PAT 204/504 (Fall 2024)

Creative Coding

Lecture 25: Review & Discussions

Instructor: Hao-Wen Dong

SCHOOL OF MUSIC, THEATRE & DANCE
PERFORMING ARTS TECHNOLOGY
UNIVERSITY OF MICHIGAN

Course Evaluation

- Your feedback is highly appreciated!
- Enter at umich.bluera.com/umich

Final Project

- Milestones (all due at the specified date at **11:59 PM ET**)

Proposal	November 25	Plans (1 page)
Presentation	December 9	Showcase & report
Final report	December 15	Full report (2-3 pages)

- Instructions will be released on Gradescope
- Late submissions: **NOT accepted**

Final Project: Rubrics

Proposal	**10pt**
Presentation	**15pt**
Final report	**25pt**
Implementation	10pt
Code documentation	5pt
Explanation of design and implementation	10pt

Final Project: Presentation

- **Introduction & motivation**
  - **Why** are you interested in this topic?
  - **Who** might want to use your work?
- **Design & implementation**
  - How did you **design your work**?
  - How did you **implement your idea**?
- **Discussions**
  - **What have you found** through your experiments?
  - What are the **limitations** and **future directions**?

What is this course all about?

An introduction to principles and practices of computer programming for musical applications. Emphasis is on **creative and artistic uses of code**.

Processing

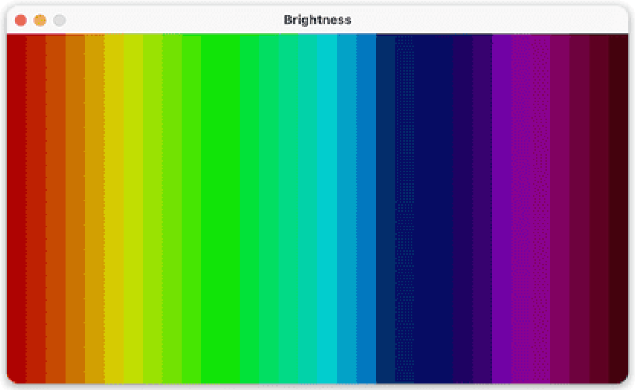
Max

Processing

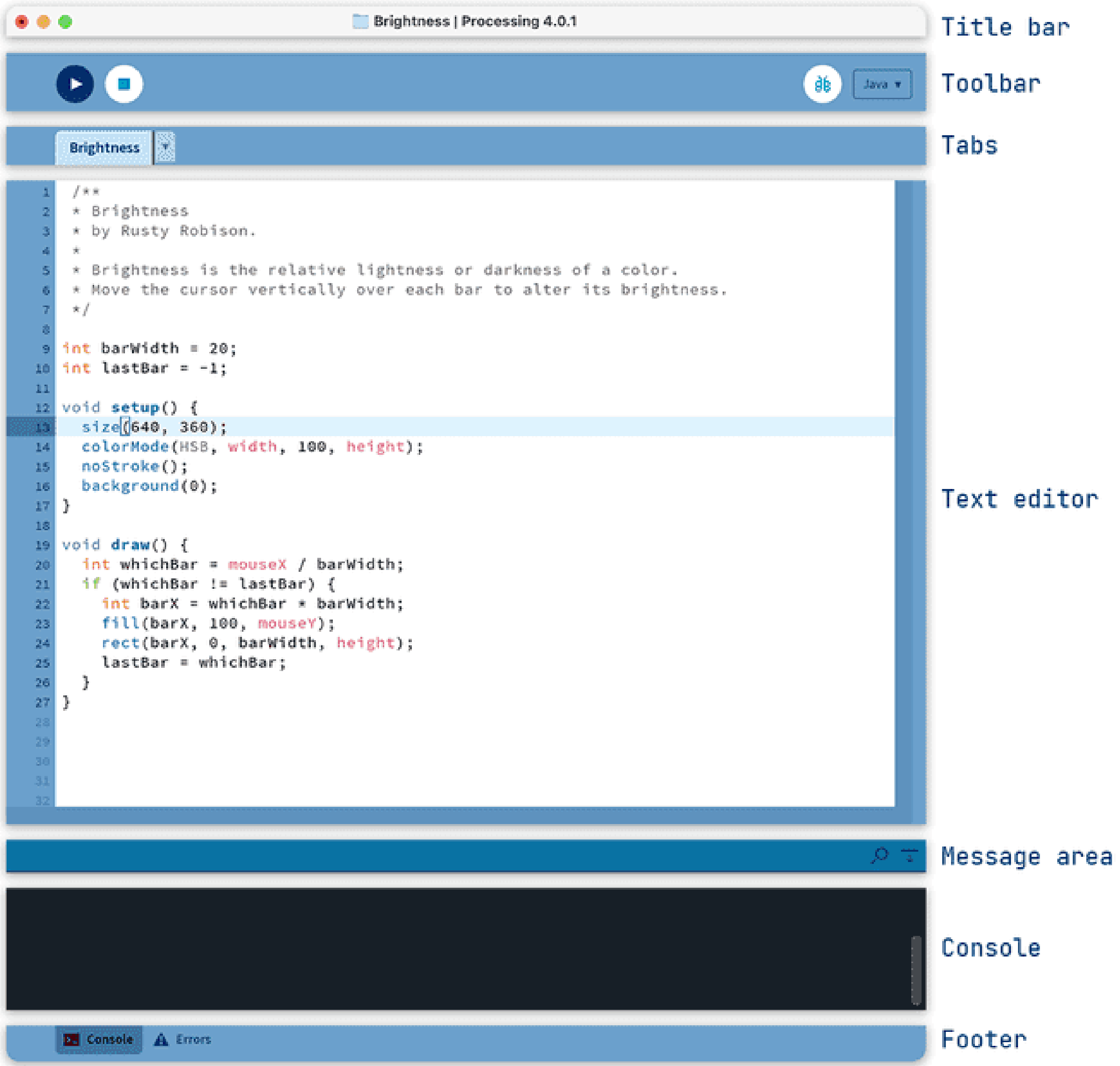
Review – Processing Basics

A Processing Sketch

- Processing comes with an IDE
  (Integrated Development Environment)
  - A **text editor**
  - A **console**
  - A **display window** (when you click the *run* button)



Display window

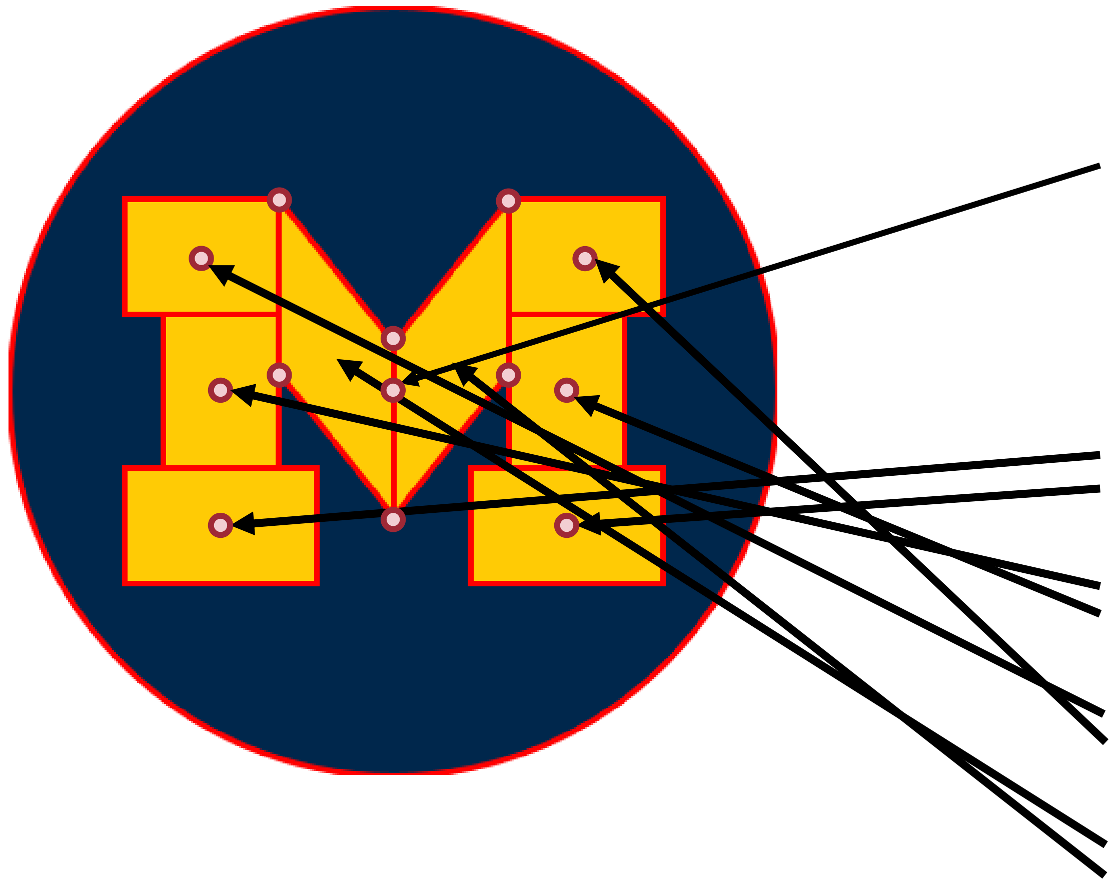


```
/**
 * Brightness
 * by Rusty Robison.
 *
 * Brightness is the relative lightness or darkness of a color.
 * Move the cursor vertically over each bar to alter its brightness.
 */

int barWidth = 20;
int lastBar = -1;

void setup() {
  size(640, 360);
  colorMode(HSB, width, 100, height);
  noStroke();
  background(0);
}

void draw() {
  int whichBar = mouseX / barWidth;
  if (whichBar != lastBar) {
    int barX = whichBar * barWidth;
    fill(barX, 100, mouseY);
    rect(barX, 0, barWidth, height);
    lastBar = whichBar;
  }
}
```

More Shapes

Circle	`circle(x, y, diameter)`
Ellipse	`ellipse(x, y, width, height)`
Square	`square(x, y, width)`
Rectangle	`rect(x, y, width, height)`
Point	`point(x, y)`
Line	`line(x1, y1, x2, y2)`
Triangle	`triangle(x1, y1, x2, y2, x3, y3)`
Quadrilateral	`quad(x1, y1, x2, y2, x3, y3, x4, y4)`

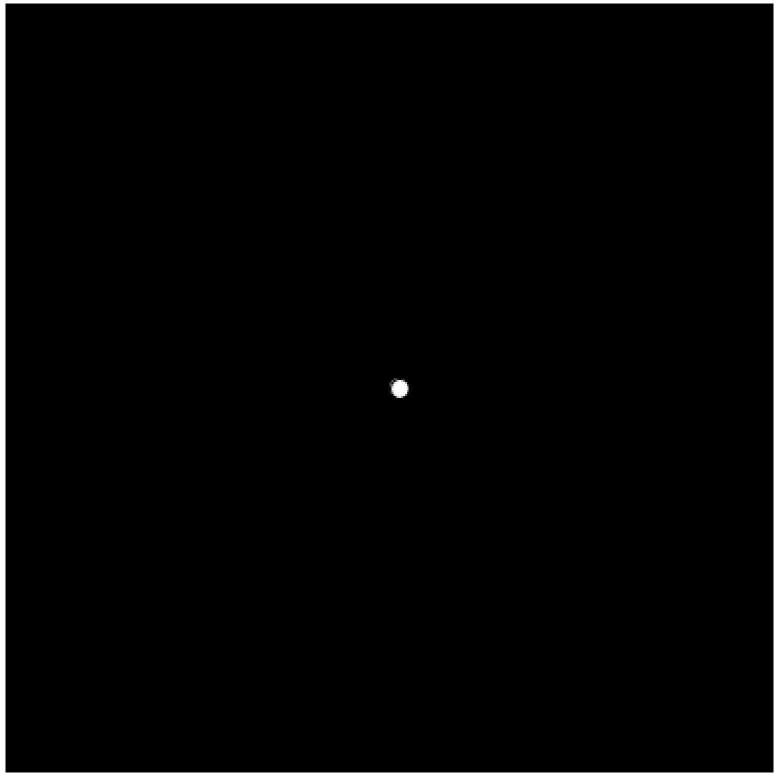
My Version



```
void setup() {
  // Create a 400x400 canvas
  size(400, 400);
}

void draw() {
  // Set the background color to white
  background(255);

  // Draw the shapes without outlines
  noStroke();

  // Draw the blue circle at the back
  fill(#00274C);
  circle(200, 200, 400);

  // Set the anchor point of rectangles to the center
  rectMode(CENTER);

  // Set up the yellow text color
  fill(#FFCB05);

  // Draw the feet
  rect(110, 270, 100, 60);
  rect(290, 270, 100, 60);

  // Draw the columns
  rect(110, 210, 60, 150);
  rect(290, 210, 60, 150);

  // Draw the caps
  rect(100, 130, 80, 60);
  rect(300, 130, 80, 60);

  // Draw the "V"
  quad(140, 100, 140, 190, 200, 265, 200, 175);
  quad(260, 100, 260, 190, 200, 265, 200, 175);
}
```

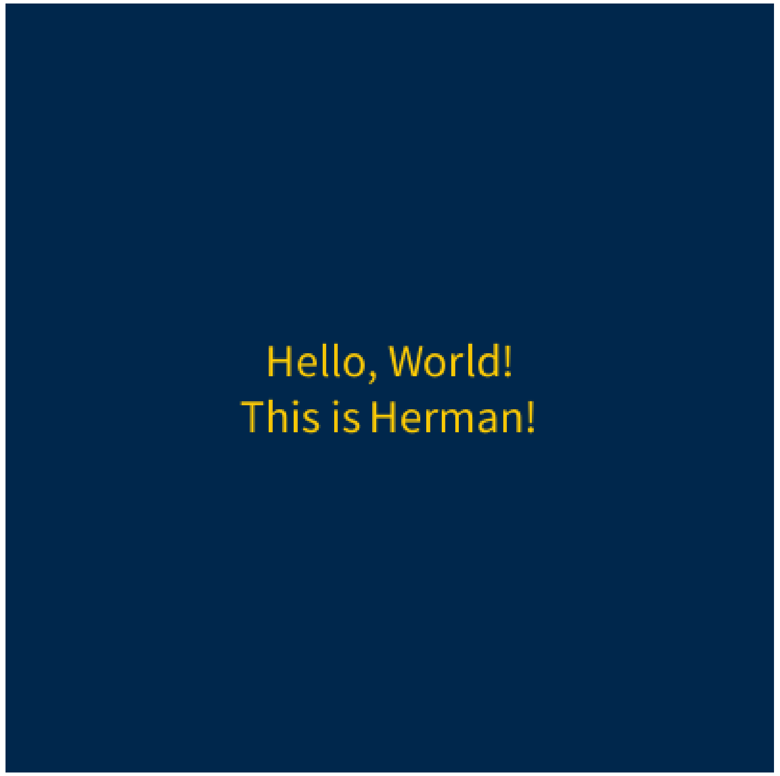
Bouncing Ball

```
float ballSize = 10;  // Size of the ball
float x;  // Current x-position of the ball
float speedX = 5;  // Current speed of the ball
boolean saveFrames = false;

void setup() {
  // Create a 400x400 canvas
  size(400, 400);

  // Initialize the ball position
  x = width / 2;
}


void draw() {
  // Create a black background
  background(0);

  // Check if the ball hits the left/right border
  if (x > width - ballSize / 2) {
    speedX = -speedX;
  } else if (x < ballSize / 2) {
    speedX = -speedX;
  }

  // Move the ball
  x += speedX;

  // Draw the ball
  circle(x, 200, ballSize);
}
```



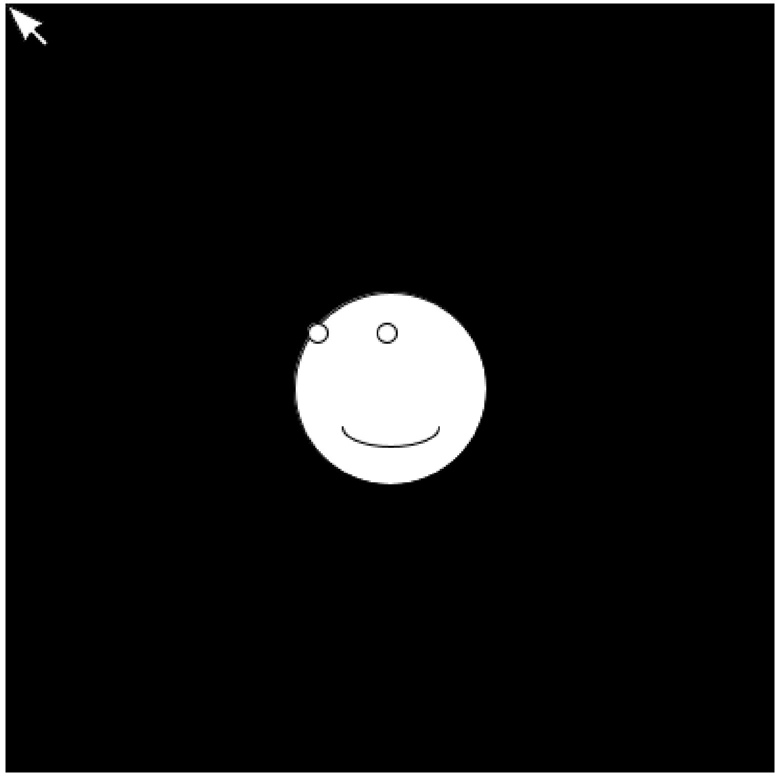
Homework 1: Bouncing Hello World

- Instructions will be released on Gradescope
- You need to find the function for **text rendering**
  - The documentation is your friend!
  - https://processing.org/reference
- You need to figure out how to calculate the **height and width of the text box**
  - There'll be many friendly hints in the instructions 😊
- Due at **11:59pm ET** on **September 6**
- Late submissions: **1 point deducted per day**



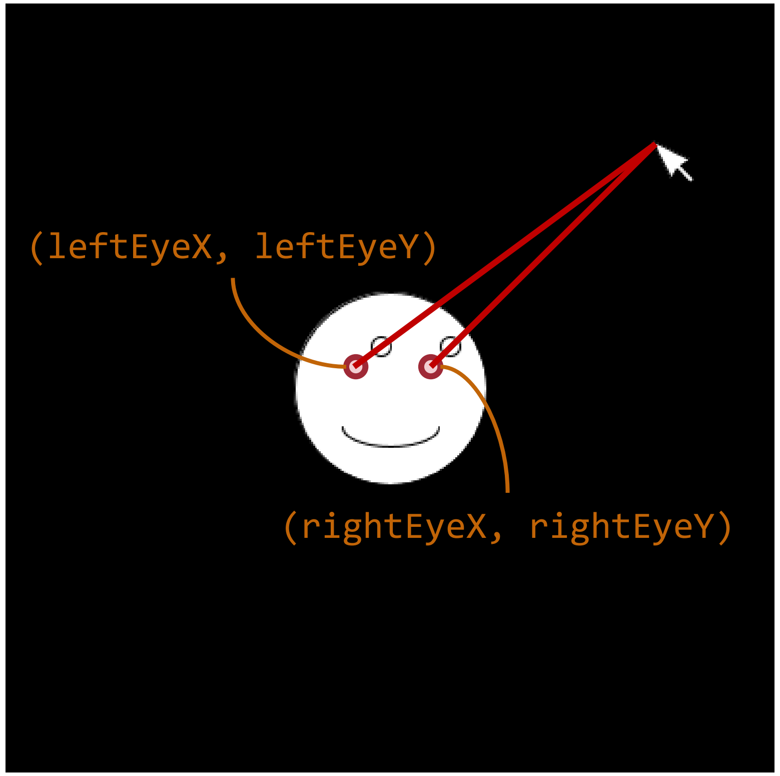
Exercise: ~~Creepy~~ Eyes

- Make a simple face where the eyes will **stare at the direction where the mouse is**!
- Hints
  - Use **mouseX** & **mouseY**
  - Use `arc()` to get the smile
    - `arc(200, 220, 50, 20, 0, PI)`



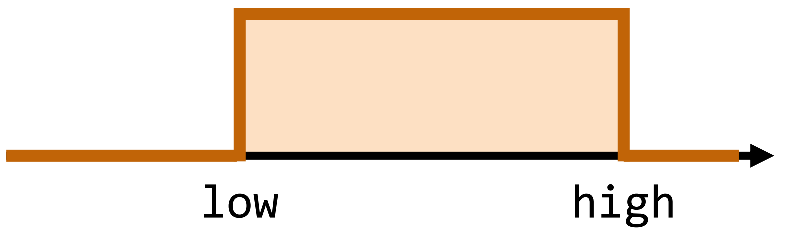
Exercise: ~~Creepy~~ Eyes

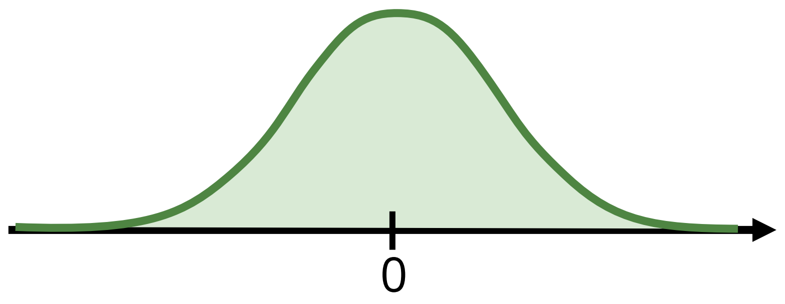
```
// Calculate the position of the eyes
leftDeltaX = (mouseX – leftEyeX) * scale;
leftDeltaY = (mouseY – leftEyeY) * scale;
rightDeltaX = (mouseX – rightEyeX) * scale;
rightDeltaY = (mouseY – rightEyeY) * scale;

// Draw the eyes
circle(
  leftEyeX + leftDeltaX, leftEyeY + leftDeltaY, 10
);
circle(
  rightEyeX + rightDeltaX, rightEyeY + rightDeltaY, 10
);
```



