MUS 4711 –
Interactive Computer Music

Course Lecture Notes

MODULE 3: Interactivity

PART 12: Pitch, Env, and Impulse Tracking

Objects – [fzero~], [retune~] (alternate usage), [pipo~ yin] and [pipo~ psy] (requires MuBu external library), [zsa.freqpeak] via the [zsa.easy_freqpeak~] abstraction (requires zsa.descriptors external library), [fiddle~] and [sigmund~] and [bonk~] (64bit versions are at https://github.com/v7b1), [patcherargs], [mstoamps~], [slide~], [zl.slice], [multislider]

Bring a harmonica or other instrument!
Pitch Tracking – This is the big one. We're going to look at a number of ways to follow the pitch of an audio signal using a number of different objects. All are different and some are slightly better than others, some are better in some ranges or applications than others, some can be filtered or not, some output in MIDI values, and others use frequency. This is designed as a shootout between the seven different methods most commonly used. You will need to install the MuBu and zsa.descriptor libraries from the Package Manager, and download [fiddle~] and [sigmund~] externals from the included links. Grab the Max External (MXE64)and Max Helpfile (MAXHELP) and put them in the same directory as the patch.

Let's have a pitch off!

Envelope Tracking – Here's a handy little abstraction from the Cycling '74 forums by Steven Miller (no, not the Nazi…). It takes an audio signal and uses [abs~] to get rid of negative values. Then it uses [slide~] to filter the signal and track its amplitude. Since [slide~] takes samples, and not MS, the [mstoamps~] object is used to convert *at the current sample rate* – really good feature as that can change! Finally, the [patcherargs] object autoloads initial values for an abstraction that load with the patch. Good way to set initial values you'll be using a lot.

Impulse Tracking – Download the [bonk~] External and Helpfile from the link and put them in the same directory as the patch. [bonk~] tracks the loudness of a signal in two ways, an 11-band "raw" version, and a summed "cooked" version. We'll be looking at the "cooked version. [zl.slice] outputs the first value out of the left outlet, then the rest out the right. We're using that to strip out unneeded values from the list before outputting them for display to the [multislider] and to the second [zl.slice]. The output here goes to a [>=] logic operator. If the signal is over 66 (scale is 0 – 100 for practical purposes), then it's true, otherwise false. This lets you easily use the volume of an audio signal to trigger events. Here, the bassoon never hits that value, the cello ramps up to it, and the drum loop's harder beats hit it while the rest does not.

Max Puzzle 8 – Time for a challenge! Using your favorite pitch tracking method, an envelope tracking method, create a patch that plays back a pre-recorded 1-minute clip of yourself playing an instrument that is then:

1) Harmonized with through the pitch analysis (HINT, convert to/from MIDI and drive a basic antialiasing oscillator – don't get too fancy!)
2) Uses the envelope tracker to control a parameter of an effect from a previous week that's applied to the audio file, and a second effect parameter for the harmonized part

HINT – Remember to use [snapshot~] to convert signals from the envelope tracker into numbers!