21M.370 Digital Instrument Design
Lab assignment 3 - Due March 10 at 2:00pm

Deliverables:

1. A PD synthesizer patch. This should be a patch you have created on your own, although it can build on a past assignment and include elements from examples. You only need to submit the main.pd patch.
2. A PD control patch. You can modify the example patch to include whatever controls you wish, though there are no requirements other than routing messages from Python to Automatonism.
3. A python script which processes keyboard and mouse input, based on the example. Your script should use different mappings than the example for both keyboard control and mouse.

Assignment description:

Our goals for this week are to get used to working with both buttons and continuous signals to control synth parameters.

You will need to install the ‘pynput’ python module to get mouse position information. ‘pip install pynput should do it. Because computers often restrict access to keystrokes, the easiest way to get that information is to use PD to get keystroke information and then send it to Python. If you can get either key or mouse information differently than shown in the example that is totally fine.

Watch the tutorial video and look at the example files. These will demonstrate one way to use keys and mouse, and will highlight some questions you will need to answer for yourself:

1. How will you group buttons? What happens when you press multiple buttons within the same group? How can you efficiently map small numbers of buttons? We are using a keyboard with lots of buttons now, but in the future you may want to make an instrument with a smaller number of buttons.

   In particular, look at the discussion on chording, binary input, and grey code in Marije's book.
2. When dealing with continuous signals there are lots of way to process them. The example shows how to calculate the magnitude of mouse movement by constantly checking the difference between current and previous mouse position. It also maps the X and Y position to different synth parameters.

   One thing to keep in mind is that we often want to smooth out continuous signals, which is what the leaky integrator and lowpass filter do. We also often want to scale continuous parameters non-linearly. Read the section 22.7 from Marije's book to see different ways of turning linear into non-linear signals.
3. Many things you might want to do can be abstracted to make your code simpler. Be on the lookout for useful 'helper functions'.

   In particular, the example moves towards an approach of using a single PD send, and using Python to define both the send destination as well as the method. This requires sending two OSC messages. Maybe there would be a way to create helper function to let you use a single line of code to send a complete message with object name, parameter name, and parameter value. This would make it easier in the future when you are controlling lots of parameters.
4. A special case not covered in the example is when you want multiple controls to affect the same parameter. In this case, you might need to make a dedicated function or class to merge the control data.

The list of resources for this lab:

1. Video Tutorial
2. Example PD patches and Python script.
   The example files are in the github under class/labs/lab3/keysynth
3. A chapter on buttons from Marije's book, included with the assignment on Canvas.
4. PD keygrid object, located under NIME/PD/externals.
5. Video on including the externals folder in PD so your patch can find keygrid as well as sendOSC and receiveOSC.
6. Here is a link to the playlist for the class, with all videos to date: https://youtube.com/playlist?list=PLHQPaeahzMrMDn-4Z-JfLQWPv8Gj2jYJw