Randomness

- `random(high)` Generate a random number in $U[0, high]$
- `random(low, high)` Generate a random number in $U[low, high]$
- `randomGaussian()` Generate a random number in $N[0, 1]$





PVector Static Methods

- **Static methods** are methods that belong to a class (rather than an instance)

`PVector.random2D`	Create a 2D unit vector with a **random direction**
`PVector.random3D`	Create a 3D unit vector with a **random direction**
`PVector.fromAngle`	Create a 2D unit vector with the **specified direction**

Instance method

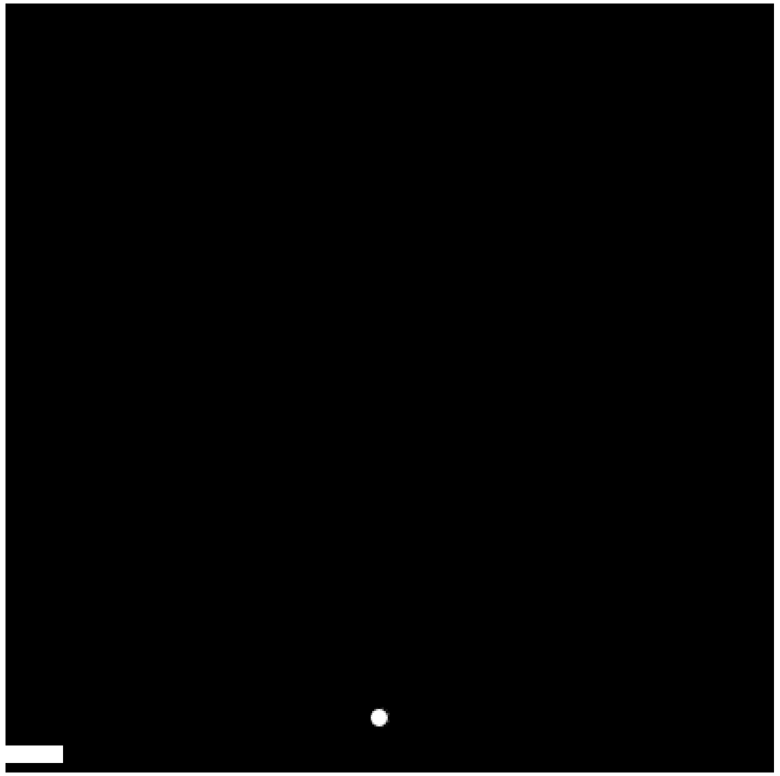
```
PVector v = new PVector(1, 0);
v.rotate(PI / 4);
println(v);
```

Static method

```
PVector v = PVector.fromAngle(PI / 4);
println(v);
```

Homework 2: Paddle Ball Game

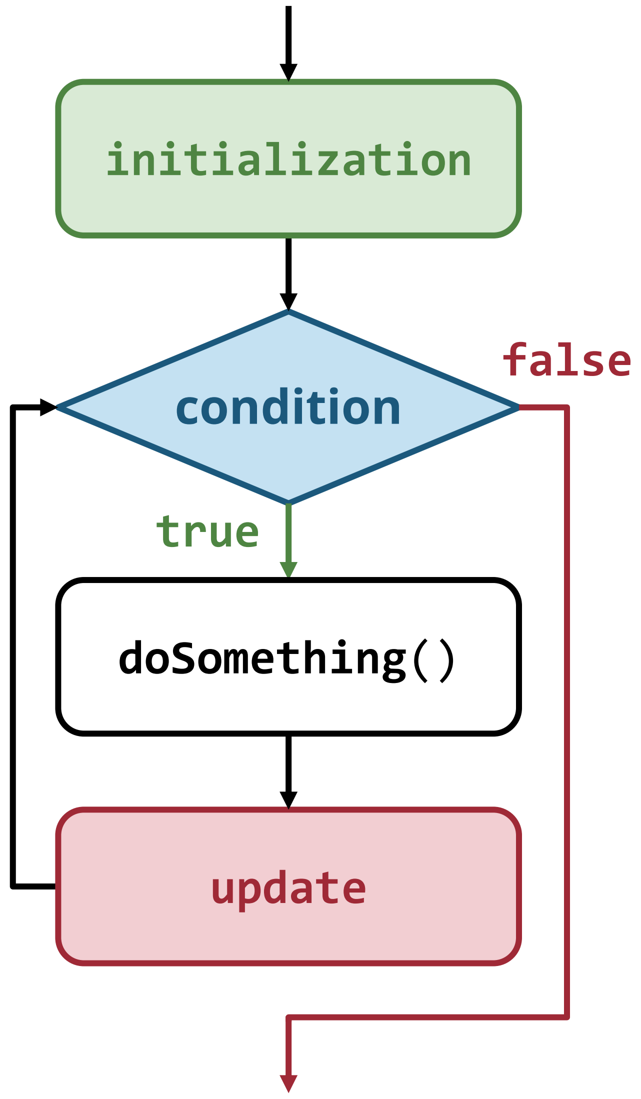
- Instructions will be released on Gradescope
- Features
  - Use the mouse to control the paddle bar
  - Show “GAME OVER!” when the paddle bar does not catch the ball
  - Click the mouse to restart the game
    - You’ll implement an `init()` function that will be called when the game starts or restarts
- Due at **11:59pm ET** on **September 13**
- Late submissions: **1 point deducted per day**



Review – Loops & Recursion

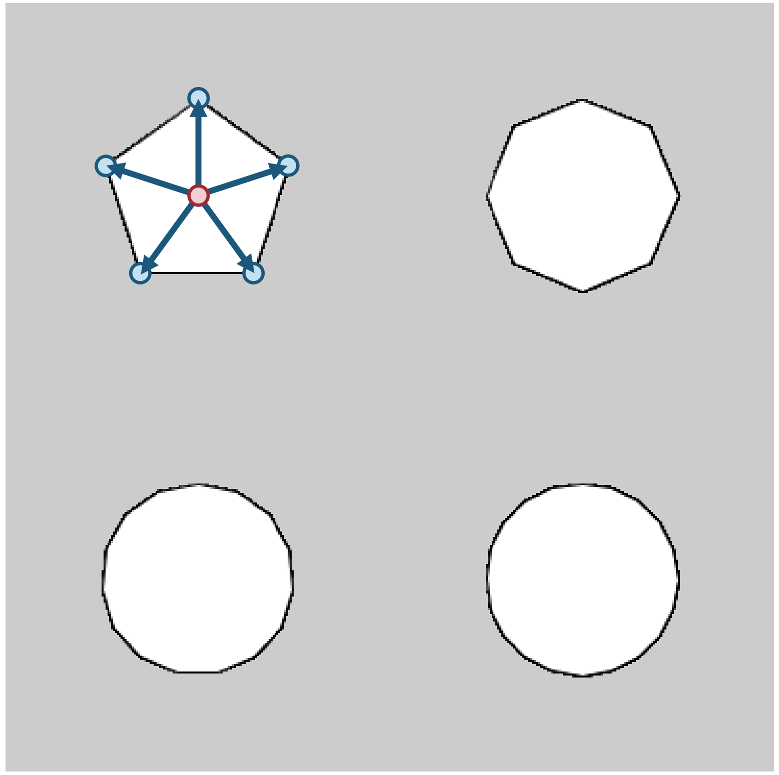
for Loop

```
for (initialization; condition; update) {
  doSomething();
}
```



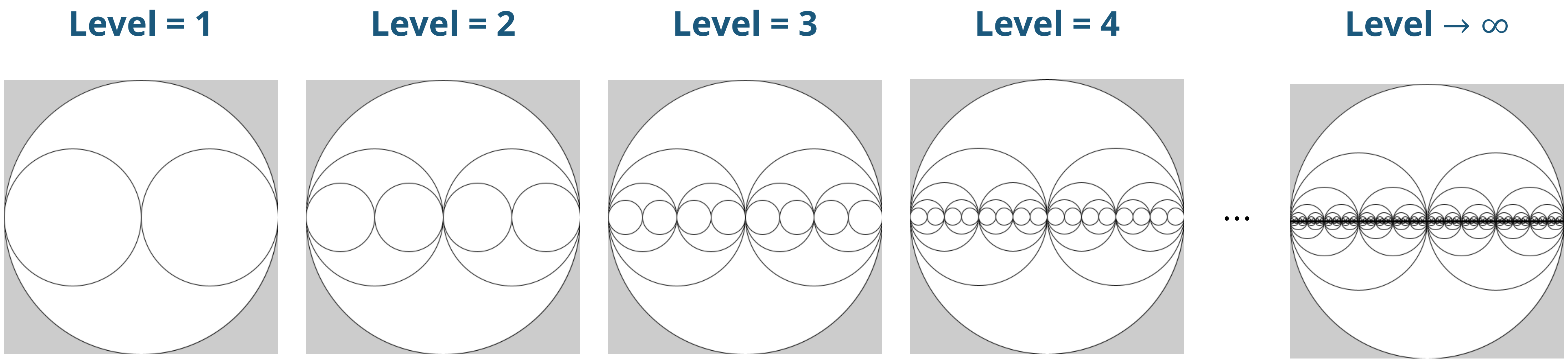
Exercise: Regular Polygons

```
void polygon(float x, float y, float radius, int n) {
  float vertexX, vertexY;
  beginShape();
  for (float a = 0; a < TWO_PI; a += TWO_PI / n) {
    vertexX = x + radius * cos(a - HALF_PI);
    vertexY = y + radius * sin(a - HALF_PI);
    vertex(vertexX, vertexY);
  }
  endShape(CLOSE);
}
```



Recursion

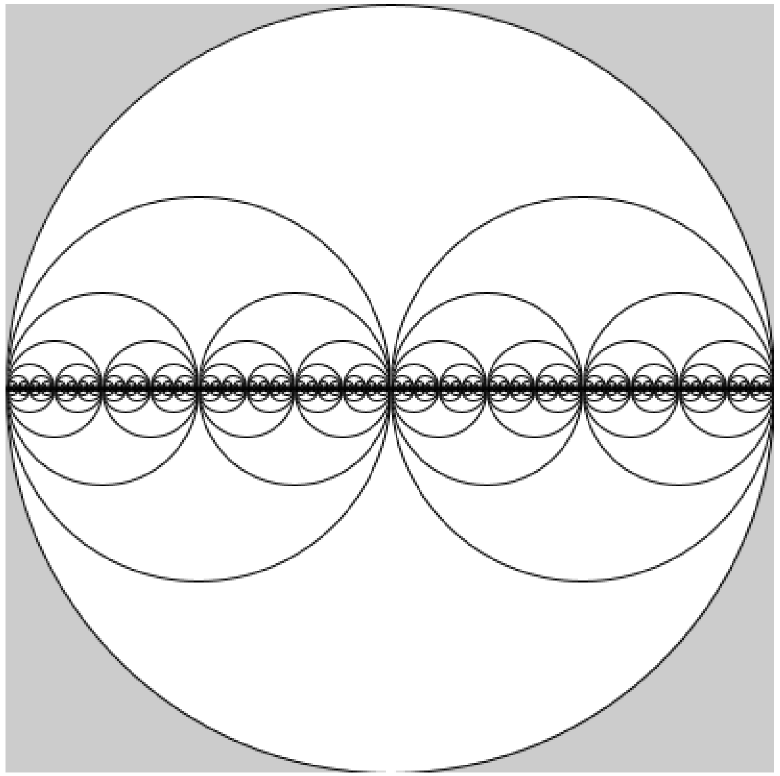
- Recursively calling a function



Example: Recursive Circles

```
void drawCircles(float x, float y, float w) {
  if (w < 1) return;
  circle(x - w / 4, y, w / 2);
  drawCircles(x - w / 4, y, w / 2);

  circle(x + w / 4, y, w / 2);
  drawCircles(x + w / 4, y, w / 2);
}

void draw() {
  circle(200, 200, 400);
  drawCircles(200, 200, w);
}
```

Stop condition



Review – Objects

Why Objects?

- **Organization**
  - Naturally organized into files or blocks
- **Re-usability**
  - A well-written class can be reused in many projects (e.g., FFT, PImage, PVector)
- **Ease of maintenance**
  - Each team member can work on different part of the code without less conflicts
- **Abstraction & Encapsulation**
  - What does FFT do internally? **Do we really need to know every detail?**
  - Define the interface rather than exposing everything

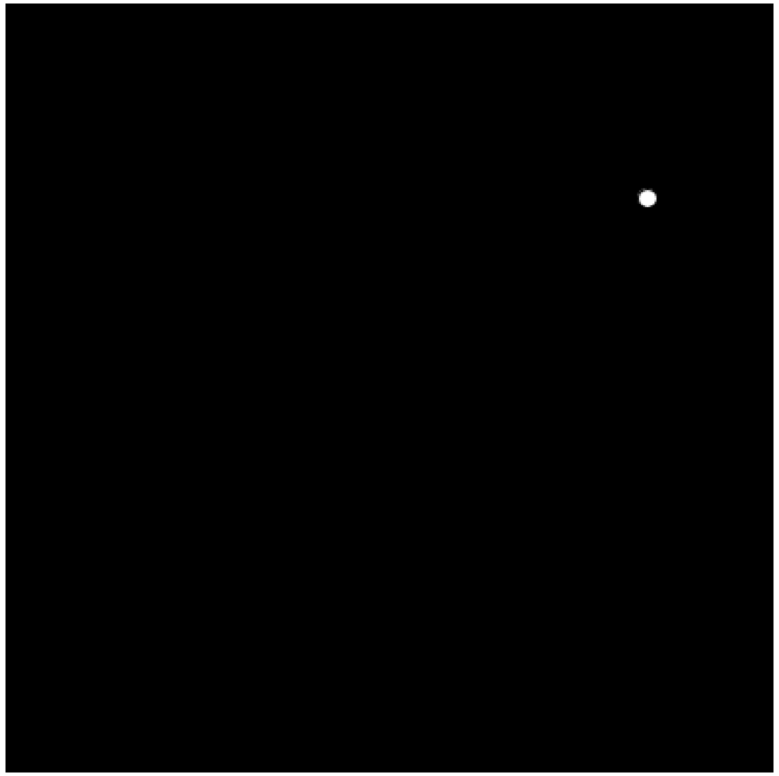
Example: Bouncing Ball

```
class Ball {
  float size = 10;
  float speed = 5;
  float x, y, speedX, speedY;

  Ball() {
    // Constructor
  }

  void show() {
    // Show the ball
  }

  void move() {
    // Move the ball
  }

  void checkWalls() {
    // Check if the ball hit the walls
  }
}
```

Fields

Constructor

Methods

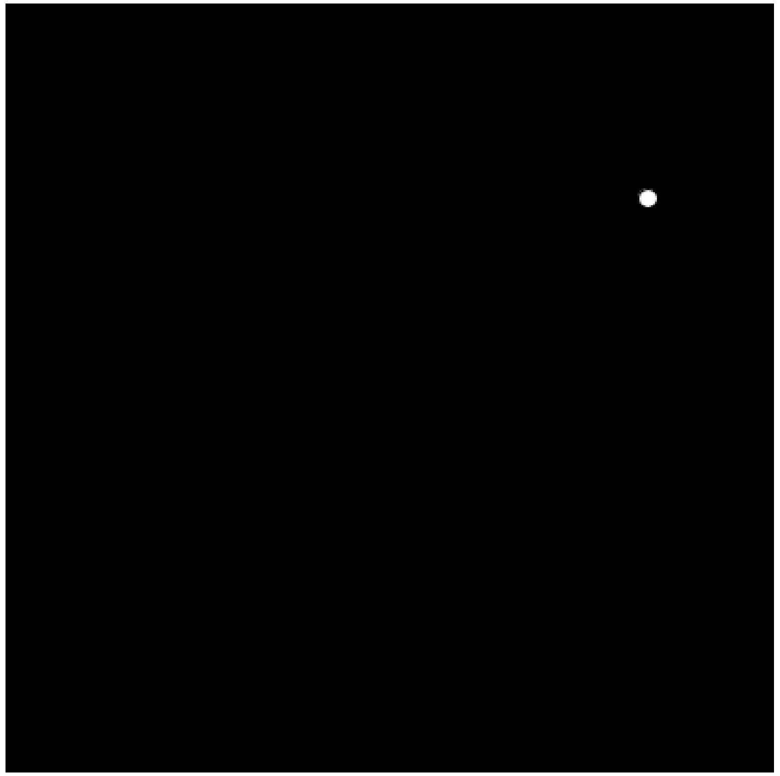


Example: Bouncing Ball

```
class Ball {
  ...

  void checkWalls() {
    float radius = size / 2;

    if (x > width - radius) {
      speedX = -abs(speedX);
    } else if (x < radius) {
      speedX = abs(speedX);
    }

    if (y > height - radius) {
      speedY = -abs(speedY);
    } else if (y < radius) {
      speedY = abs(speedY);
    }
  }

  ...
}
```

Check if the ball hit the left and right walls

Check if the ball hit the left and right walls



Example: Bouncing Ball

```
Ball ball;
```

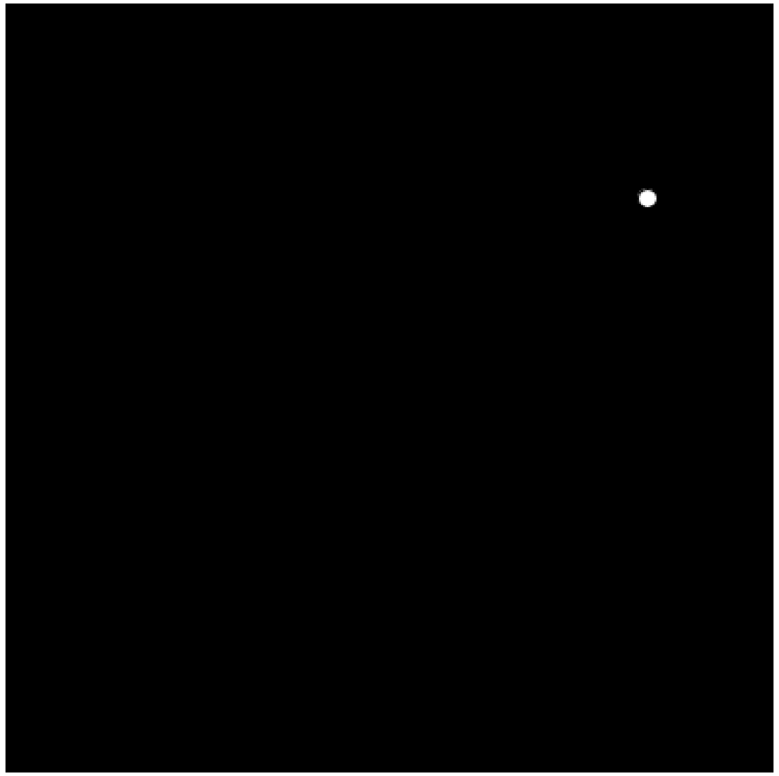
Declaration

```
void setup() {
  size(400, 400);

  ball = new Ball();
}
```

Initialization

```
void draw() {
  background(0);

  ball.move();
  ball.checkWalls();
  ball.show();
}
```

Call the methods!



Example: Bouncing Balls

```
Ball[] balls = new Ball[20];
```

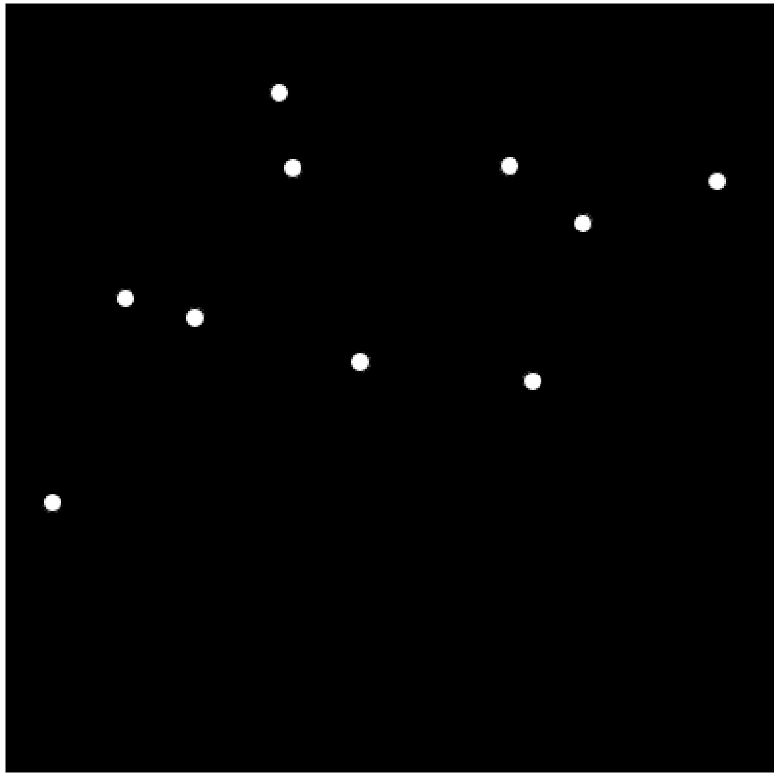
An array of objects

```
void setup() {
  size(400, 400);

  for (int i = 0; i < balls.length; i++) {
    balls[i] = new Ball();
  }
}
```

Initialization

```
void draw() {
  background(0);

  for (int i = 0; i < balls.length; i++) {
    balls[i].move();
    balls[i].checkWalls();
    balls[i].show();
  }
}
```

Call the methods!

