UNIVERSITY OF MICHIGAN

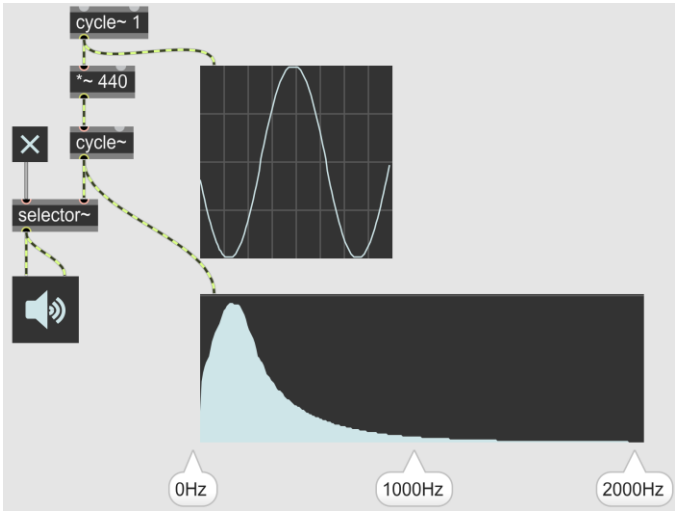
Creative Coding (PAT 204/504, Fall 2025)

Lecture 15: Polyphony

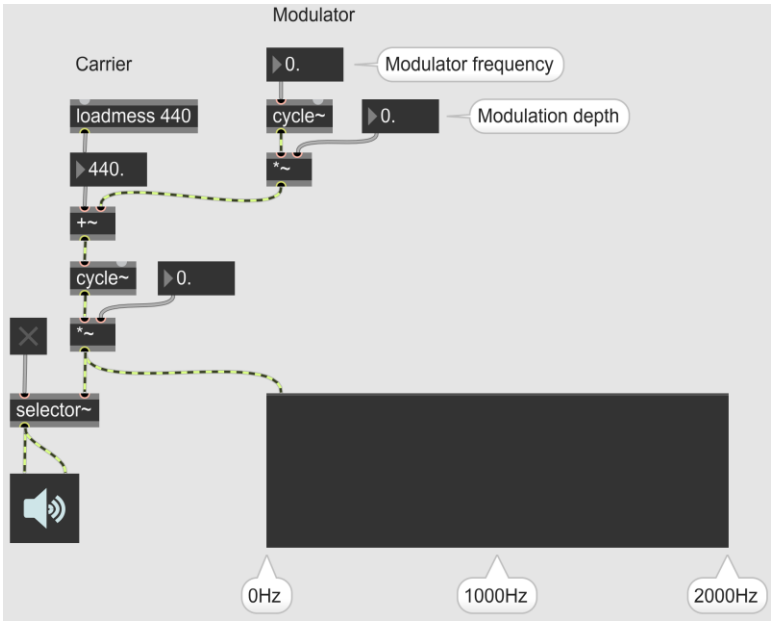
Instructor: Hao-Wen Dong

Example 1: FM Synthesis ("1_fm_synth.maxpat")

