname] objects to allow you to play the instrument with your computer keyboard.

### Assignment 3

Develop your additive synthesis instrument from assignment 2 so that the amplitude relationships of the partials are not fixed, but evolve over the course of each note. The upper harmonics should still always be lower than the fundamental frequency.

Then, choose a single voice from any piece that is at least 1 minute long (or compose your own), and create a sequencer to drive your new instrument. This will require that you notate your melody as 2 lists: one full of pitches, and one full of durations. These lists can be stored and drawn from using either arrays or combinations of list objects.

### Assignment 4

Make a wavetable synthesis patch that plays arpeggios ascending from any flavor of chord based on A, to any type of B chord, to a C chord, etc. A number box should control the number of notes per second. Use an array of numbers to control the pitches with [tabread].

Make the wavetable reader choose randomly between several different hand-drawn waveforms on each note.

### Assignment 5

Make a patch that plays forward and backward, in a loop, through the “voice.wav” sample (from the Pd distribution; look in the subdirectory “sound”).

Make sure you read through the sample at the correct rate. You should hear the entire phrase “continuous soft and relaxing”, intelligibly, going forward. You might want the backward part to go by a bit faster than the forward part. Add a slider to control the playback rate, along with a few message buttons for some useful preset rates.

The patch should load the sample into a wavetable using [loadbang] and [soundfiler] objects (as in the documentation patch B07.sampler.pd), but make sure the wavetable in this case is big enough to hold the entire soundfile. To keep the patch itself reasonably small, clear the “save contents” property for the array.

From this point forward, all of your patches should have a master volume control that adjusts the main output level in dB. You can accomplish this using the [dbtorms] object.

Extra credit: use the [openpanel] object to allow users to load any sound file they want.

## Assignment 6

Create a patch that loads an extremely noisy sound file and plays it back in a way that jumps to random locations, in small grains (if you need help finding/creating a noisy sound source, see me for suggestions). A number box should control the size of the grain; the transposition should be kept constant when you change the grain size. (In other words, the grains should come out at the original pitch of the sample.) All grains should be run through a [vcf~] object, and the patch should randomly choose different filtering parameters for each grain.

You will need to envelope each grain to eliminate clicks (or buzzes, for really small grain sizes.) An example is in patch B09 in the documentation. You'll also need to use one or more samphold~ objects (as shown in example B11.)

## Assignment 7

Create two streamlined abstractions: one that performs AM synthesis and takes arguments for carrier and modulator frequencies, and another that performs FM synthesis and takes arguments for carrier and modulator frequencies, as well as a modulation index.

Next, create a master patch that calls several instances of these abstractions to allow polyphonic instrument playback. Create a randomly generated or composed sequence of notes, using several [random] objects to constantly vary the synthesis parameters from one note to the next.

Extra credit: use the [notein] object to control your polyphonic instrument with a MIDI keyboard. Or, use DarwiinRemoteOSC and a Nintendo Wii remote to control synthesis parameters live.

## Assignment 8

Put together an abstraction that realizes the Karplus-Strong plucked string algorithm using delay reads and writes, with feedback. Your abstraction should let you choose between at least three different types of "noise bursts" to initiate the synthesis process. It should also include a creation argument that clearly affects string timbre.

In a master patch, create three or four polyphonic string instruments, each with a different string timbre. Transcribe the exposition of any Fugue from JS Bach's Well-tempered Clavier into pitch and duration information, and play all of the voices using your plucked string instruments. If you need help with the transcription process, or would like to compose your own material, please check with me first.