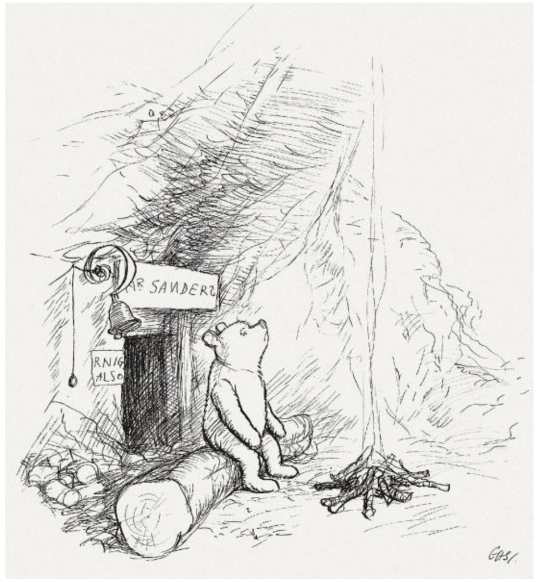


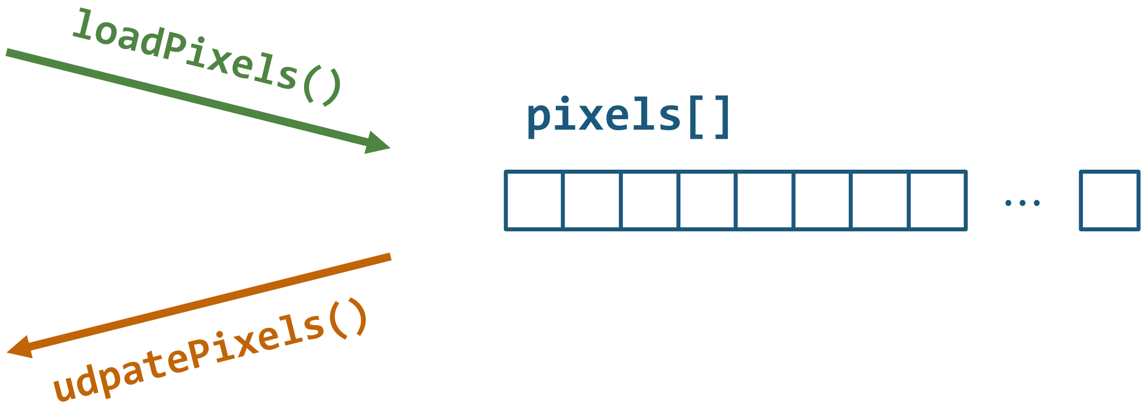
Review – Images

Loading the Pixels

- We can directly interact with the pixels of an image

`Image.pixels[]`	Array of all the pixels in the image
`Image.loadPixels()`	Load the image content to `Image.pixels[]`
`Image.updatePixels()`	Update the image content with `Image.pixels[]`

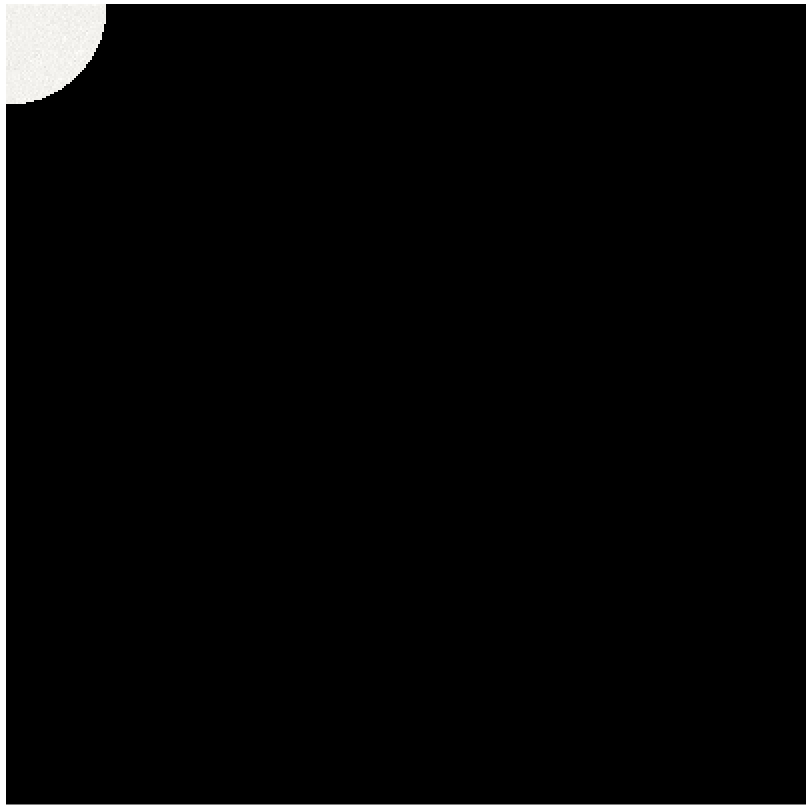




Exercise: The Reveal Effect

```
void setup() {
  size(400, 400);
  img = loadImage("pooh.jpg");
  image(img, 0, 0, 400, 400);
  loadPixels();
  org = pixels.clone();
  background(0);
  loadPixels();
}

void draw() {
  for (int x = 0; x < width; x++) {
    for (int y = 0; y < height; y++) {
      int loc = x + y * width;
      float d = dist(x, y, mouseX, mouseY);
      if (d < 50) {
        pixels[loc] = org[loc];
      }
    }
  }
  updatePixels();
}
```

Update the pixel values



Example: Pointillism

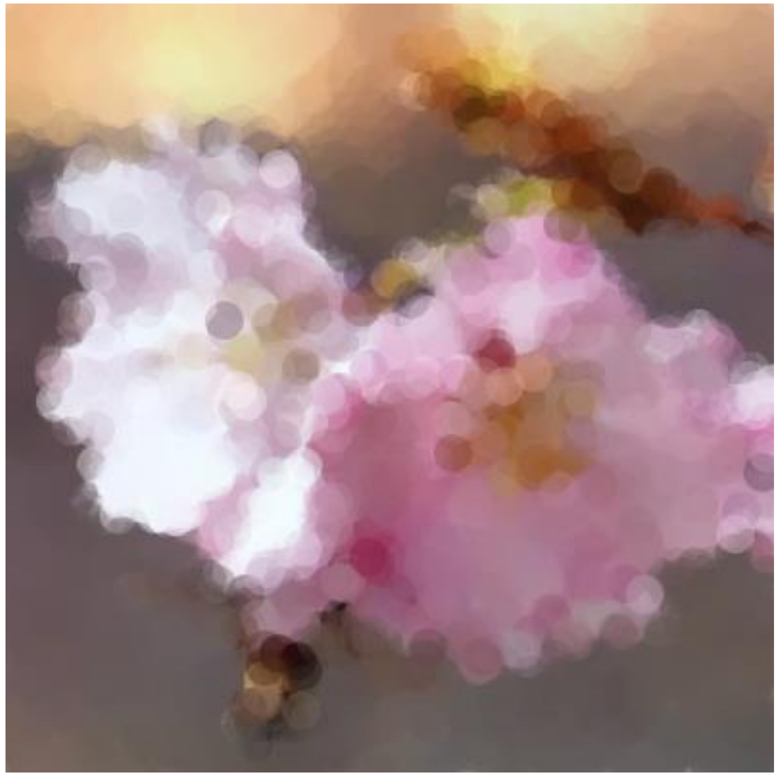
```
PImage img;

void setup() {
  size(400, 400);
  img = loadImage("sakura.jpg");
  background(255);
  noLoop();
}

void draw() {
  for (int i = 0; i < 10000; i++) {
    int x = int(random(img.width));
    int y = int(random(img.height));
    int loc = x + y * img.width;

    img.loadPixels();
    float r = red(img.pixels[loc]);
    float g = green(img.pixels[loc]);
    float b = blue(img.pixels[loc]);

    noStroke();
    fill(r, g, b, 100);
    circle(x, y, 20);
  }
}
```

Pick a random pixel

Find the color of the pixel

Set the color of the circle

Draw the circle



Review – Transformation

Transformations

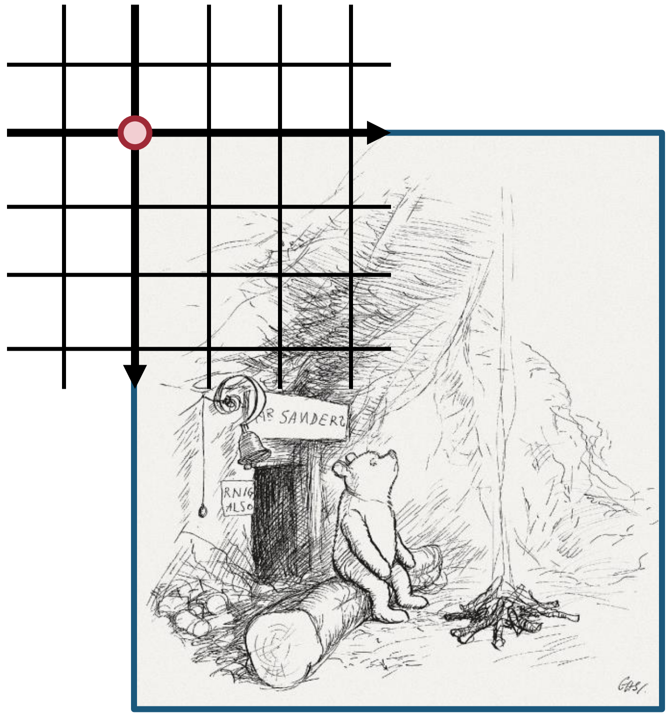
`translate(x, y)`	Translate the object
`rotate(angle)`	Rotate the object
`scale(s)`	Scale the object
`scale(x, y)`	Scale the object

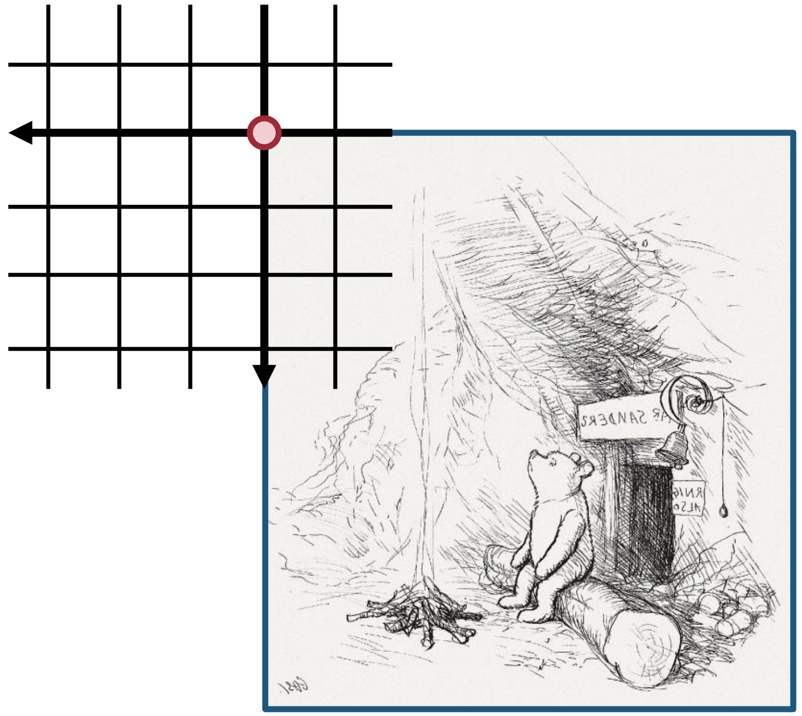
Example: Mirroring Capture

```
void draw() {
  image(video, 0, 0);
}
```

→

```
void draw() {
  scale(-1, 1);
  image(video, -video.width, 0);
}
```



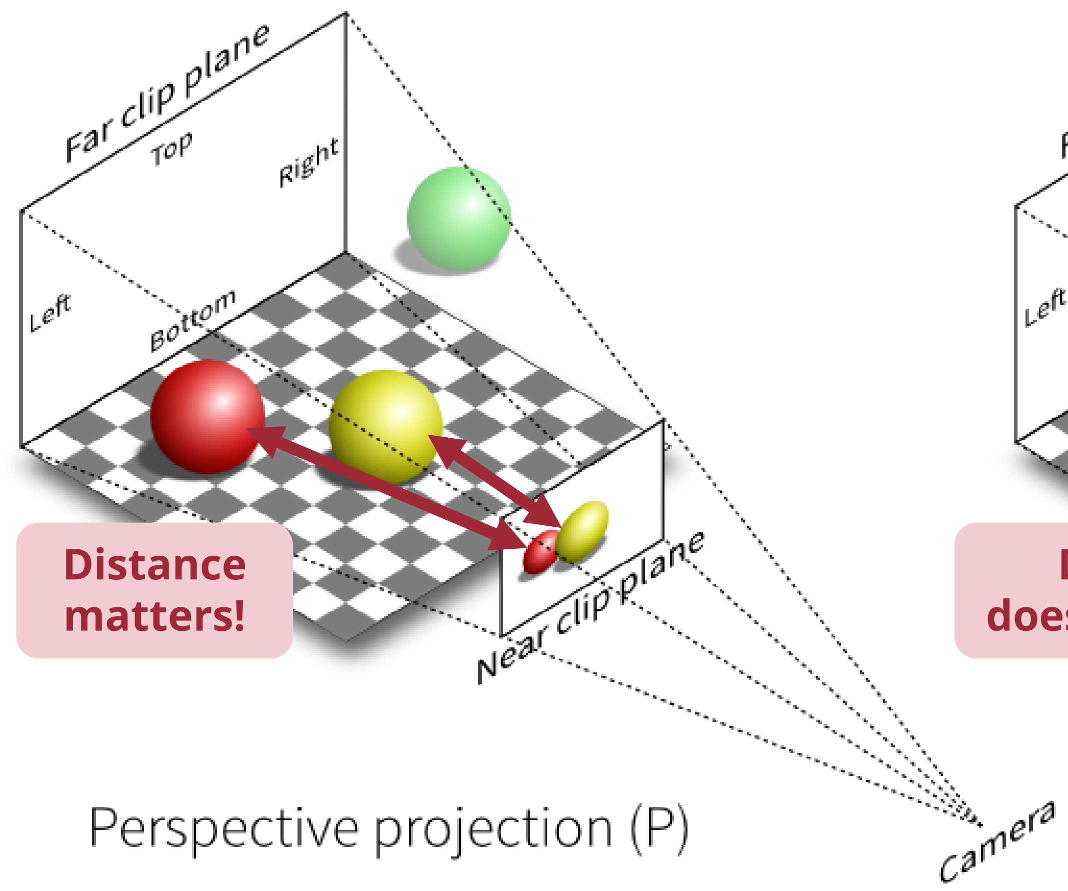


Review – 3D Graphics

Perspective vs Orthographic Projections

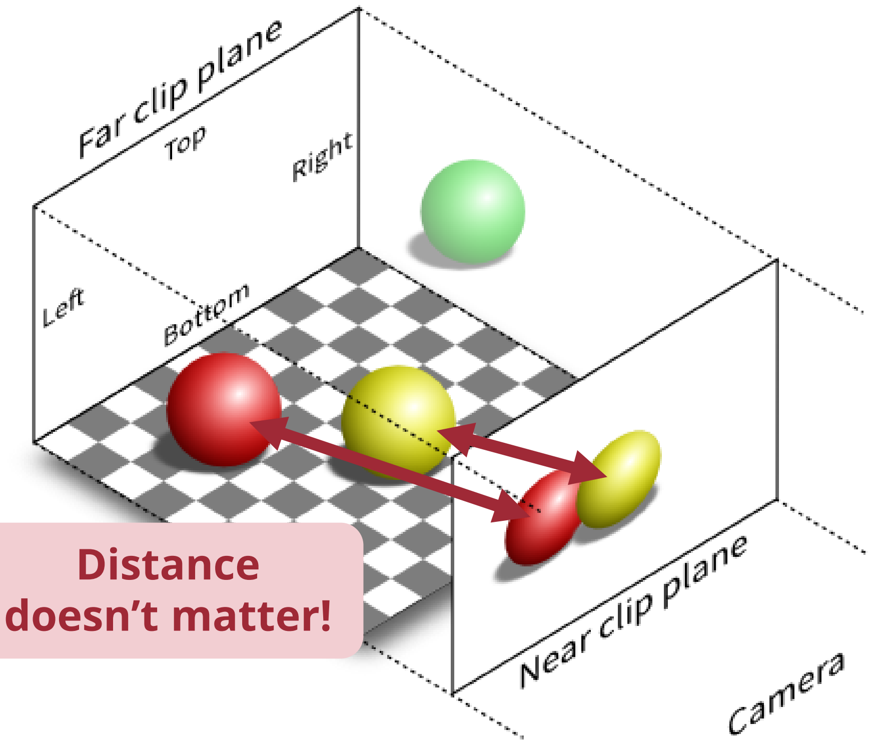
perspective()

ortho()



Distance matters!

Perspective projection (P)

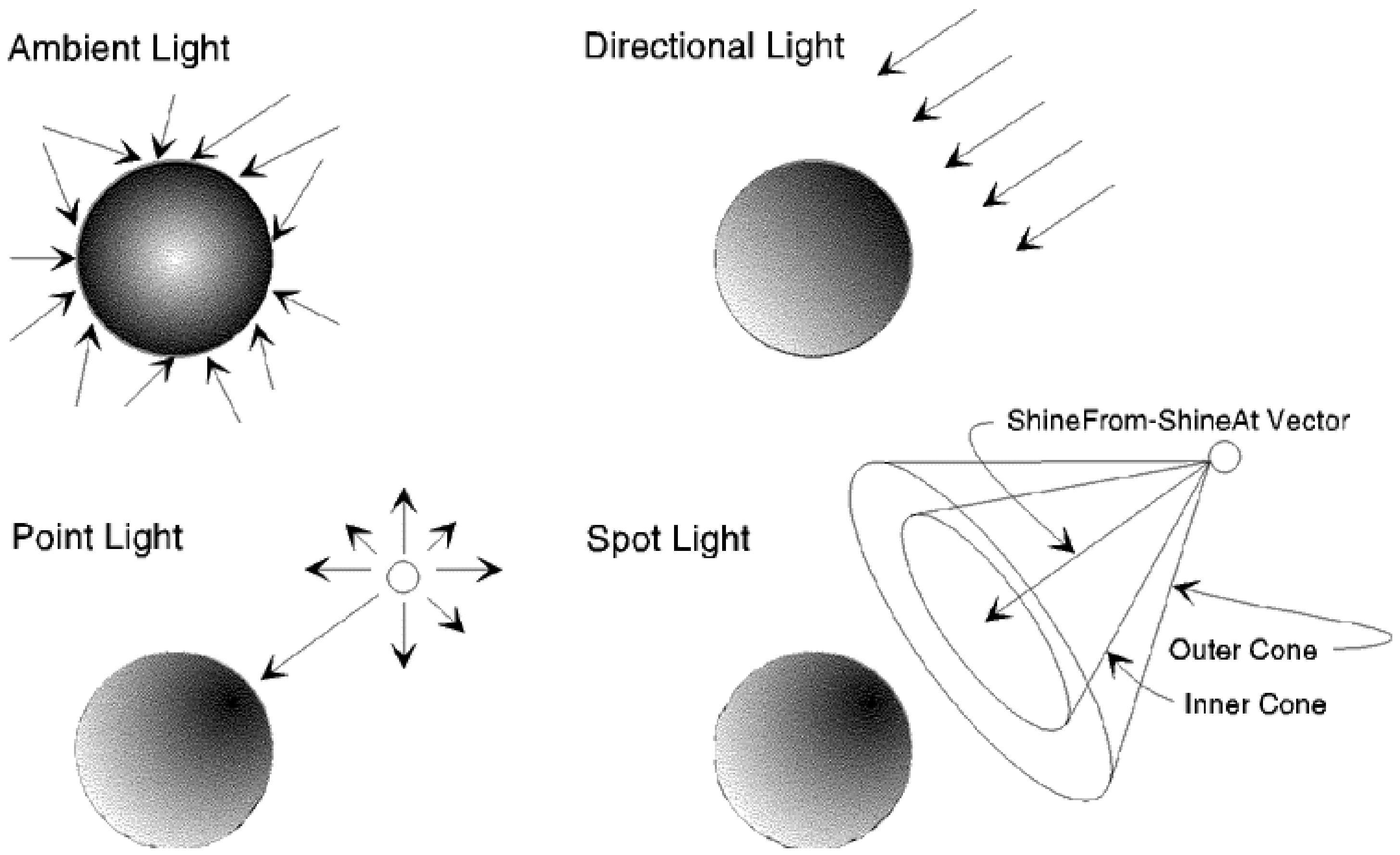


Distance doesn't matter!