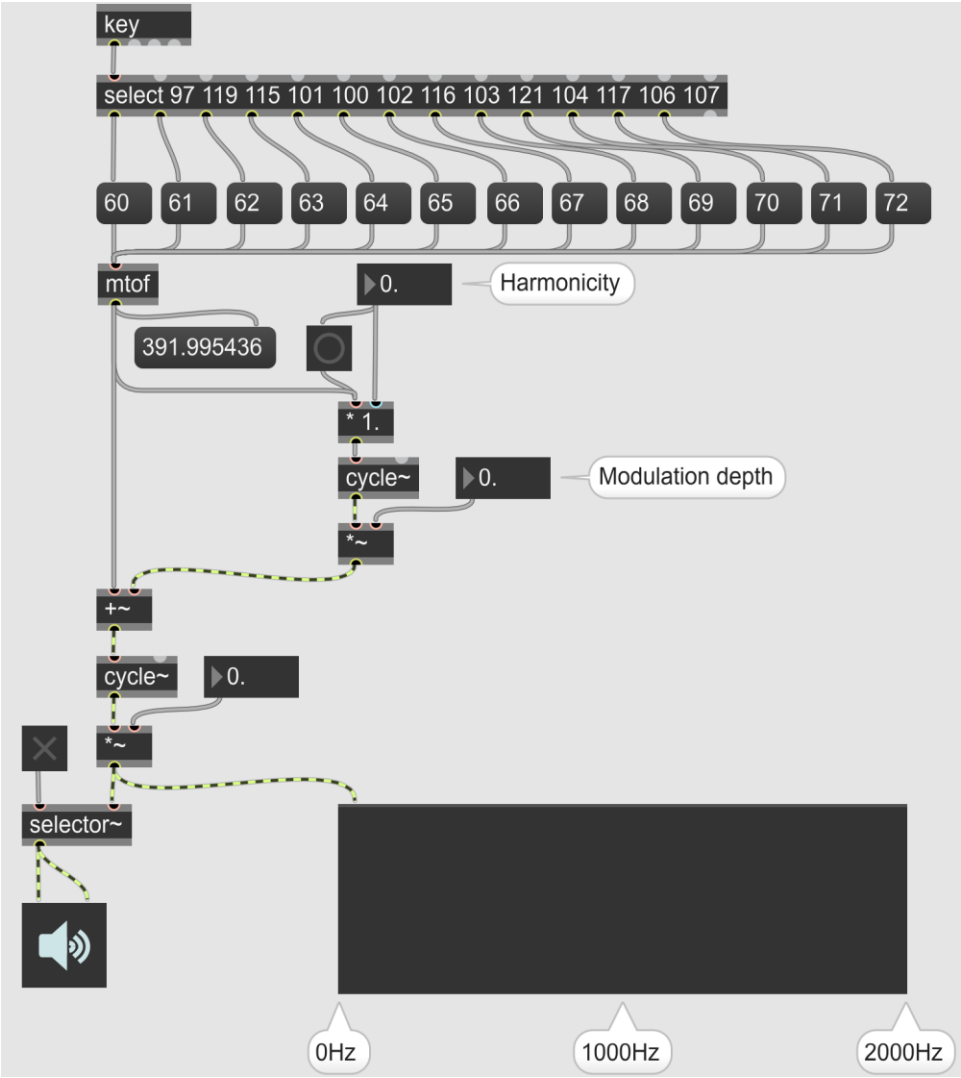
- Send a sinusoid signal to the first inlet of the "cycle~" object that "modulates" the frequency of the carrier signal (the second sinusoid signal)



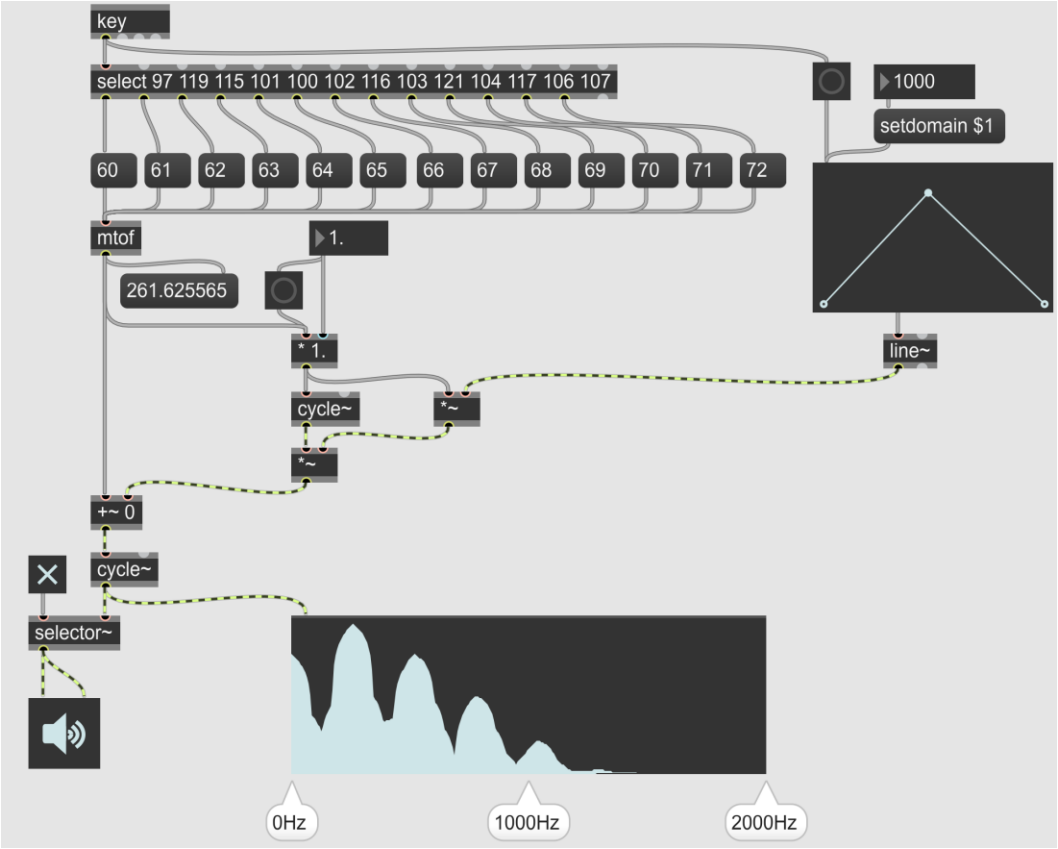
- Add a sinusoid signal to the carrier frequency to modulate the frequency



