

Lab 1: Making Music With Pd

The Write-up (due on Monday, April 4 at 10PM)

For the write-up, please follow all the directives and answer all the questions asked in this lab and include pd patches of your solutions. After completing the lab, create a text-file answering questions from the lab, describing what you authored/modified and anything we need to know to test your patch. Put this file in folder named lab1. Then zip up your lab1 folder and email it to Karl and Matt. Please make the subject of the email "Lab 1 - yourlastname".

1) Install Pd

- Go here: <http://puredata.info/community/projects/software/pd-extended> and download the version on Pd-extended that will work on your computer.

2) Download The Lab

- Go here: http://www.mat.ucsb.edu/240/240C/software_lab_pd.zip

3) Play Around With the Patches

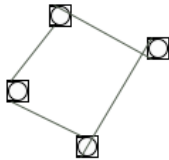
- Open the patch labeled myinstrument. Play around with this patch. You should be able to exhaust its musical potential in a matter of minutes. Reflect on its strengths and limitations.
- Try to understand how the patch works as a piece of software. Try clicking on some boxes and reading through the code. But, please don't get hung up on the arcana. As always, if you get stuck, ask for help rather than waste time.

4) Explore Documentation

- Right-click or option-click on any object to get a contextual menu including "help," which opens that object's help patch.
- Right-click on a blank portion of a Pd patch. Now when you select "help" you get a list of Pd's built-in objects, arranged by category.
- In the upper right hand corner of each Pd window is a "help" menu. This accesses the Pd tutorials as well as some online reference documentation.

5) Short Questions

- Why is the following patch a bad idea? What happens when you click one of those bangs?



- Make a patch that calculates the function $1-x$ where x is an input number. Your patch should have a number box for input and another number box to show the output of your function. Save the patch in a file called OneMinus.

6) Key Repeat (getting back to the myinstrument patch)

- What happens with myinstrument when you hold down a key (The behavior may depend on what operating system you are running)?
- Is this behavior desirable? Why or why not? What operating system are you running?

7) Design A Different Musical Interaction

- Pick one (or a small number) of the ideas below and work on it in depth, iterating on both the program/test/debug cycle as well as the design/implement/play cycle to craft something that has actual musical potential or is at least more fun to play. If you have an existing idea for your class project, you could use this lab to start thinking about implementing some of the modes and mappings. By all means, if you're inspired to try

something else, go for it. If you'd rather spend today getting more of a broad sense of Pd's capabilities, feel free to work on many of these suggestions. If you prefer a different platform, you may port this Pd code to Processing, MaxMSP, or Chuck and move on from there.

- Involve more QWERTY keys
 - Involve the mouse (see the [MouseState] object)
 - Load in a larger collection of samples.
 - Implement a mechanism to switch among banks of samples
 - Design a way for the parameters of each triggered note to depend on multiple key presses (multiple gestures to one result). For example, maybe only the space bar triggers notes, and all the other keys determine parameters of notes.
 - Make your patch automatically generate a bass-line as a function of the key presses
 - Set multiple parameters modally, as volume works in the sample patch
 - Use chording: keep track of all the keys that are currently pressed, and use only that information to set the parameters for each note.
- One gesture to multiple results
- Use the "metro" object to trigger a steady stream of notes. Now you have two new parameters: repetition rate, and whether the metro is on or off.
 - Use the "counter" object to step through a cycle (of samples, parameter settings, etc.)
 - You could combine "metro" and "counter" to build a rudimentary sequencer that can step through a rhythmic pattern
 - Invent a mechanism to record short sequences of key-presses and play them back in time.
 - Incorporate looping or other interactive controls over the sound file playback
 - Use Pd's "spigot" object to route control information to different parts of your patch at different times
 - Use some additional signal processing such as a filter, delay line, reverb, tremolo, etc. This gives you more parameters to control.
 - Polyphony: make it so the patch can play multiple samples at the same time. (Hint: put multiple copies of "play-sample" in your patch)
 - Use a totally different form of sound synthesis, such as FM, granular, or physical modeling.

8) The Pd Community

- There is a large, dedicated, and very generous community of Pd users on the Internet. Do some web searching (e.g., with a search engine, or else starting from some more specific resources) and look for interesting externals and/or patches. Download, install, and play **with at least one**. Can you incorporate it into what you programmed in the previous part?
- For more help in finding resources, don't forget to look at some links on the 240C site.