Orthographic projection (O)

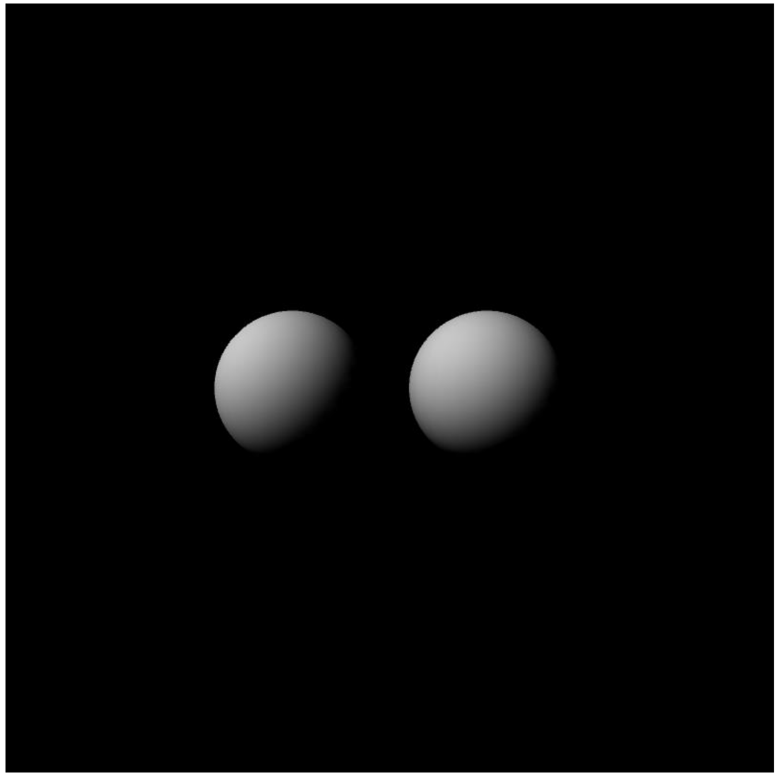
Lights

- `ambientLight()`
- `directionalLight()`
- `spotlight()`
- `pointLight()`



Example: Creepy Eyes 3D

```
void setup() {
  size(800, 800, P3D);
}

void draw() {
  background(0);

  float dirX = (mouseX - width / 2) / (width / 2.0);
  float dirY = (mouseY - height / 2) / (height / 2.0);
  directionalLight(200, 200, 200, -dirX, -dirY, -1);

  fill(255);
  noStroke();
  translate(300, 400, 0);
  sphere(80);
  translate(200, 0, 0);
  sphere(80);
}
```



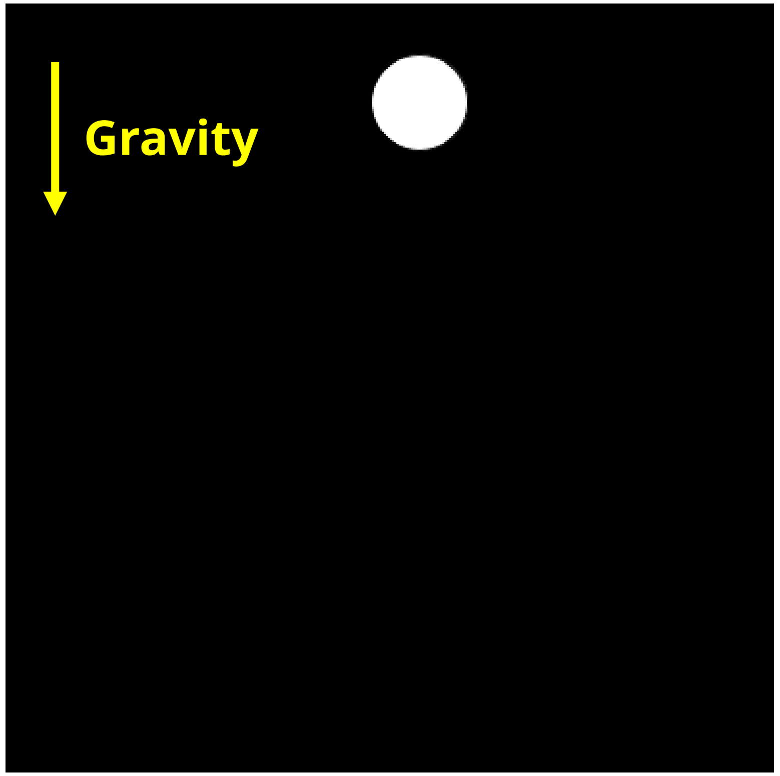
Review – Motion & Physics

Example: Gravity

```
// Apply gravity to the ball
void applyGravity() {
  speedY += gravity;
}

// Check if the ball hit the walls
void checkWalls() {
  ...

  // Check if the ball hit the top and bottom walls
  if (y > height - radius) {
    speedY = -abs(speedY) * decay;
    y = height - radius;
  } else if (y < radius) {
    speedY = abs(speedY);
    y = radius;
  }
}
```

Apply gravity as y-acceleration

Reduce the speed a little bit when it hits the bottom wall



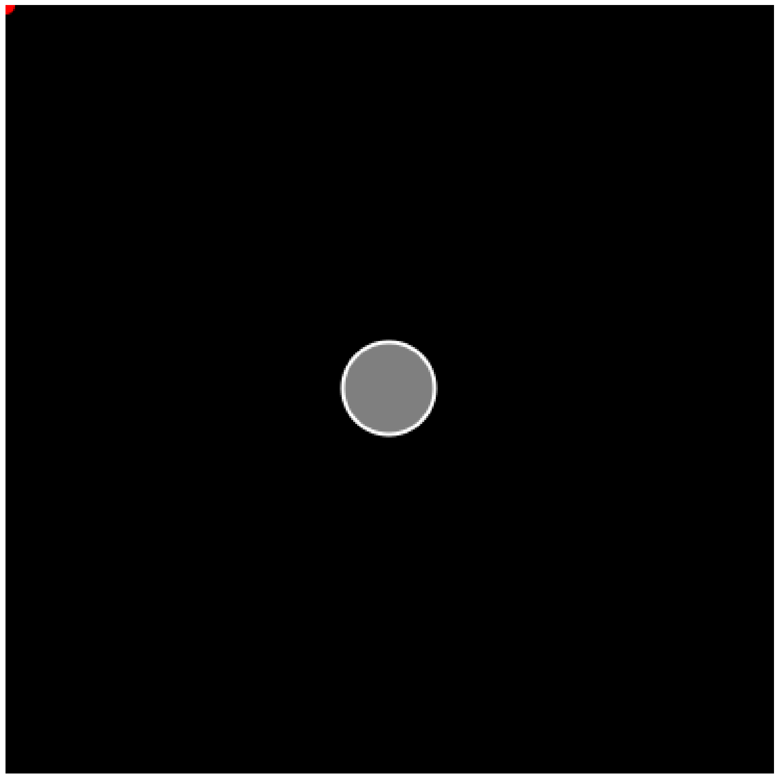
Example: Acceleration

```
class Mover {
  PVector location;
  PVector velocity;
  PVector acceleration;
  float topspeed = 5;

  ...

  void update() {
    PVector mouse = new PVector(mouseX, mouseY);
    PVector acceleration = PVector.sub(mouse, location);
    acceleration.setMag(0.2);

    velocity.add(acceleration);
    velocity.limit(topspeed);

    location.add(velocity);
  }
}
```

Calculate acceleration

Apply the acceleration

Move the ball



Example: Bouncing Balls with Collision Detection

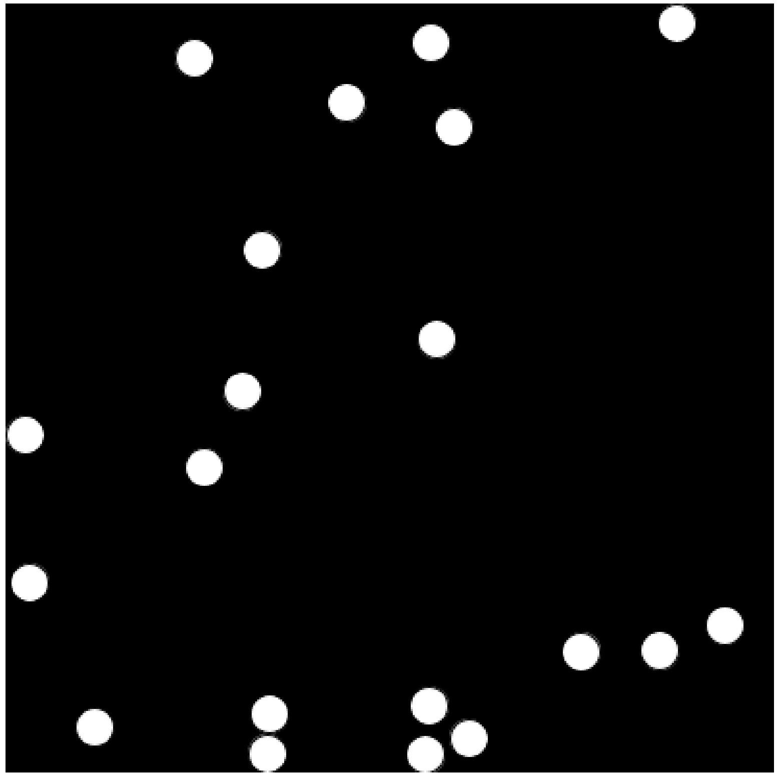
```
void collide(Ball other) {
  if (other == this) return;

  float dist = dist(x, y, other.x, other.y);

  if (dist >= size) return;

  x -= speedX;
  y -= speedY;

  float theta = atan2(other.y - y, other.x - x);
  float orgAngle = atan2(speedY, speedX);
  float newAngle = (theta - PI + theta - orgAngle);
  speedX = speed * cos(newAngle);
  speedY = speed * sin(newAngle);
}
```

Do nothing if it's the same ball

Do nothing if they do not collide

Revert the ball back to where it was before the collision

Find the velocity after the collision



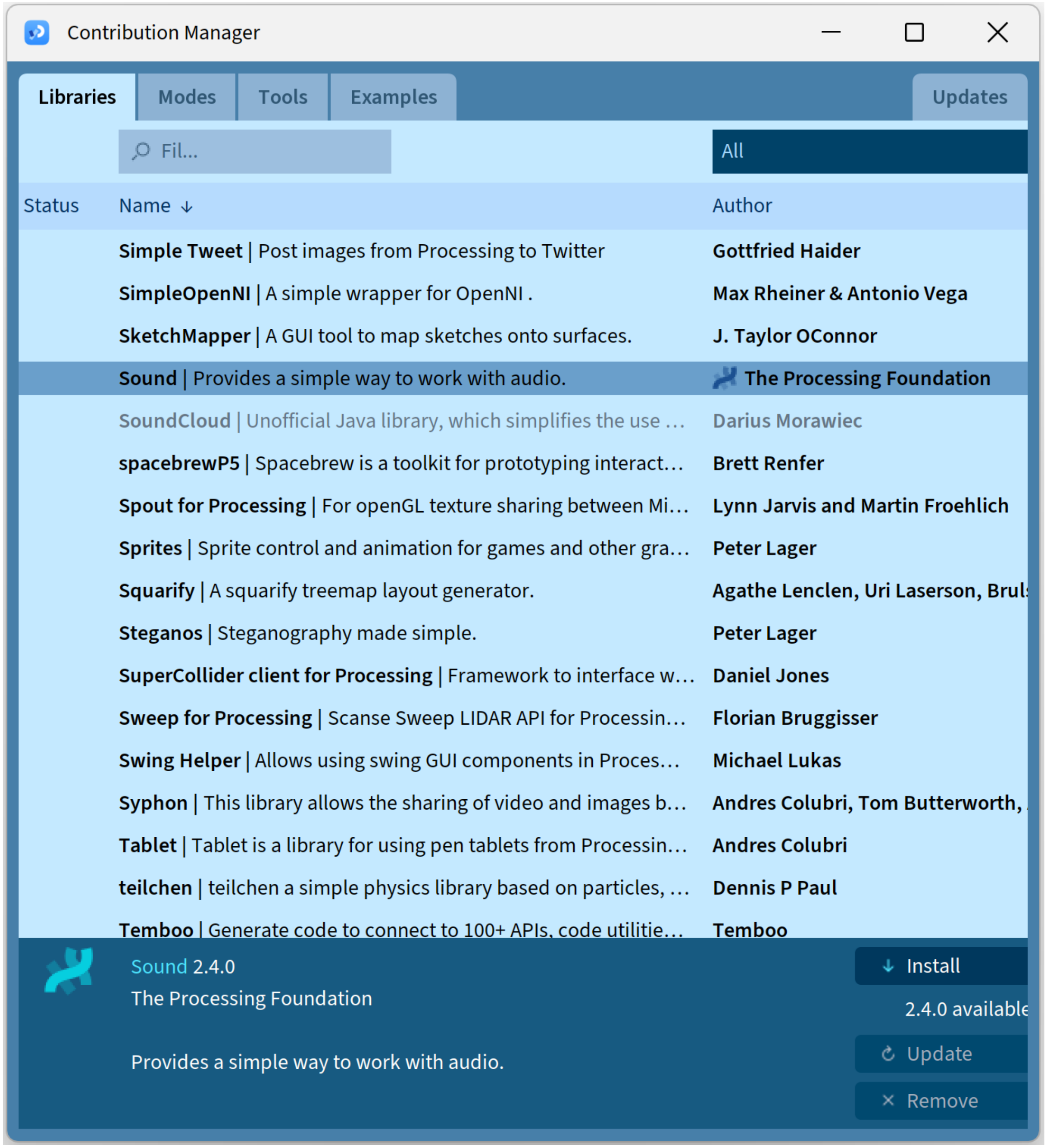
Example: Fireworks

- A **`Firework`** object
  - Starts as one single **`Particle`** object
    - Initialized with random force up
    - Flies up with gravity slowing it down
    - Explodes when speed reaches zero
  - Becomes many **`Particle`** objects after explosion
    - Initialized with random forces towards random directions
    - Fall with gravity
    - Die after invisible on the canvas

Review – Audio

Library Manager

- Official Libraries maintained by the **Processing Foundation**
  - Sound
  - Video
  - Hardware I/O
  - JavaFX
- Many other libraries
  - Networking
  - GUI
  - Animation



Status	Name ↓	Author
	Simple Tweet | Post images from Processing to Twitter	**Gottfried Haider**
	SimpleOpenNI | A simple wrapper for OpenNI .	**Max Rheiner & Antonio Vega**
	SketchMapper | A GUI tool to map sketches onto surfaces.	**J. Taylor OConnor**
	Sound | Provides a simple way to work with audio.	**The Processing Foundation**
	SoundCloud | Unofficial Java library, which simplifies the use …	**Darius Morawiec**
	spacebrewP5 | Spacebrew is a toolkit for prototyping interact…	**Brett Renfer**
	Spout for Processing | For openGL texture sharing between Mi…	**Lynn Jarvis and Martin Froehlich**
	Sprites | Sprite control and animation for games and other gra…	**Peter Lager**
	Squarify | A squarify treemap layout generator.	**Agathe Lenclen, Uri Laserson, Brul**
	Steganos | Steganography made simple.	**Peter Lager**
	SuperCollider client for Processing | Framework to interface w…	**Daniel Jones**
	Sweep for Processing | Scanse Sweep LIDAR API for Processin…	**Florian Bruggisser**
	Swing Helper | Allows using swing GUI components in Proces…	**Michael Lukas**
	Syphon | This library allows the sharing of video and images b…	**Andres Colubri, Tom Butterworth,**
	Tablet | Tablet is a library for using pen tablets from Processin…	**Andres Colubri**
	teilchen | teilchen a simple physics library based on particles, …	**Dennis P Paul**
	Temboo | Generate code to connect to 100+ APIs, code utilitie…	**Temboo**

Sound 2.4.0
The Processing Foundation

Provides a simple way to work with audio.

(Recap) Amplitude Class

```
import processing.sound.*;

Amplitude amp = new Amplitude(this);
AudioIn in = new AudioIn(this, 0);
float a;

void setup() {
  size(400, 400);

  in.start();
  amp.input(in);
}

void draw() {
  background(0);
  a = amp.analyze();
  circle(200, 200, a * 400);
}
```

Initialize an Amplitude object

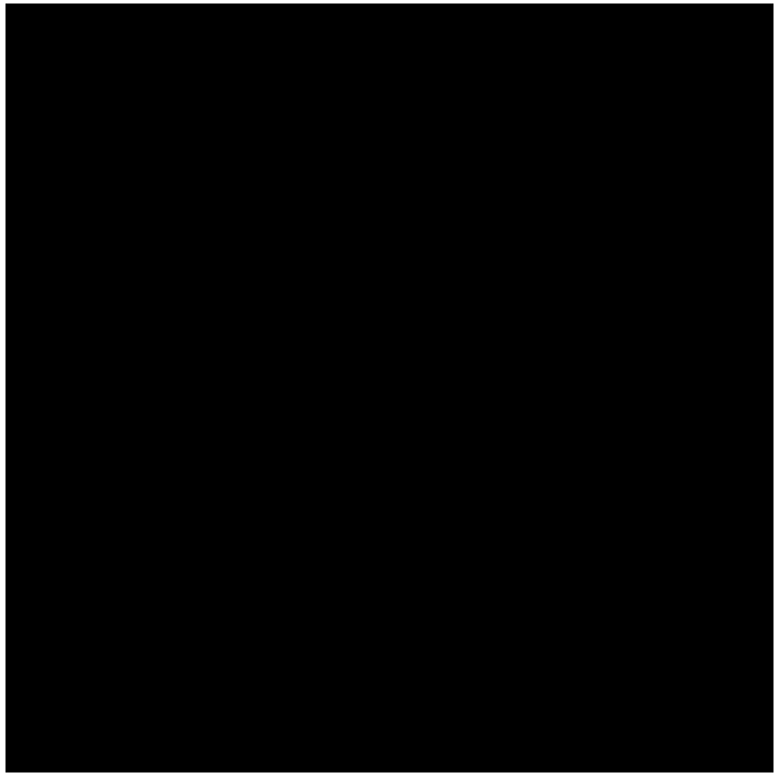
Initialize an AudioIn object

Start taking audio input

Route the audio input to the amplitude meter

Measure the amplitude

Normalized to [0, 1]



FFT Class

```
import processing.sound.*;

int bands = 512;
FFT fft = new FFT(this, bands);
AudioIn in = new AudioIn(this, 0);
float[] spectrum = new float[bands];

void setup() {
  size(512, 360);

  in.start();
  fft.input(in);
}

void draw() {
  background(255);

  fft.analyze(spectrum);

  for(int i = 0; i < bands; i++){
    line(i, height, i, height - spectrum[i] * height * 5);
  }
}
```

Import the Sound library

Initialize an FFT object

Initialize an AudioIn object

Initialize an array to store the spectrum

Start taking audio input

Route the audio input to the FFT analyzer

Specify the array to store the outputs

Run Fast Fourier Transform

Normalized to [0, 1]

Homework 3: Spectrum Visualizer

- Modify the template code to implement a spectrum visualizer
- Instructions will be released on Gradescope
- Due at **11:59pm ET** on **September 23**
- Late submissions: **1 point deducted per day**

Review – Extensions

Processing on Different Platforms

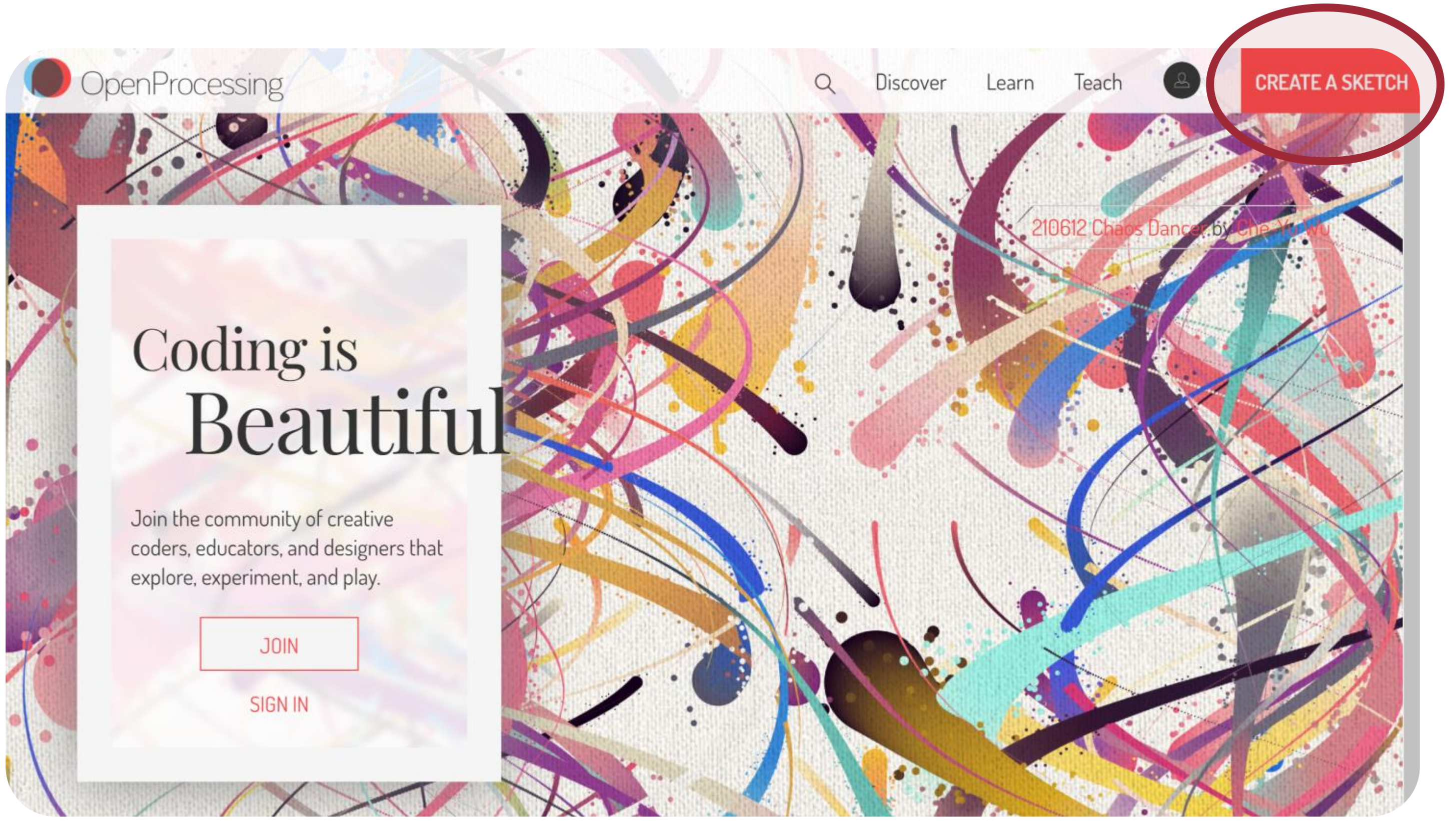
• JavaScript	p5.js	p5js.org
• Python	processing.py	py.processing.org
• Android		android.processing.org
• Raspberry Pi		pi.processing.org

p5.js – Processing for Java Script

- Ideal for web programming
- Can be embedded on websites
- Slightly different syntax but same design philosophy
- Online editor: editor.p5js.org

OpenProcessing

- Large community of creative coders, educators and designers!