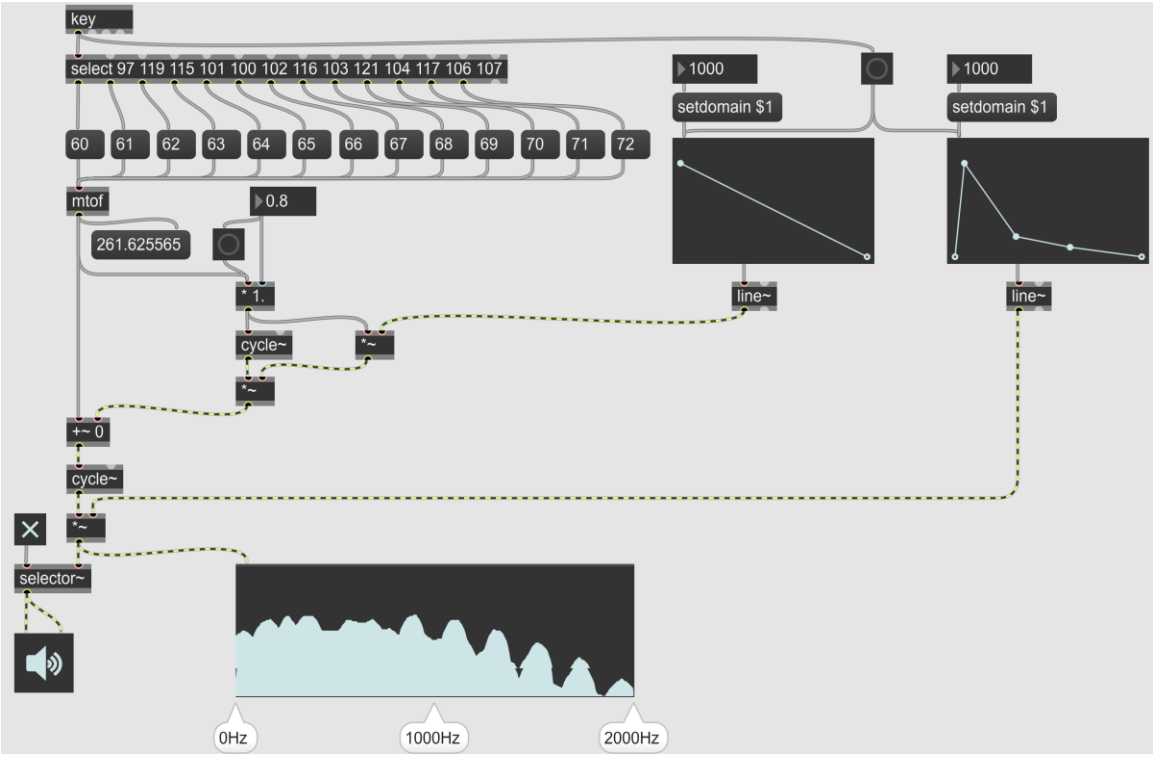
- Use the “kslider” object to allow easy control of the carrier frequency and set the modulator frequency as a specific multiple (doesn’t need to be a perfect multiple)



- Use a “function” object to create an automation that controls the modulation depth

