AUD6204 - Programming Environments
Lesson 4.1

Audio Basics

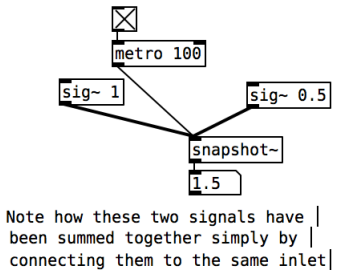

Figure 1: Connecting two audio signals to the same inputs is the same as adding/summing those signals together.

NOTES:

- In PD, audio objects are designated by a tilde ('∼')
- Audio objects in PD run at sample rate
  - i.e. if your sample rate is 44.1kHz, PD audio objects process data 44,100 times every second!!!
- Audio rate objects output values constantly!
- If we want to change the volume of a signal we need to keep some things in mind
  - Multiplying a signal by '0' results in **silence**.
  - To **increase** the volume of a signal we multiply by a value more than '1.'
  - To **attenuate** a signal we multiply by a value less than '1,' but **more than '0'**.
  - If we multiply by '-1' we **invert the polarity**.
- With the audio realm of PD there is no strict order of operation (i.e. PD calculates all signals simultaneously every sample). This means that when connecting **only** audio objects together there is no such thing as hot or cold inlets...[1]
- Because there is no order of operations and all signals are calculated at once, connecting two outlets to the same inputs (see Fig. 1) is the same as adding/summing those signals together!

[1]NOTE: In actual fact this is not entirely true, and the order of operation can be explicitly determined using sub patchers...

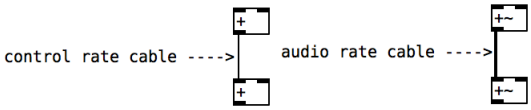

Figure 2: Connections between audio rate objects are slightly thicker than those between control rate objects.

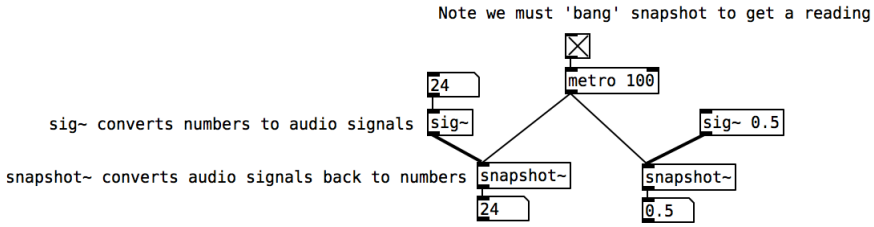

Figure 3: **sig**~ allows us to convert numbers to audio signals.
snapshot~ allows us t convert audio signals to numbers (NOTE: we have to bang snapshot for the object to take a sample).

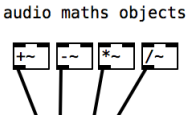

Figure 4: Audio rate math objects are the same as control rate audio objects, aside from the fact that they accept audio signals (obviously). Note that each symbol is followed by a tilde (e.g. "*****~").

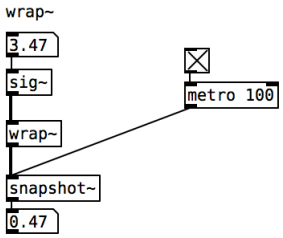

Figure 5: The '**wrap**~' object is the audio equivalent of the mod (remainder) object. However, '**wrap**~' only outputs numbers between 0→1.

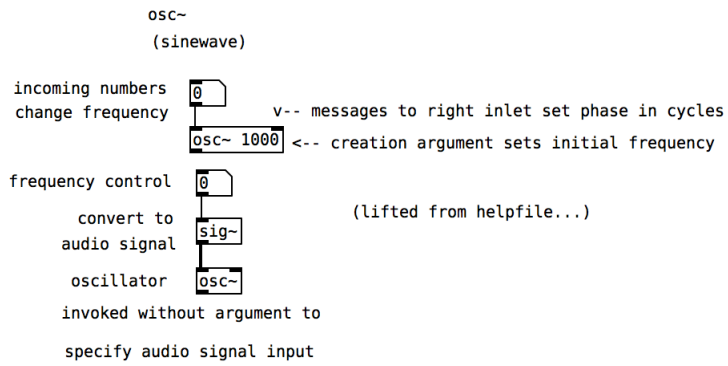

Figure 6: The '**osc**~' object generates a periodic waveform (a sine wave, actually cosine... but still). The argument determines the frequency.

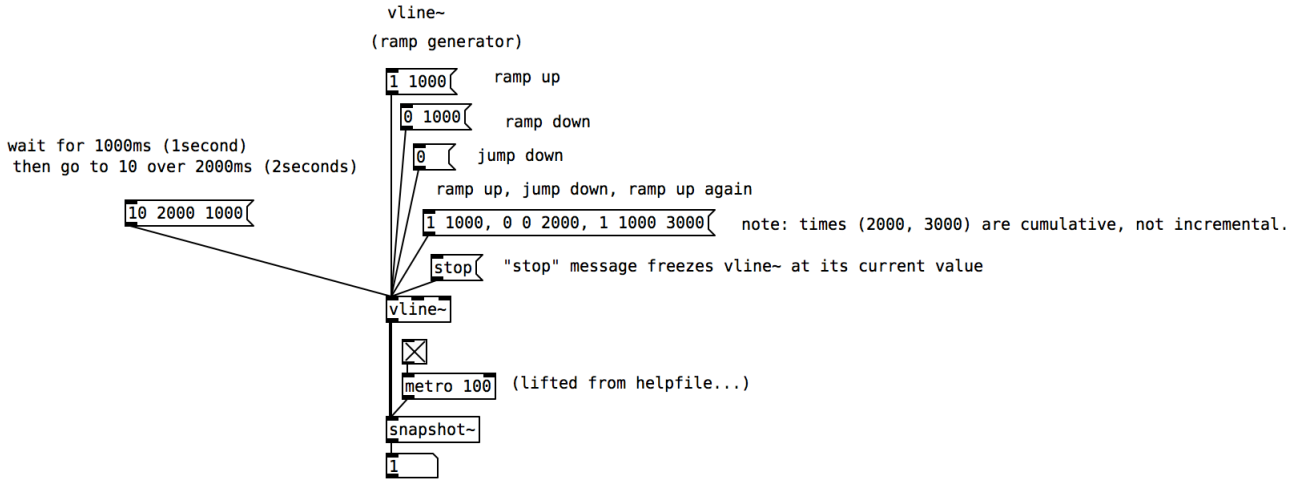

Figure 7: The '**vline**~' object outputs an audio rate ramp. If 'vline~' is given one number it immediately jumps to that value, if it is given a list of 2 numbers (X and Y) it ramps to X over Y milliseconds. If it is given a list of 3 numbers (X Y Z) it waits for Z millisecond then ramps to X over Y millisecond (this allows you to create long, multistage ramps).