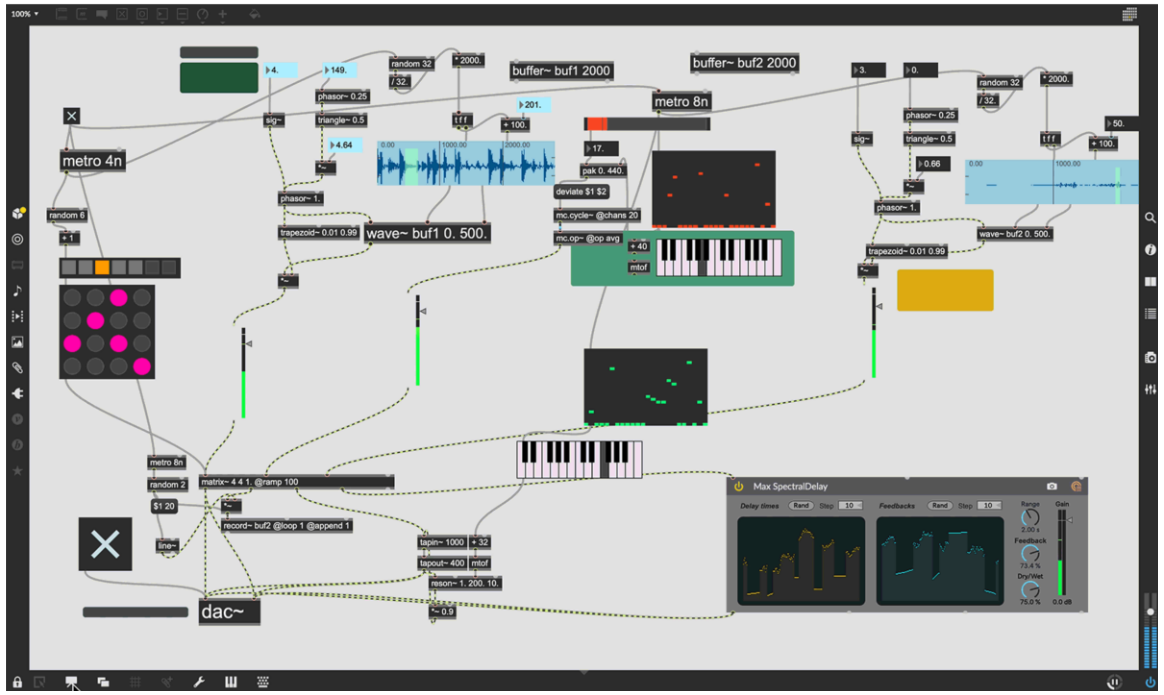

OpenProcessing
Discover
Learn
Teach
CREATE A SKETCH
210612 Chaos Dance by Che-Yu Wu
Coding is Beautiful
Join the community of creative coders, educators, and designers that explore, experiment, and play.
JOIN
SIGN IN

openprocessing.org

Max

Review – Max Basics

What is Max?

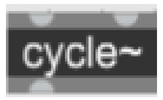


(Source: cycling74.com/products/max)

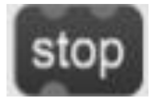
(Source: Pete Brown from Gambrills, MD, USA via Wikimedia)

Basic Max Components

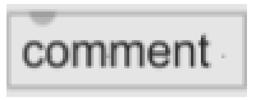
Object (**n**)



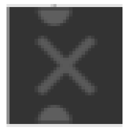
Message (**m**)



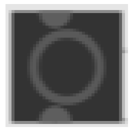
Comment (**c**)



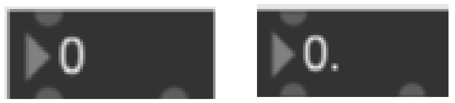
Toggle (**t**)



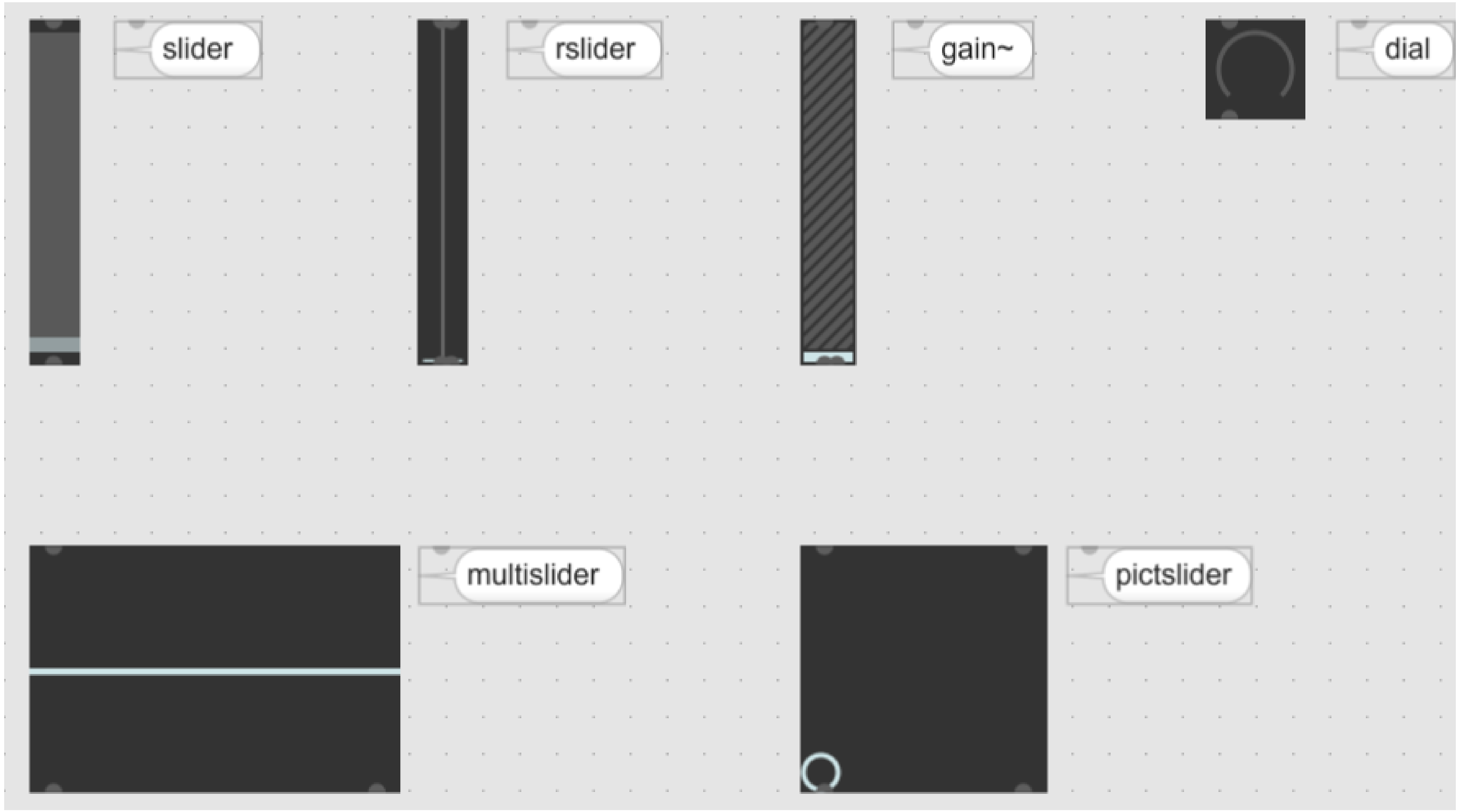
Button (**b**)



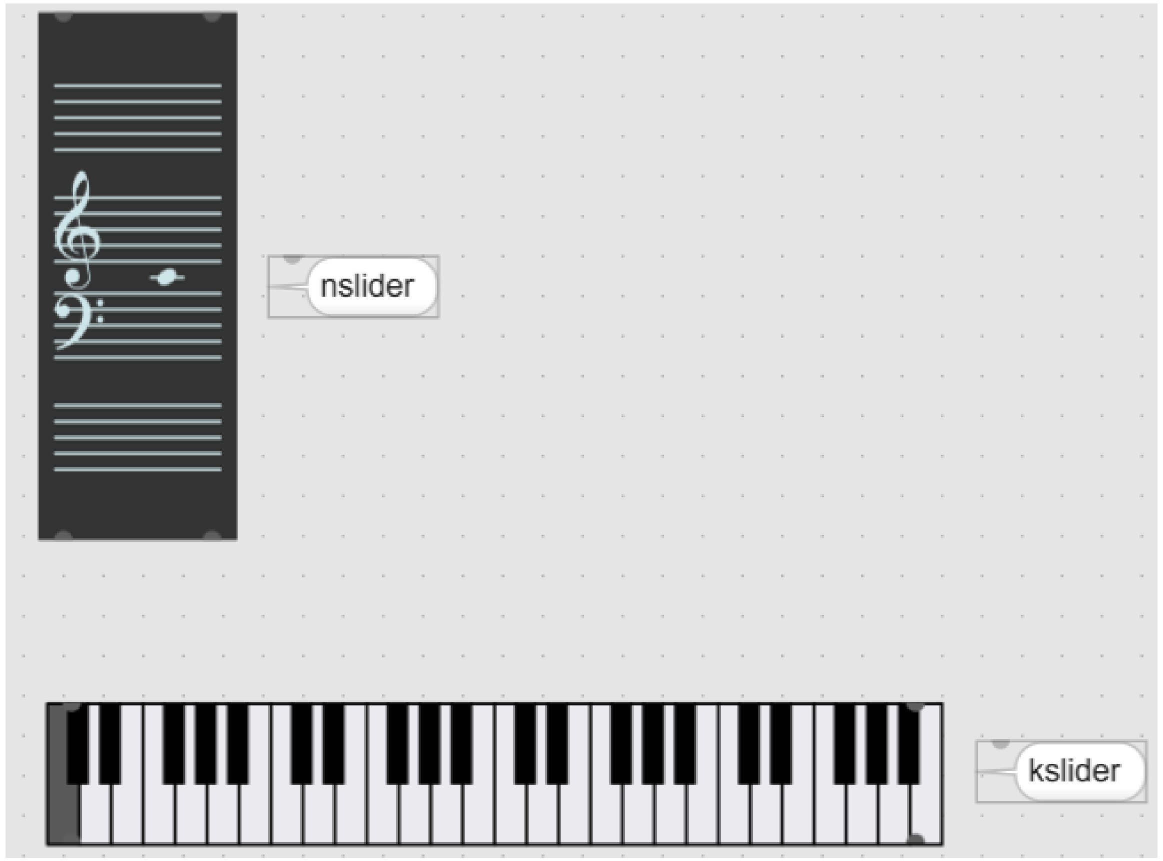
Number (**i**, **f**)