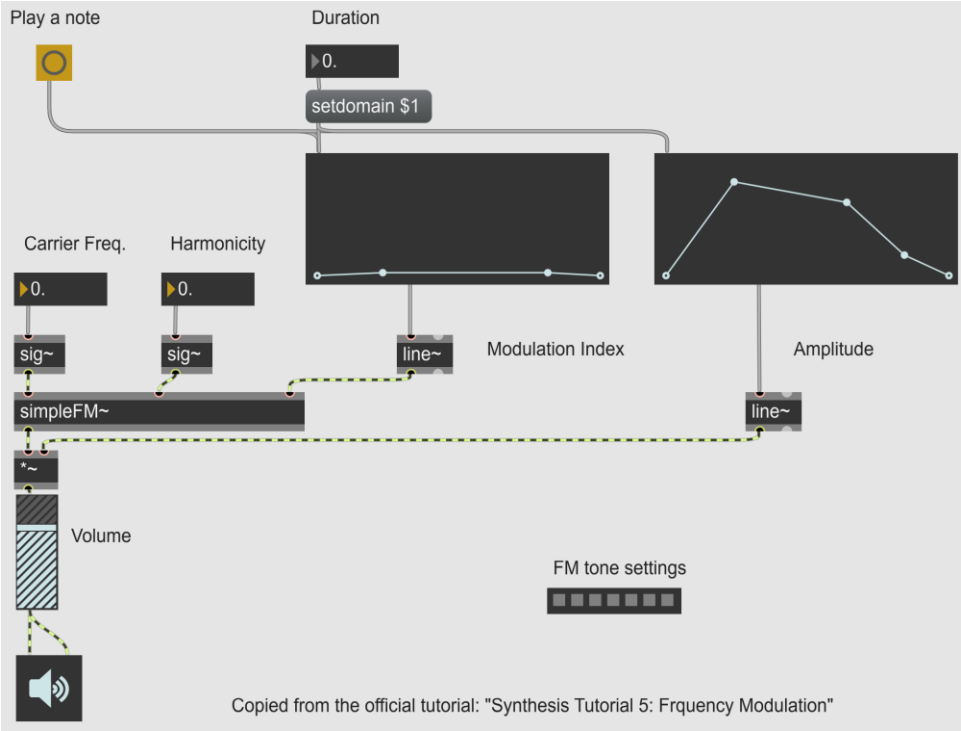


- Use another “function” to create an ADSR envelope to control the magnitude



Example 2: FM Synthesis (Alternative) (“2_FMSynthesis.maxpat”)

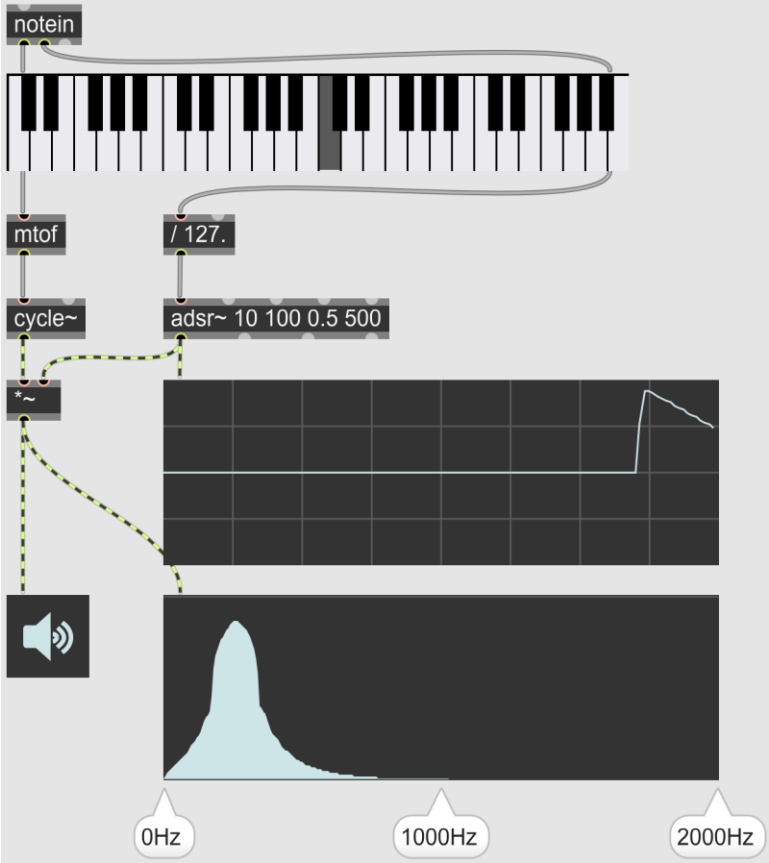
- This is the official MAX tutorial on FM Synthesis, where it provides several nice presets



