AUD6204 - Programming Environments
Lesson 1.1

PD Overview

Contents

1 Introduction

Pure Data (or just PD) is an open source visual programming language for multimedia. As we will see we create computer code (primarily) by connecting different 'objects' together (a bit like modular synthesis, or even Meccano).

Please Note:

I will only be giving shortcuts for Mac OS. If you are using Windows/Linux it should, in most cases, only be a matter of using **ctrl** *instead of* **cmd** *but you may need to search these out on your own.*

As PD is open source you may be using a different distribution to me and some features may not be available (or you may have features I do not). I will endeavour to only use basic objects whenever possible. Again though, you may need to do some searching on Google to find objects if you do not have them in your distribution.

1.1 Before we start

Download and install SimpleSynth from: `https://notahat.com/simplesynth/`
We will use this to trigger simple MIDI sounds before we begin looking at sampling and synthesis.
Open SimpleSynth and set the MIDI input to "SimpleSynth virtual input"
NOTE: you need to have SimpleSynth open before you open PD or PD will not be able to communicate with it!

Download and install PureData from: `https://puredata.info/downloads`
You can start with the "vanilla" distribution and add libraries as necessary
When PureData is installed and open go to:
Media → MIDI Settings...
and set the MIDI output from Pure Data to "SimpleSynth virtual input"

2 The Basics

We will need some basic commands and understanding in order to navigate and code within PD.

- Open the **PD Browser**: **cmd+b**
  The browser is where you can find PD example patches.
- Create a **new patch**: **cmd+n**
  Patches/canvases are where we 'write' our code.
- Enable/disable **Edit Mode**: **cmd+e / cmd+alt+e**
  When in edit mode we can create new objects and connections.
  When edit mode is disabled we can interact/perform with our patches.
- Create a generic **new object**: **cmd+1**
  If there is no shortcut available to create an object we will need to create a generic object and type in the name.
  A useful list of (some) objects and their function can be found here.
- Open the **PD Window** (see Fig 1): **cmd+r**
  The '**PD Window**' displays messages from PD. We will need to look at it when debugging and analysing our code.
- **Clear** the PD Window: **cmd+shift+l**
- Enable **Autopatch**: **cmd+shift+a**
  Autopatch will automatically connect new objects together (particularly useful when used together with other shortcuts).
- **Automatically connect** two objects: select them and press **cmd+k**
- **Tidy** objects: select them and press **cmd+shift+r**

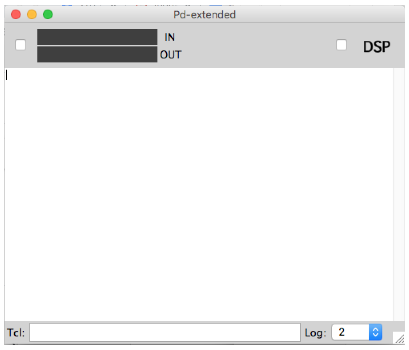

Figure 1: The '**PD Window**'

Throughout your time using PD you should always:

- Look at the **helpfiles** for object (right click on an object and select "Help")
- **Experiment** with creating your own patches - you will learn much quicker that way so anytime you find out something new you should try to use it in a patch!
- Keep your code **tidy**!
- Leave **comments** in your code! That way when you come back to it you (and I) know what it does.
- **Write** your ideas down in a notebook (using pen/pencil and paper, I know proper old school)!

3 Connecting Objects

In PD the bottom connection points of objects are outputs, and the top connection points are inputs (see Fig. 2). We connect objects by clicking (and holding) on an output, and then dragging the resulting cable to an input of another object.

3.1 Hot & Cold inputs

In PD, inputs can be either 'hot' or 'cold,' and as a general rule **leftmost inlets are hot**, while **other inlets are cold**.

3.1.1 Hot Inputs

Any data/trigger received to a 'hot input' will result in the object outputting something (a number, a symbol, etc).

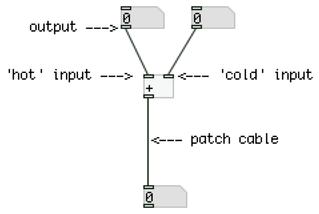

Figure 2: Inputs to objects in PD can be either ‘hot’ or ‘cold.’ The output of one object is connected to the input of another via a ‘patch cable.’

3.1.2 Cold Inputs

Any data/trigger received to a ‘cold input’ will result in the object storing that data until it receives a trigger in the ‘hot input’[1].

[1]Don’t worry! This will all become a lot clearer once you actually see it in action

4 Order of operations

At the most basic level, computers complete one task at a time. These tasks are completed so quickly they appear to happen at the same time, but the order things happen in is vitally important! In text based code (e.g. this) the order is simple - we start at the top and move (generally one line at a time, although that can be varied) to the bottom. In graphical languages like PureData, or Max, this top to bottom approach doesn't make sense as there is also a left/right axis...

When looking at our patch information is passed into the top of objects (the inlets, see Sec. 3.1) and out of the bottom of objects.
If an object has multiple outlets the rightmost output occurs first and the leftmost outlet occurs last - i.e. the order goes from **right** → **left**.

4.1 Algorithms

Algorithms are basically recipes. i.e. they are a serious of steps the computer must complete in a specific order to complete the task at hand. We will need to be very careful planning the order in which the computer must completes each task or are code will not behave as expected (computers don't make mistakes, they just follow the instructions they are given).

Always use print objects and the PD window to test the output of your code and see whether it is working the way you expect!