Sliders

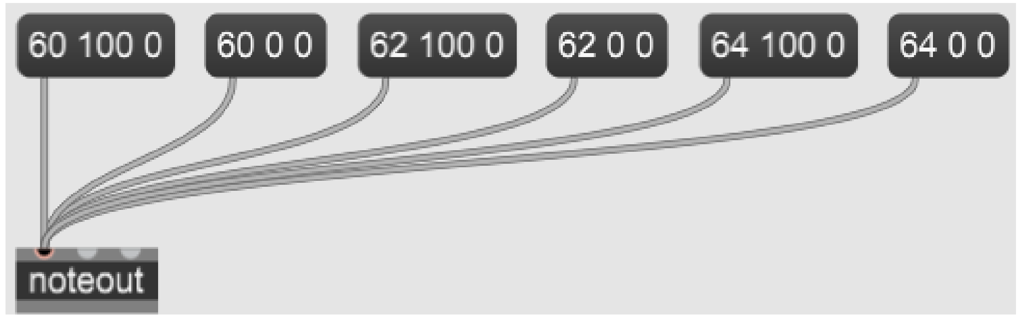


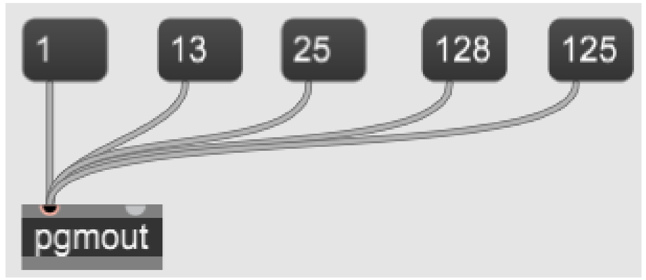
Advanced Sliders

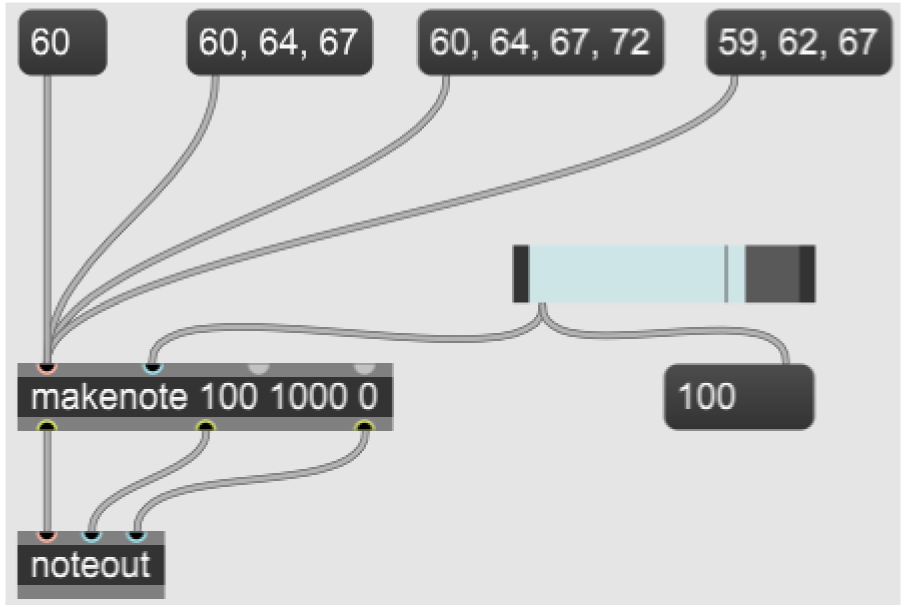


Review – MIDI

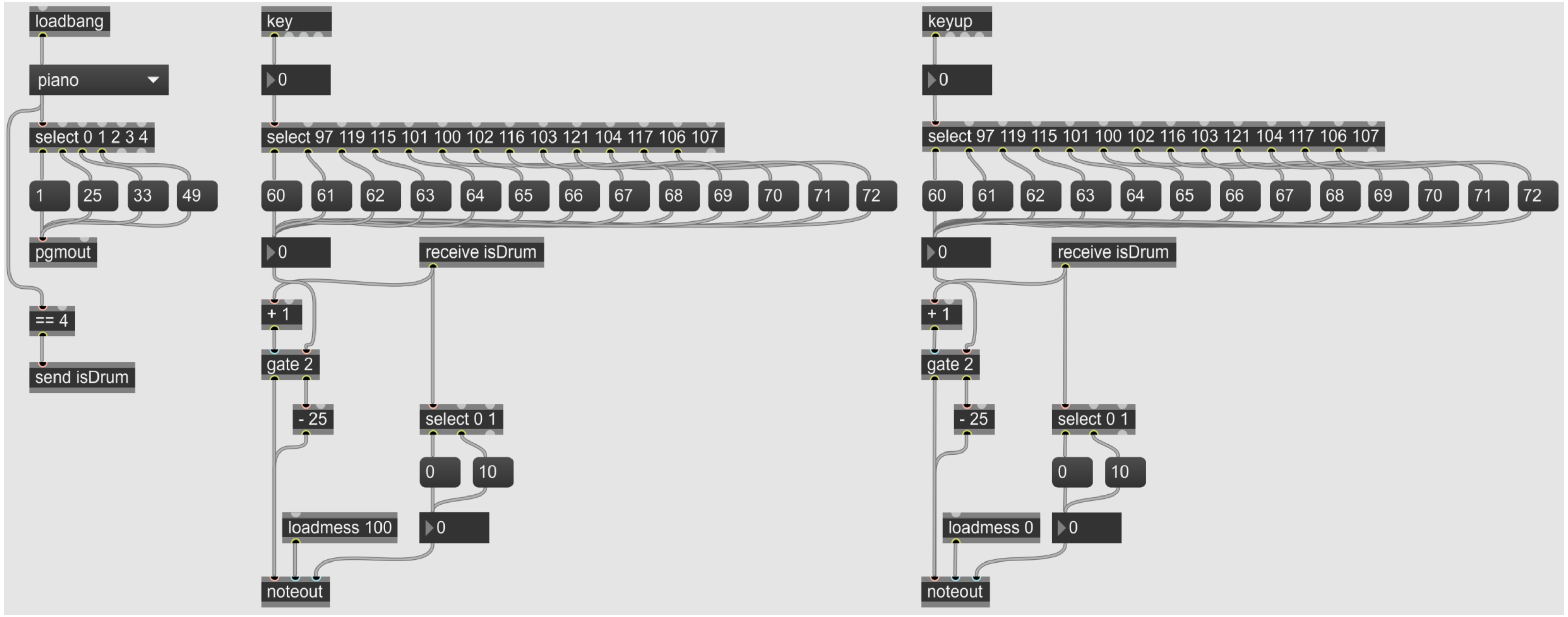
MIDI


60 100 0
60 0 0
62 100 0
62 0 0
64 100 0
64 0 0
noteout

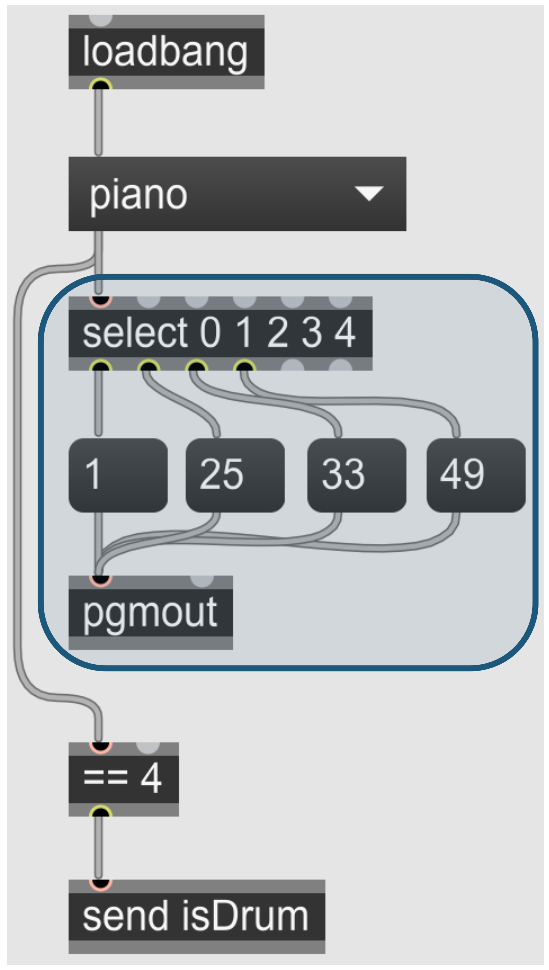

1
13
25
128
125
pgmout


60
60, 64, 67
60, 64, 67, 72
59, 62, 67
makenote 100 1000 0
100
noteout

Homework 4: MIDI Keyboard

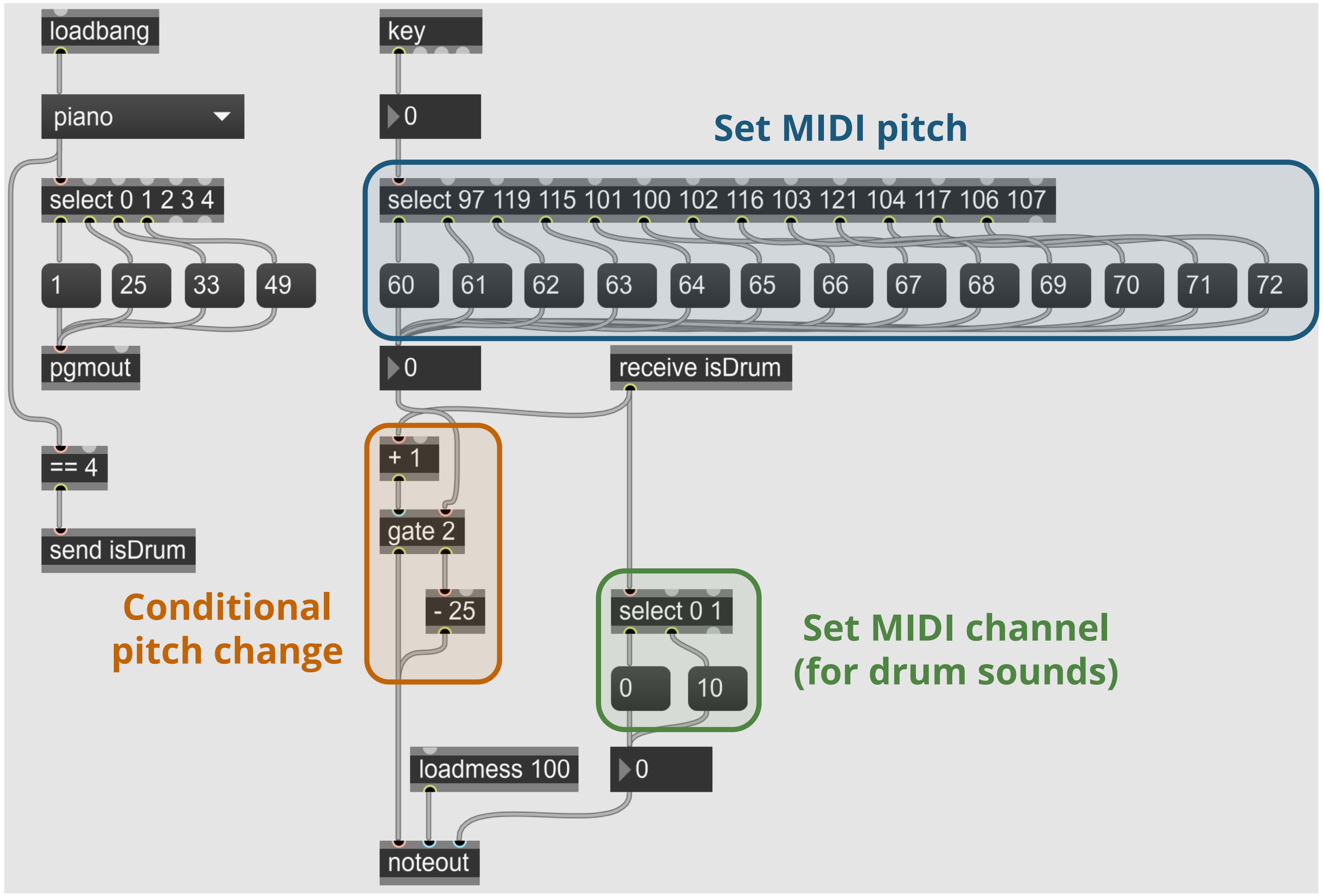


Homework 4: MIDI Keyboard



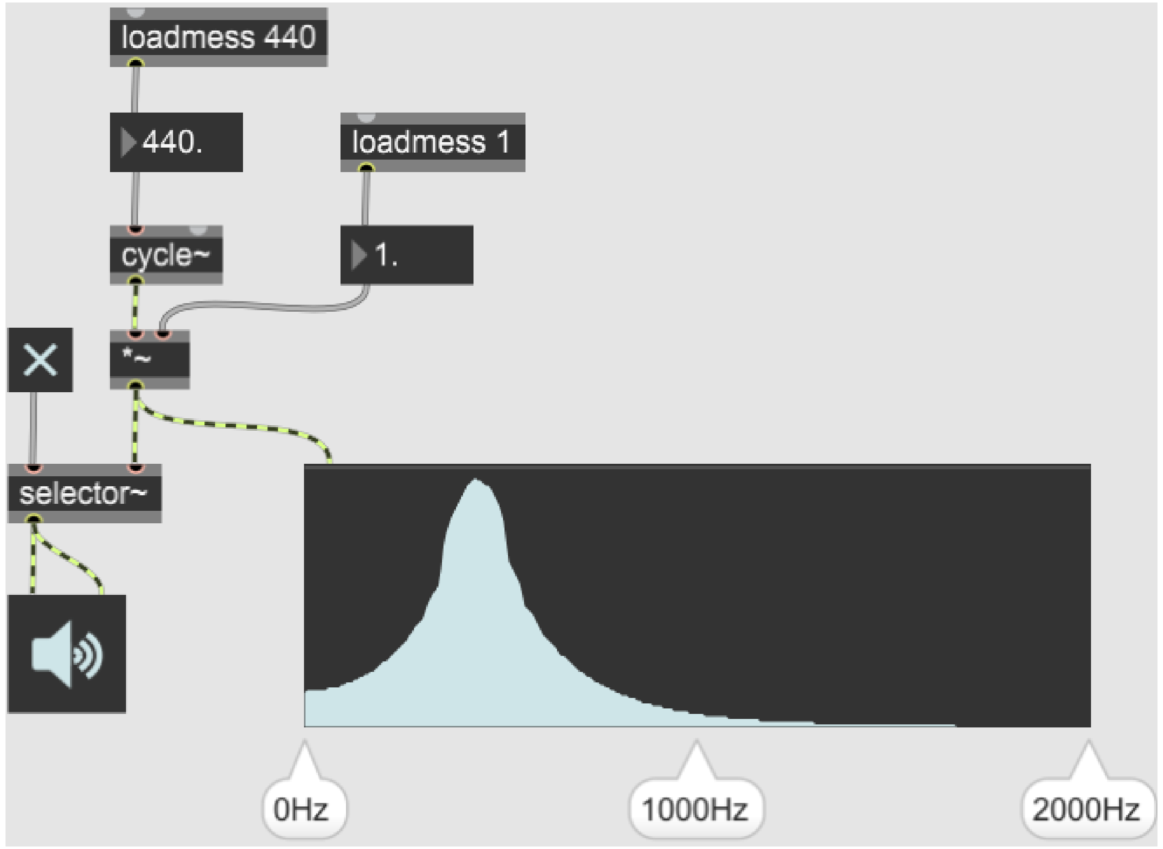
Set MIDI program

Homework 4: MIDI Keyboard

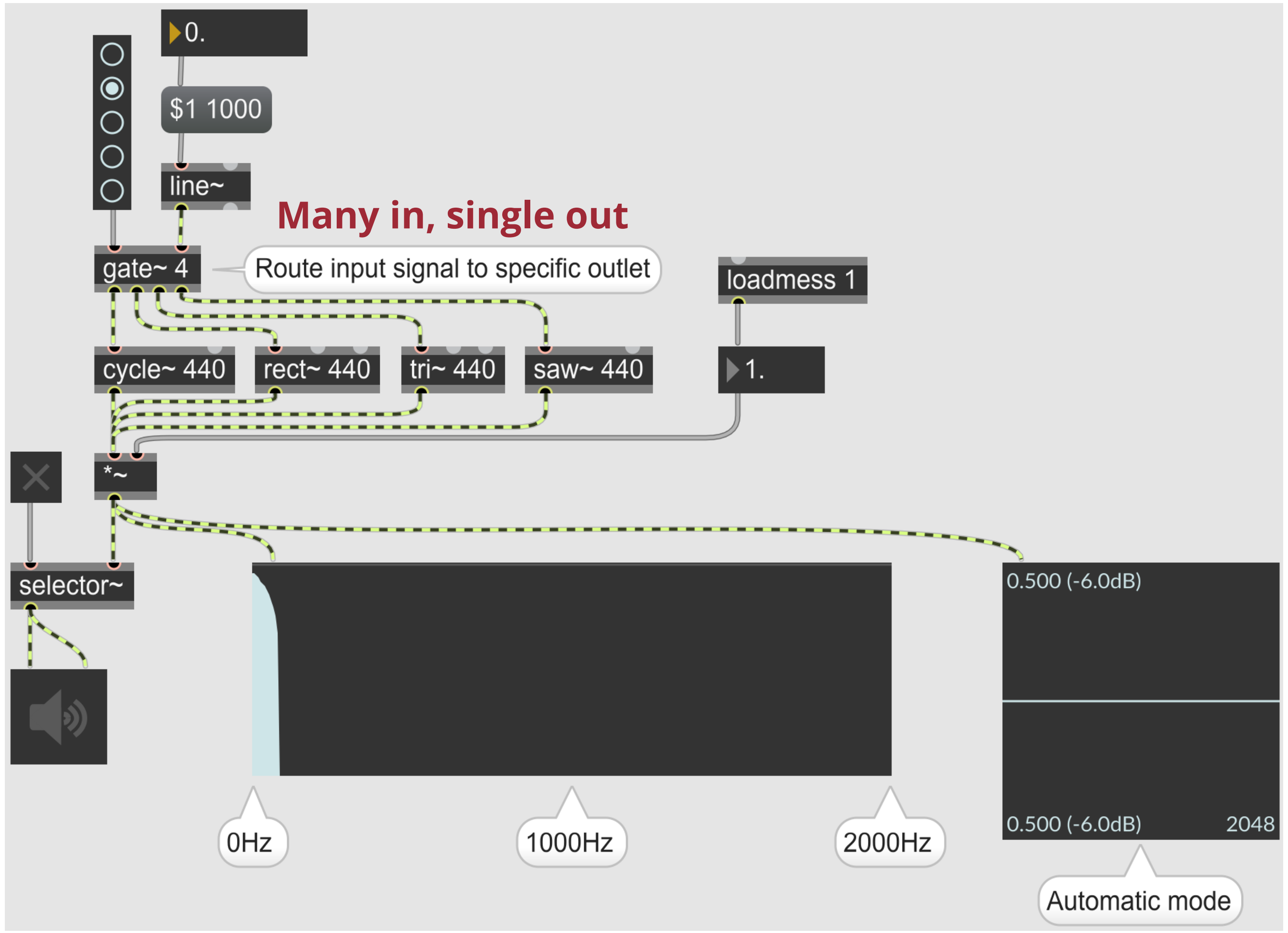


Review – Signals

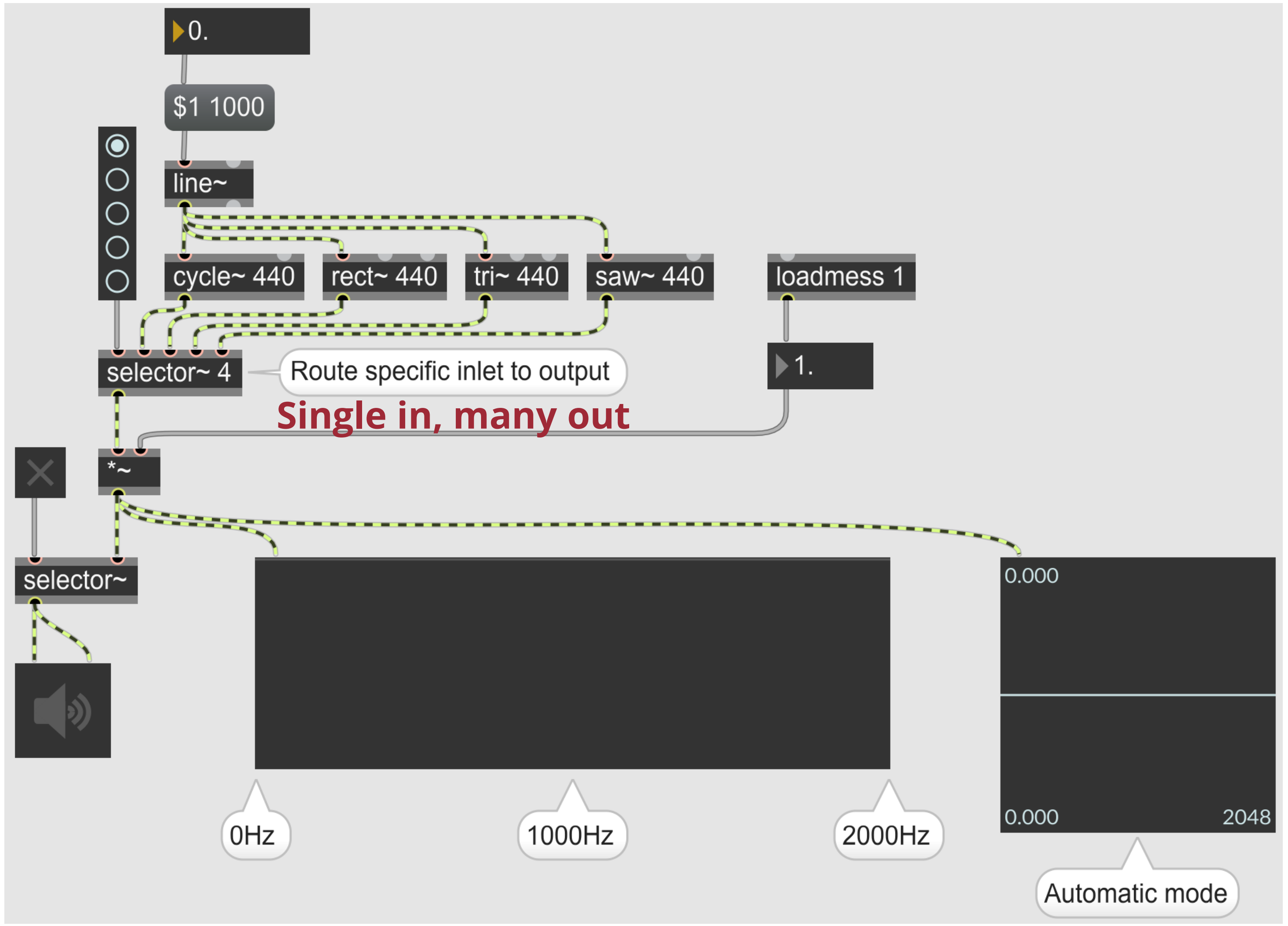
Sinusoid Oscillator



gate~


0.
$1 1000
line~
Many in, single out
gate~ 4
Route input signal to specific outlet
loadmess 1
cycle~ 440
rect~ 440
tri~ 440
saw~ 440
1.
*~
selector~
0.500 (-6.0dB)
0.500 (-6.0dB)
2048
0Hz
1000Hz
2000Hz
Automatic mode