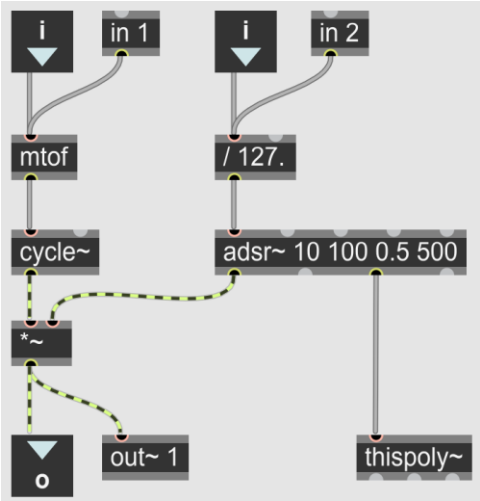
  - Use the “preset” object to store the values of all interactable objects as a preset
  - Click on a certain square to recall, and Shift-click it to store/update a preset

Example 3: Polyphony (“3_polyphony.maxpat”)

- Let’s first create a simple sinusoid oscillator with an ADSR envelope



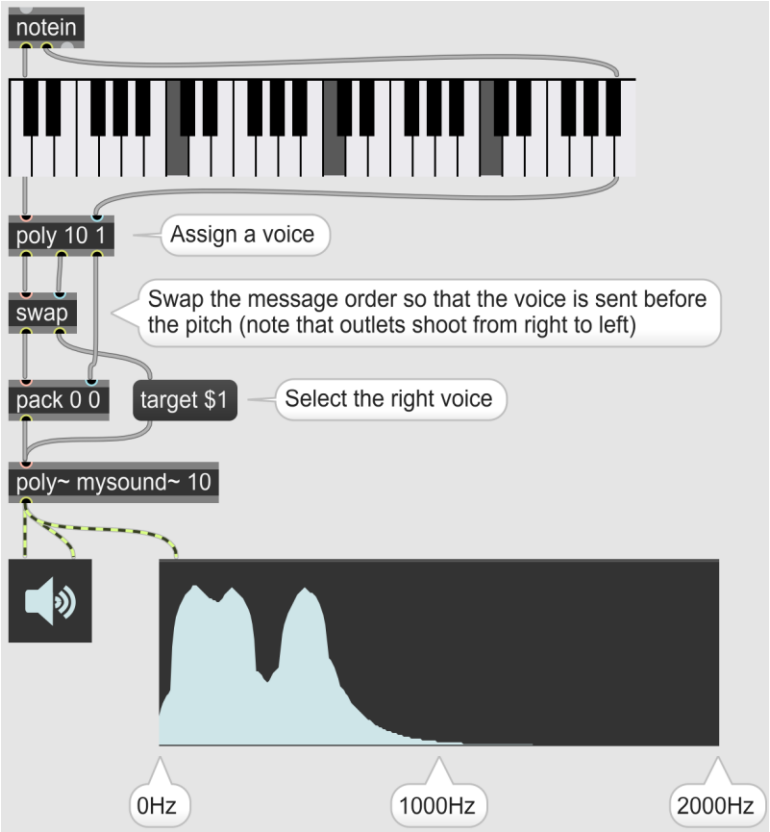
  - Use the “adsr~” object to create an ADSR envelope without creating a “function” object
- Now, let’s make this single-note synthesizer a standalone Max patch



  - The “thispoly~” object is connected to the third outlets of the “adsr” object, which sends a “mute 1” message when it completes the envelope and sends

a “mute 0” message when it restarts. The “thispoly~” allows MAX MSP to know whether the current voice is active. If a voice is not active, MAX MSP can temporally pause the computation in that voice to avoid unnecessary computation.

- Now, we can create our polyphonic with the “poly~” object that creates multiple instances of the Max patch we specify, e.g., the “mysound~” patch file in our case



  - The “poly” object handles the voice assignment that takes cares of the MIDI note on and note off messages
  - The “swap” object swaps the input message ordering (so the left inlet is sent to the right outlet, and the right inlet is sent to the left outlet), but more importantly, it changes the ordering when the message is sent. In Max, the outlets shoot from right to left, and that’s why we need the “swap” object so that the voice and the “target $1” message is sent before the MIDI pitch and velocity messages.
  - The “poly~” object accepts a “target X” message to select voice X