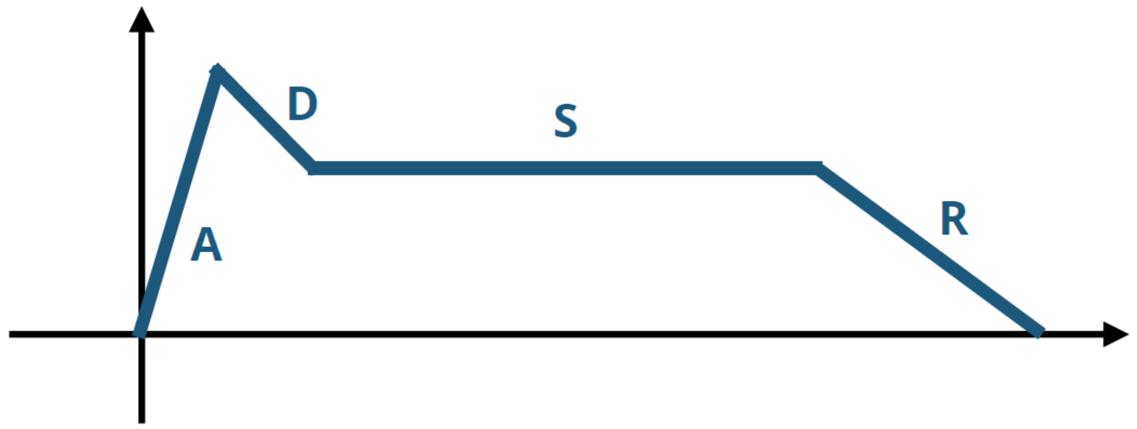
selector~

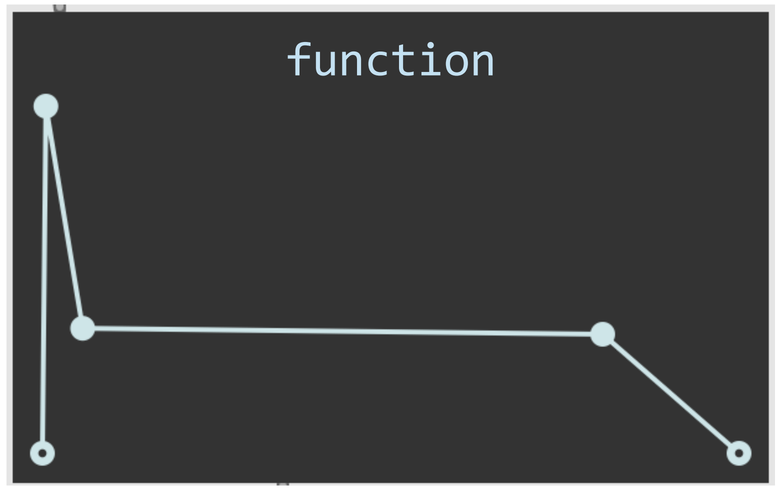


Single in, many out

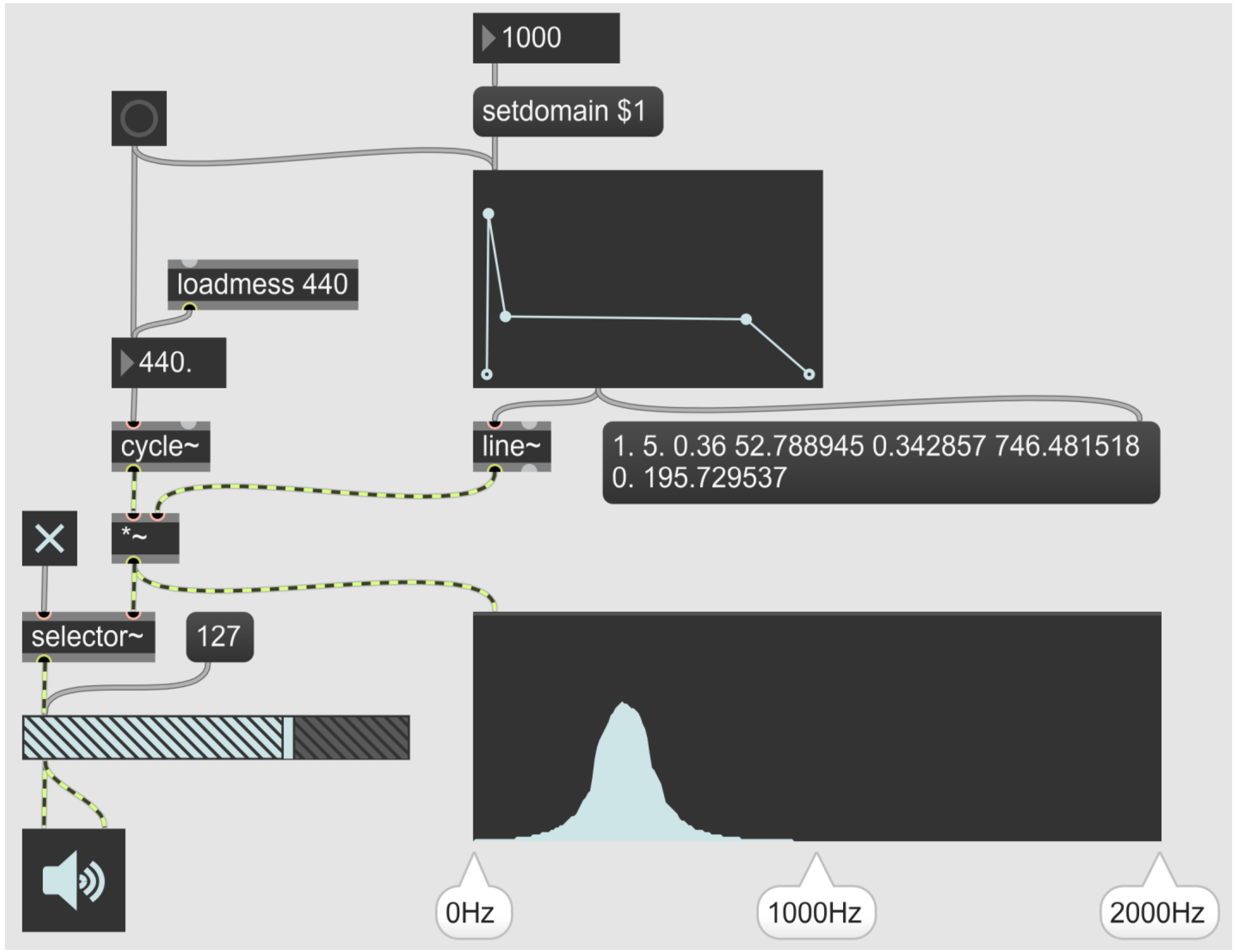
Review – ADSR

ADSR Envelope





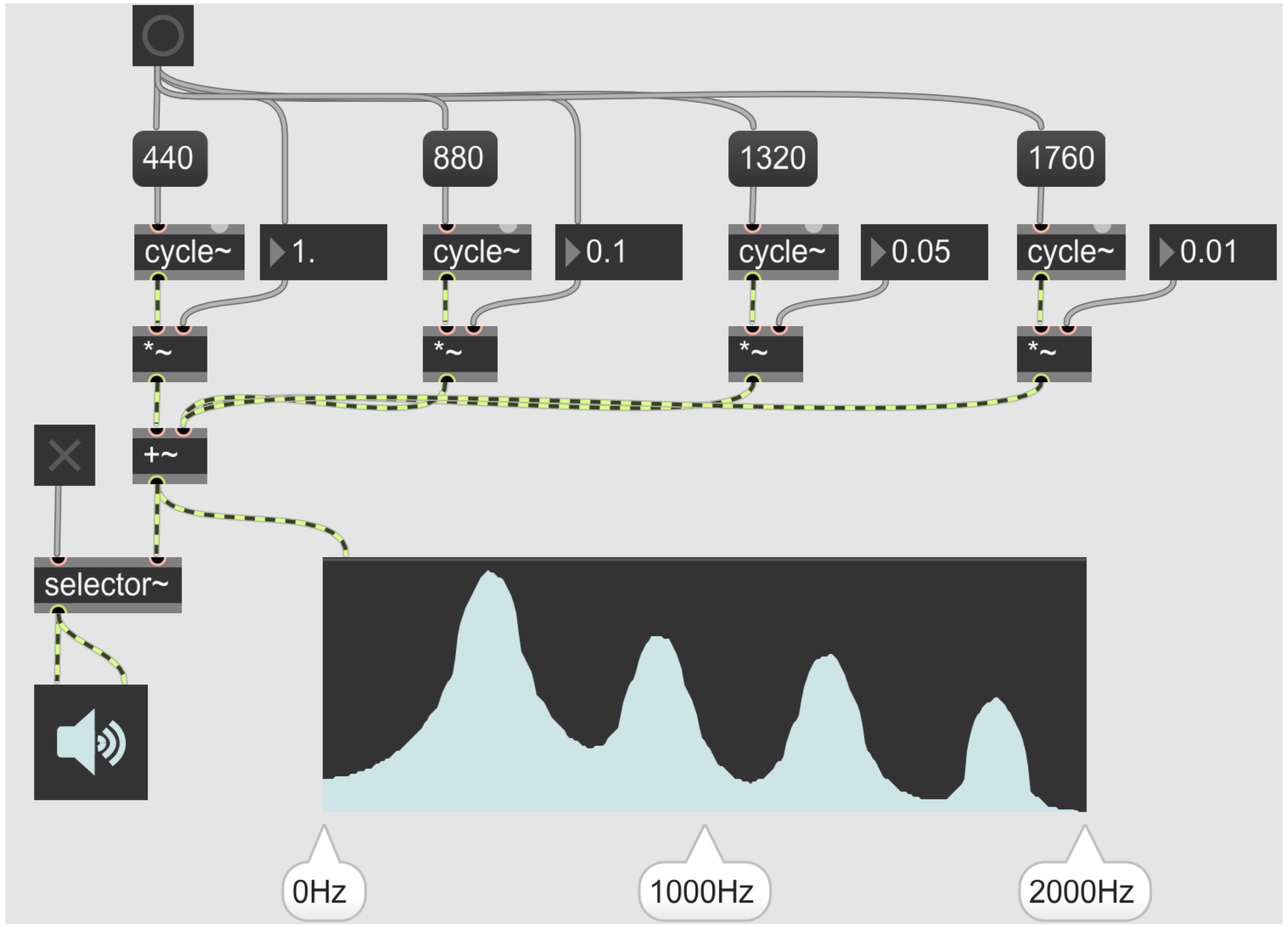
Amplitude Modulation



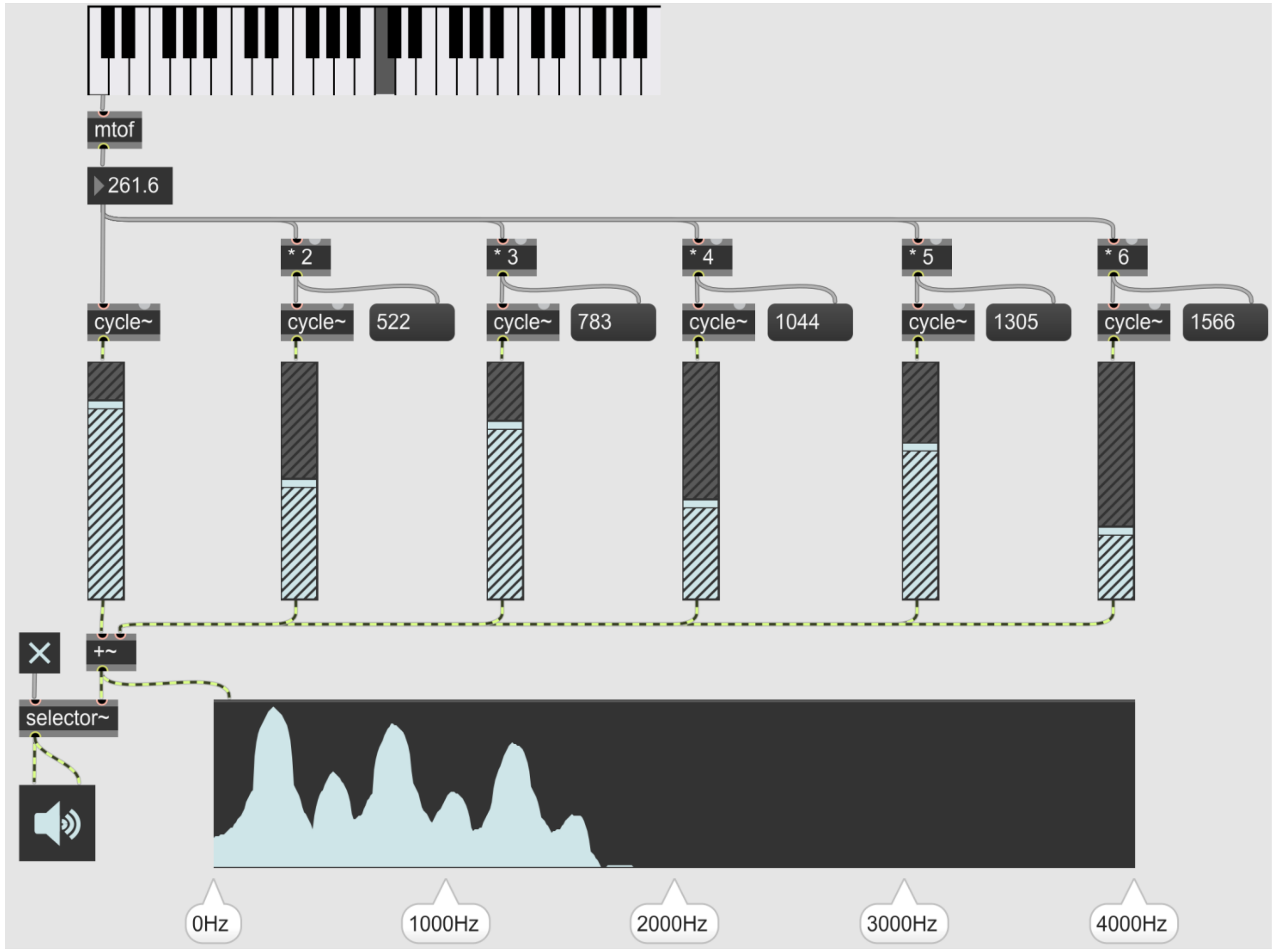
1. 5. 0.36 52.788945 0.342857 746.481518 0. 195.729537

Review – Additive Synth

Additive Synthesis

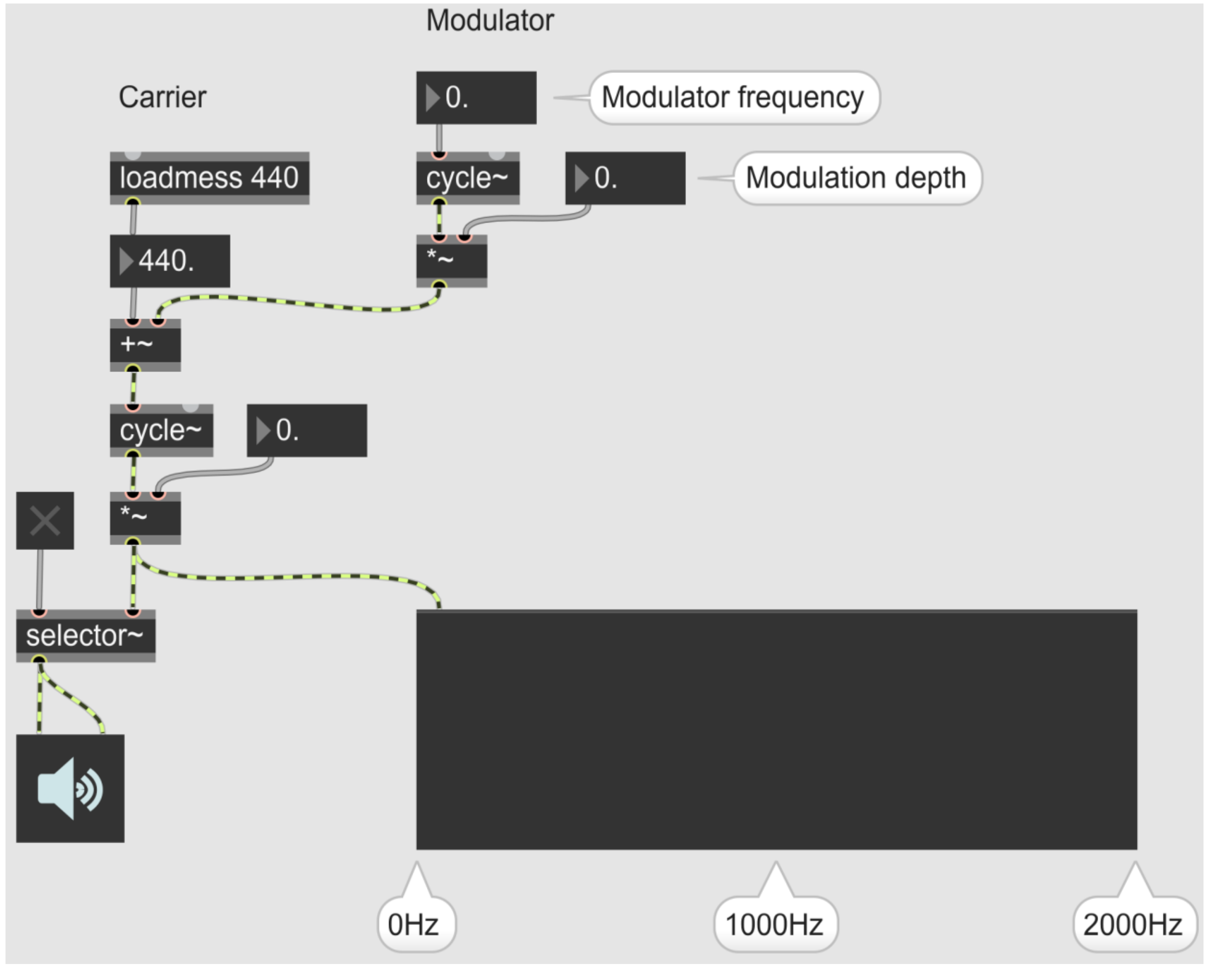


Additive Synthesis

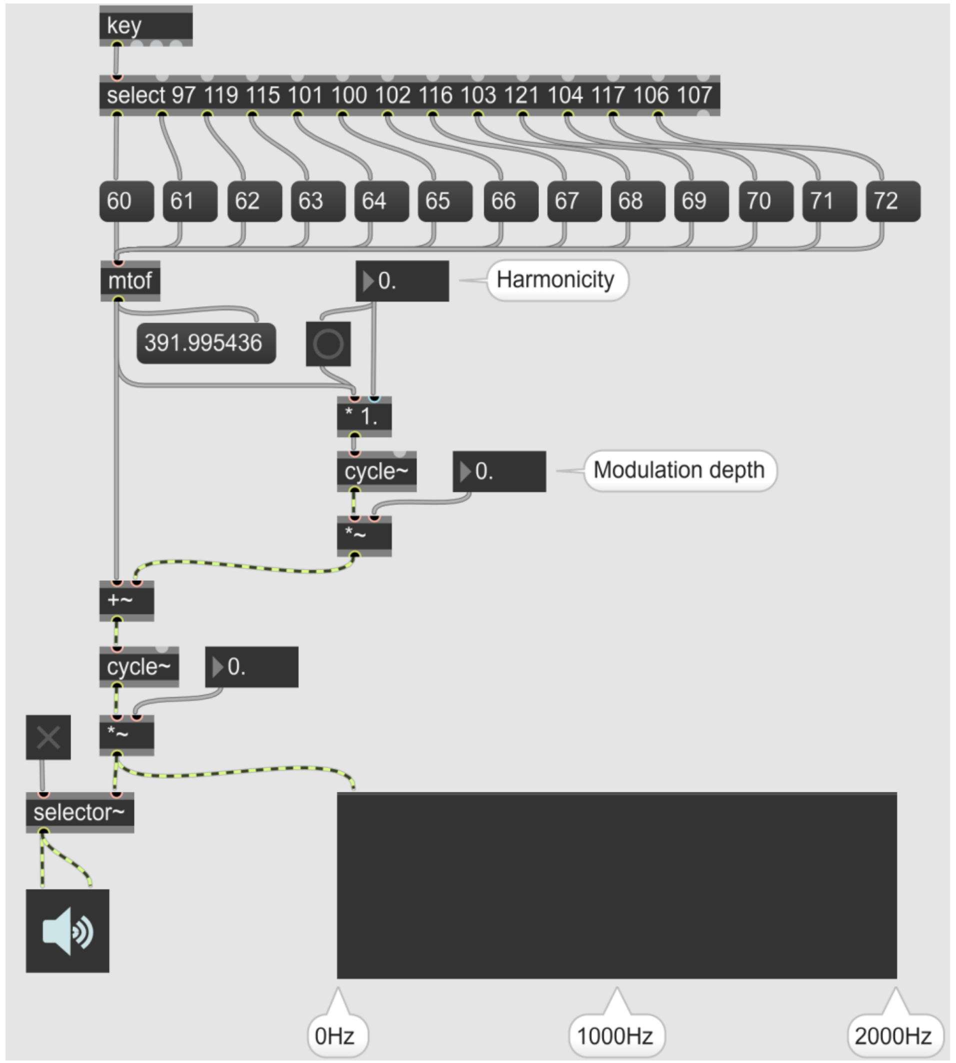


Review – FM Synth

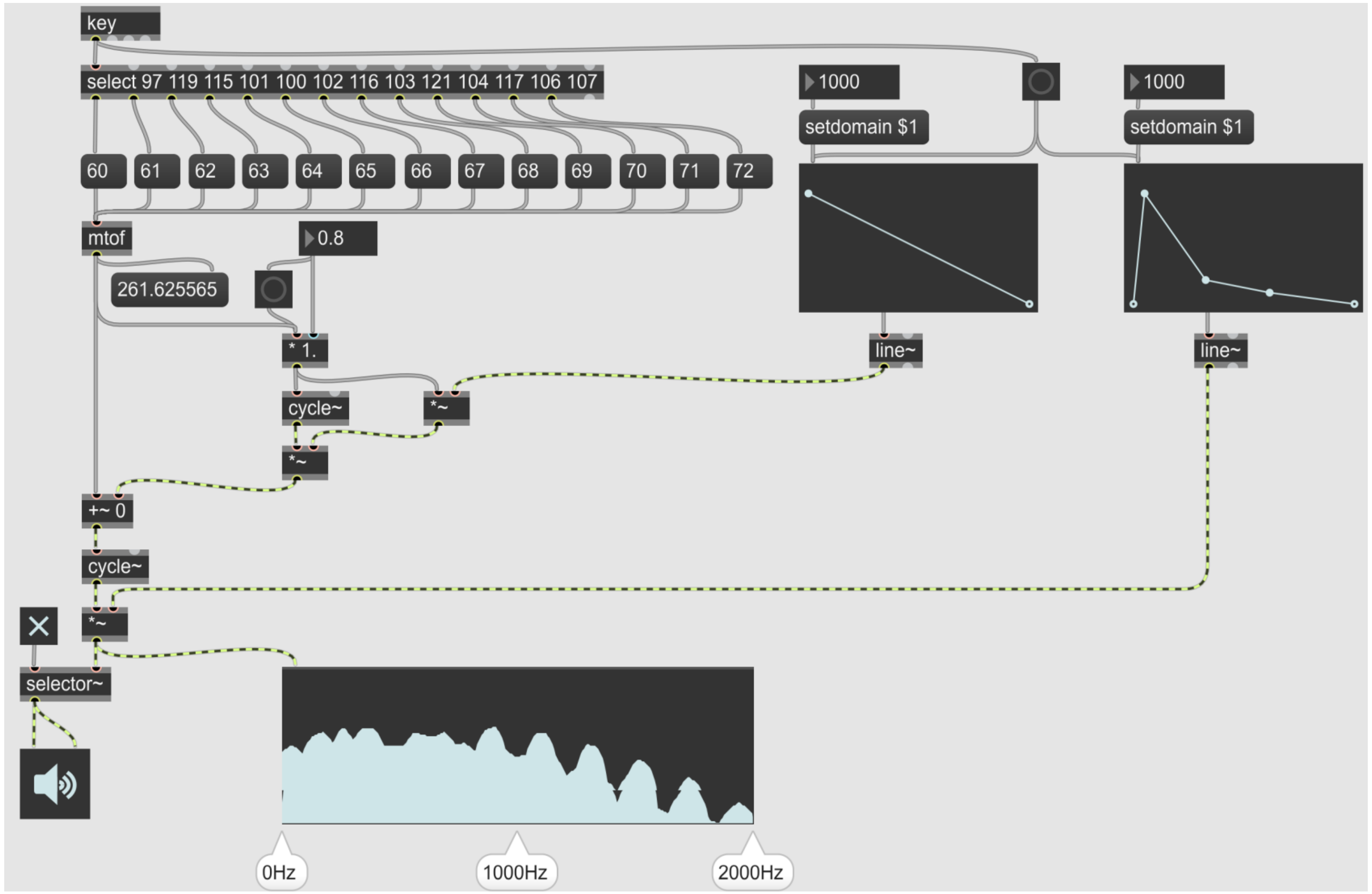
Frequency Modulation



FM Synth

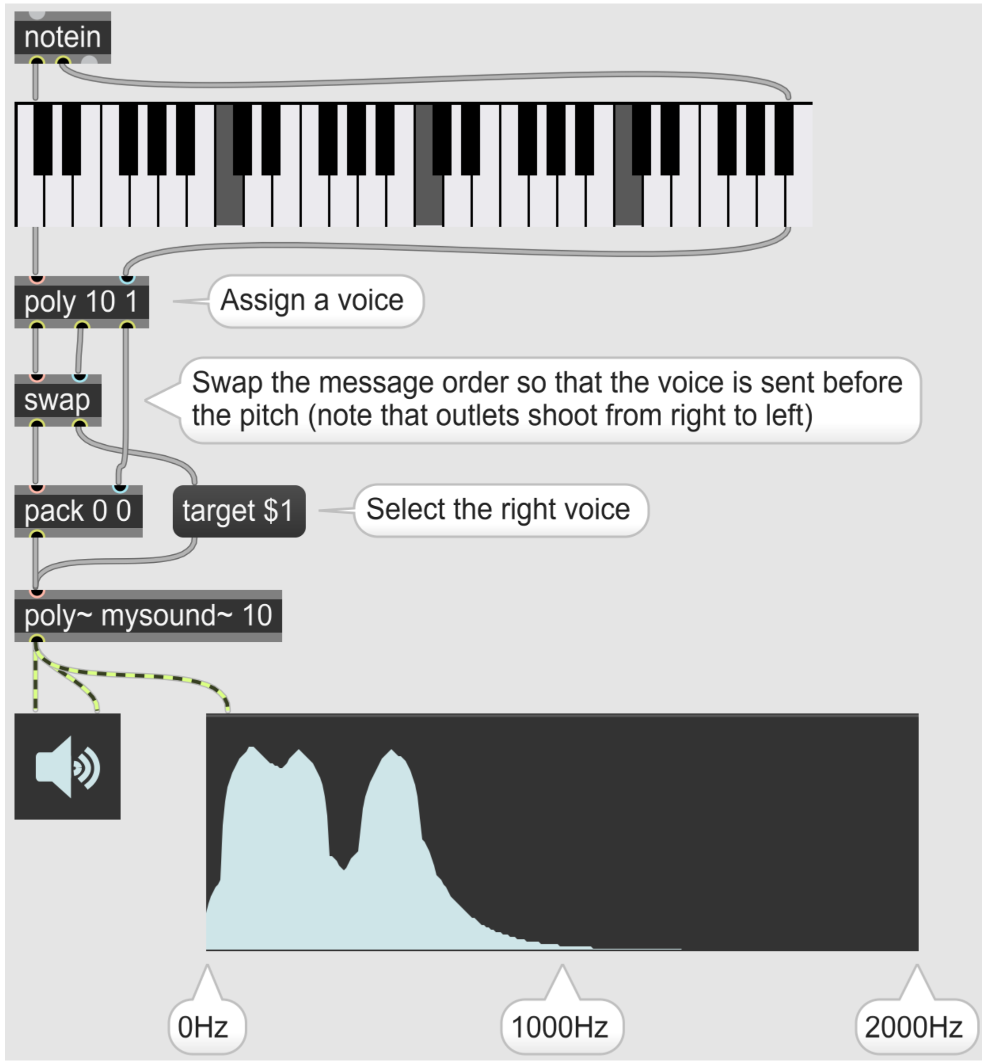

key
select 97 119 115 101 100 102 116 103 121 104 117 106 107
60
61
62
63
64
65
66
67
68
69
70
71
72
mtof
0.
Harmonicity
391.995436
* 1.
cycle~
0.
Modulation depth
*~
+~
cycle~
0.
*~
selector~
0Hz
1000Hz
2000Hz

FM Synth


key
select 97 119 115 101 100 102 116 103 121 104 117 106 107
60
61
62
63
64
65
66
67
68
69
70
71
72
mtof
261.625565
0.8
* 1.
cycle~
*~
*~
+~ 0
cycle~
*~
selector~
1000
setdomain $1
1000
setdomain $1
line~
line~
0Hz
1000Hz
2000Hz

Review – Polyphony

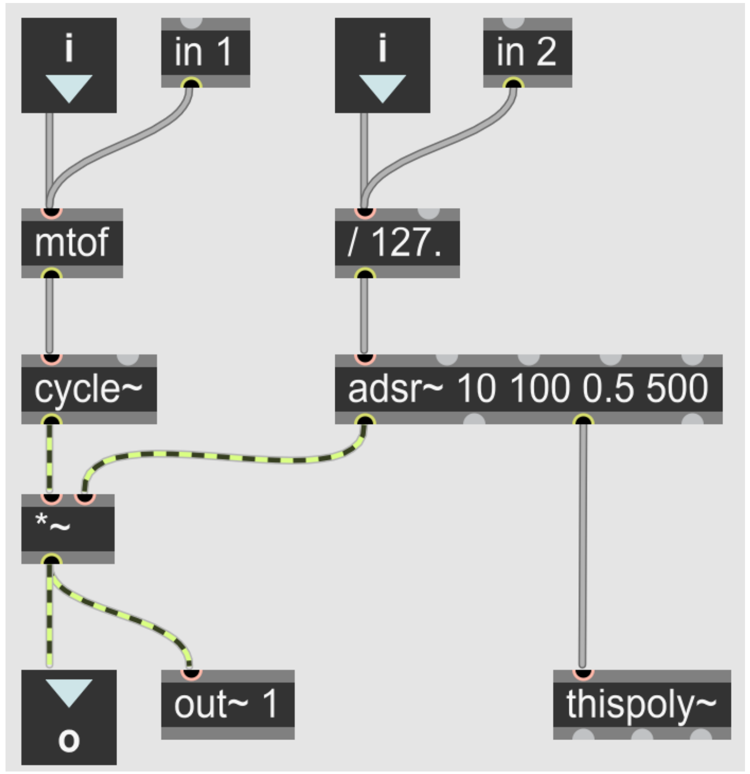
poly & poly~



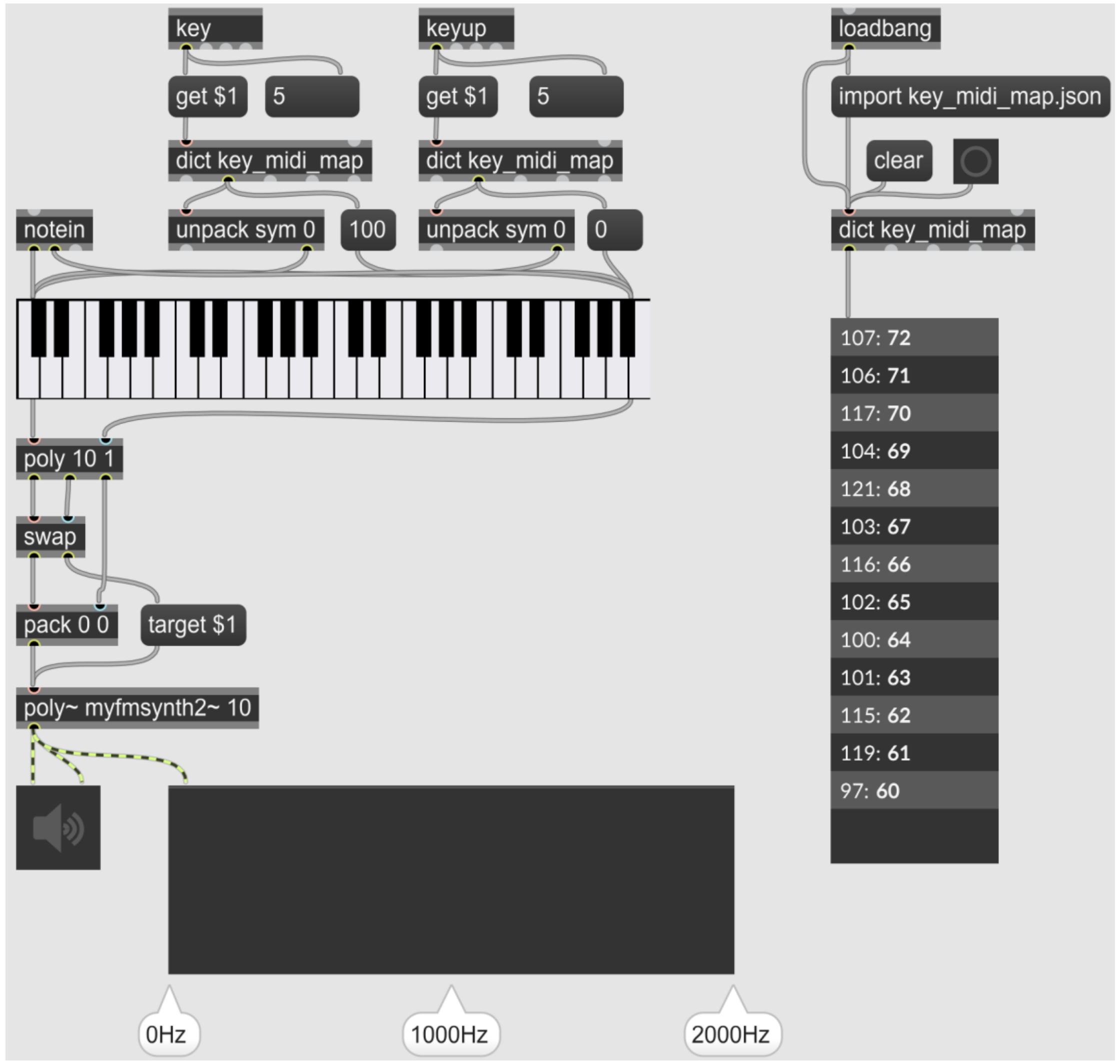
Assign a voice

Swap the message order so that the voice is sent before the pitch (note that outlets shoot from right to left)

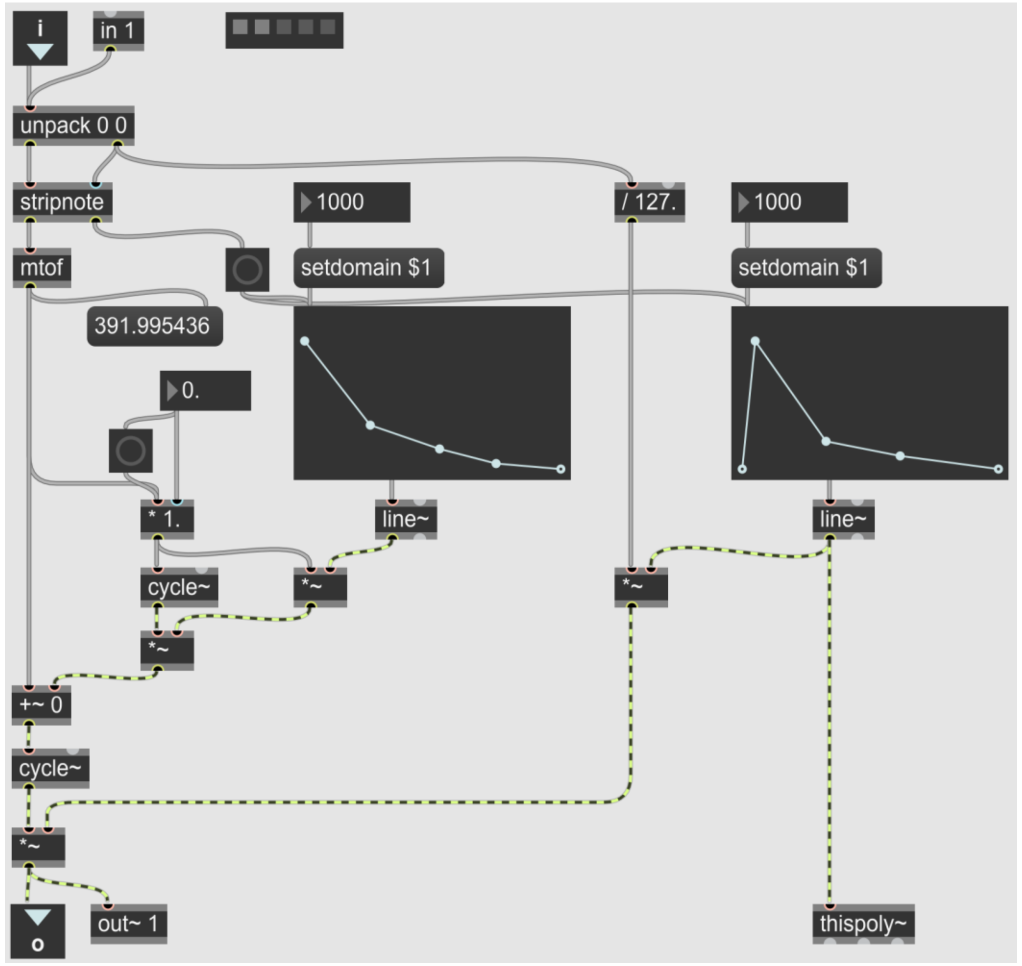
Select the right voice



Homework 5: Polyphony FM Synth

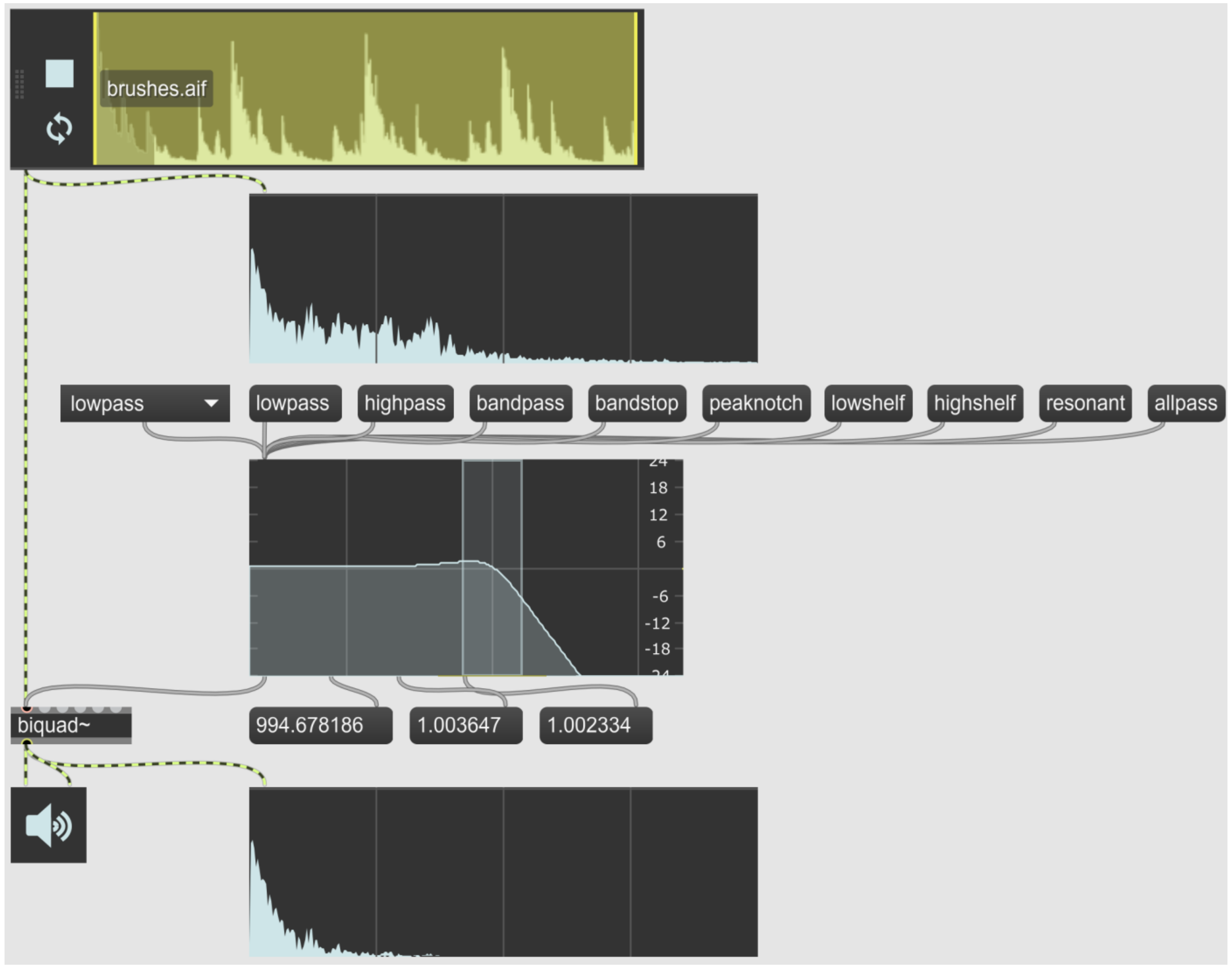


```
107: 72
106: 71
117: 70
104: 69
121: 68
103: 67
116: 66
102: 65
100: 64
101: 63
115: 62
119: 61
97: 60
```

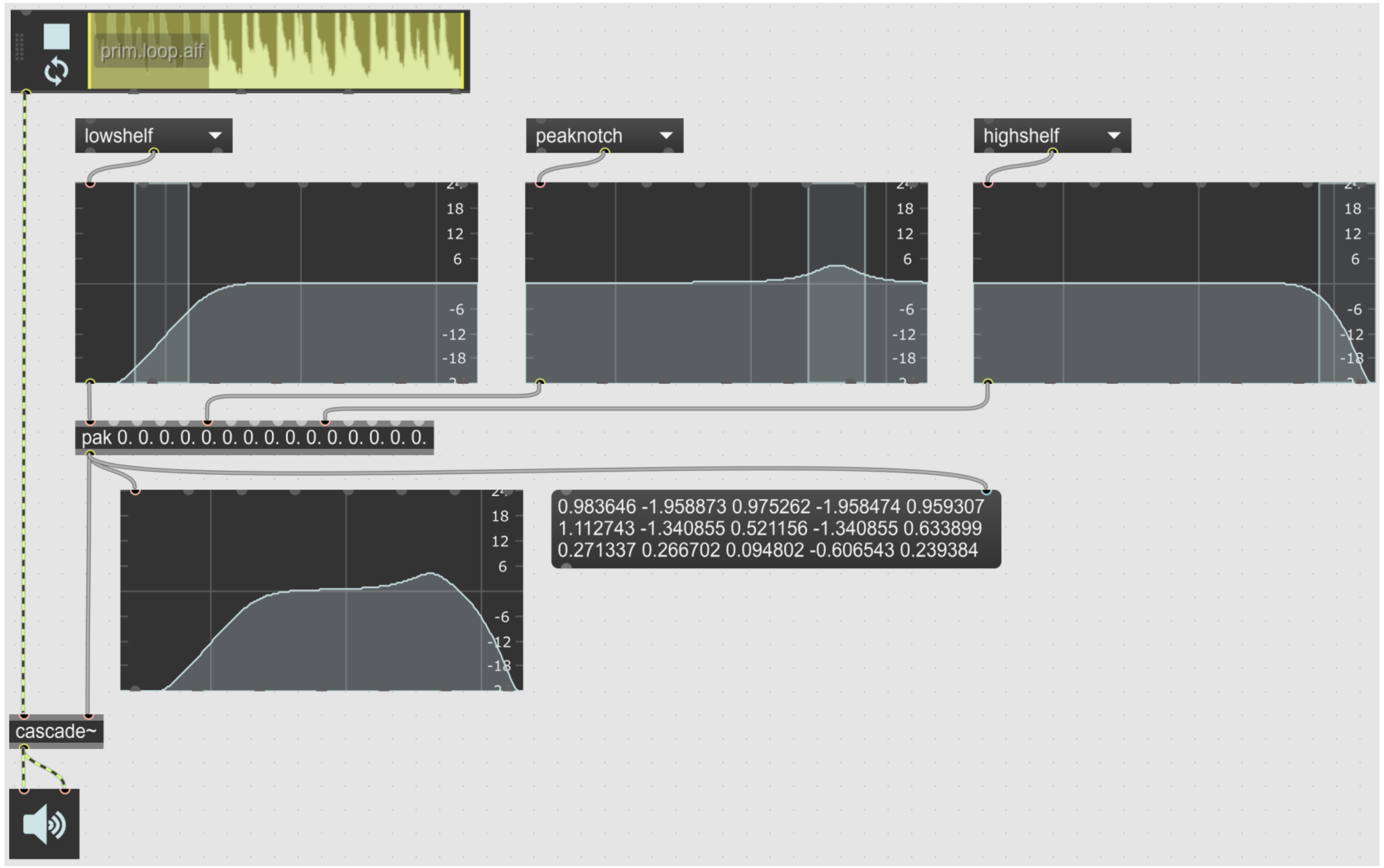


Review – Filters

biquad~


brushes.aif
lowpass
lowpass
highpass
bandpass
bandstop
peaknotch
lowshelf
highshelf
resonant
allpass
18
12
6
-6
-12
-18
biquad~
994.678186
1.003647
1.002334

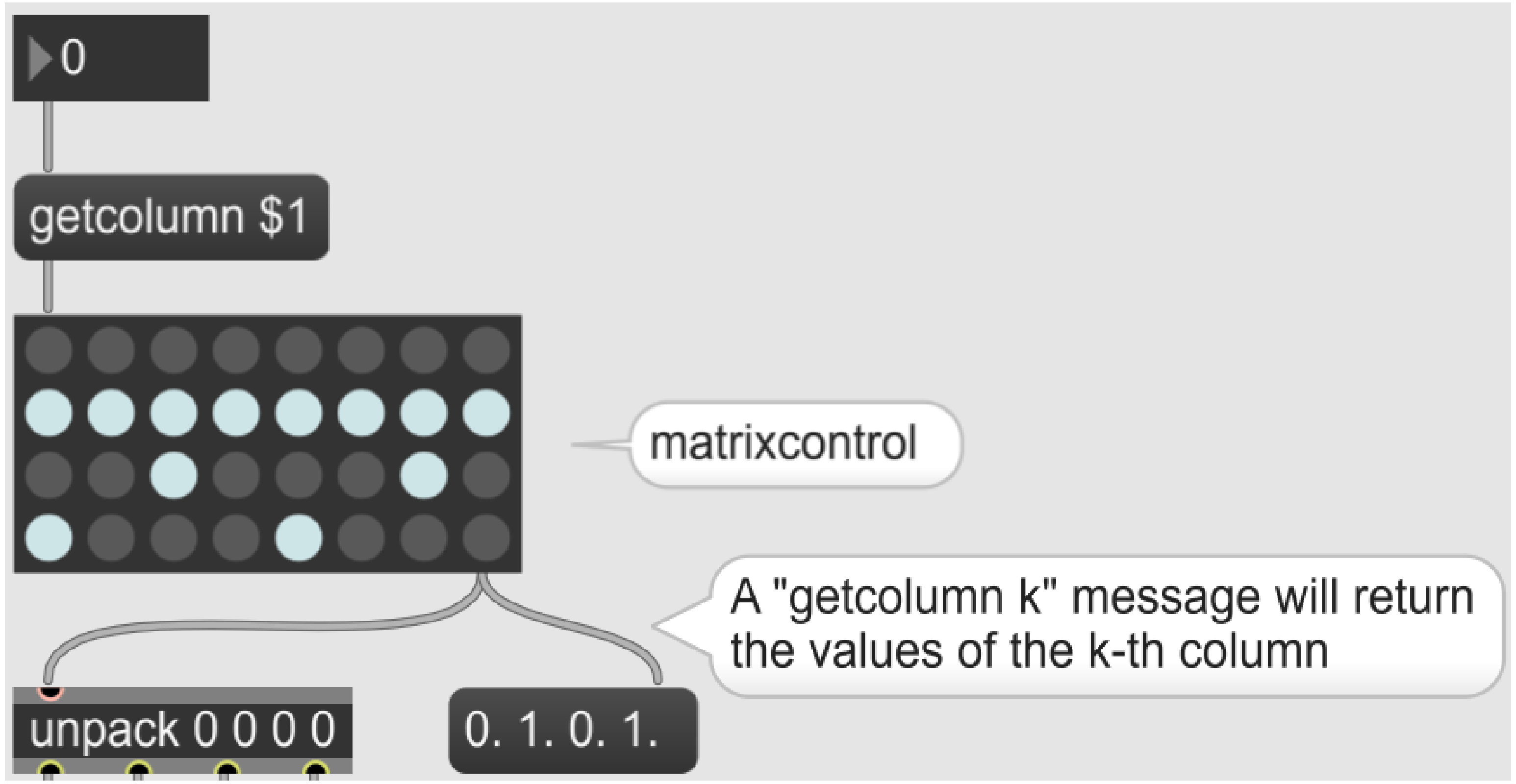
Multiband Equalizer



```
0.983646 -1.958873 0.975262 -1.958474 0.959307
1.112743 -1.340855 0.521156 -1.340855 0.633899
0.271337 0.266702 0.094802 -0.606543 0.239384
```

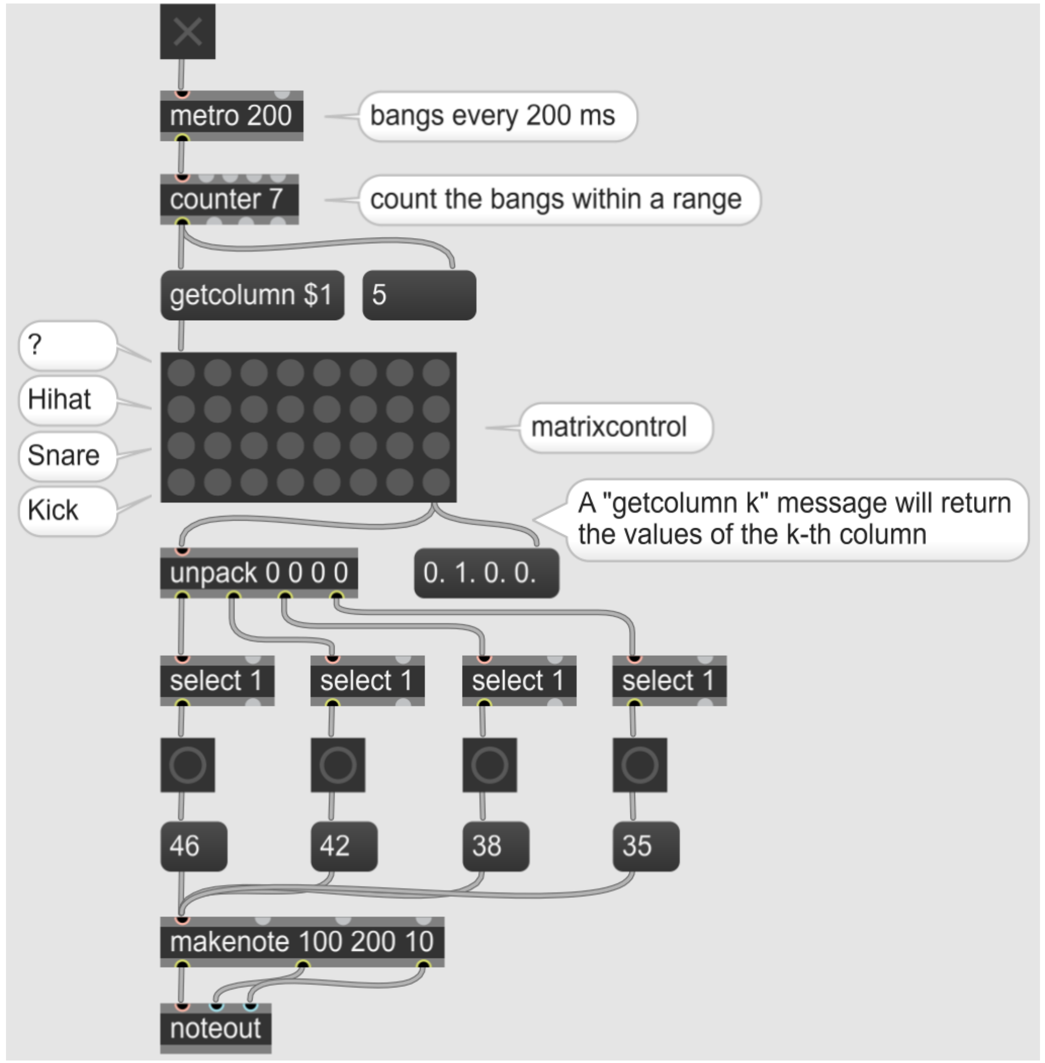
Review – Drum Machine

matrixcontrol

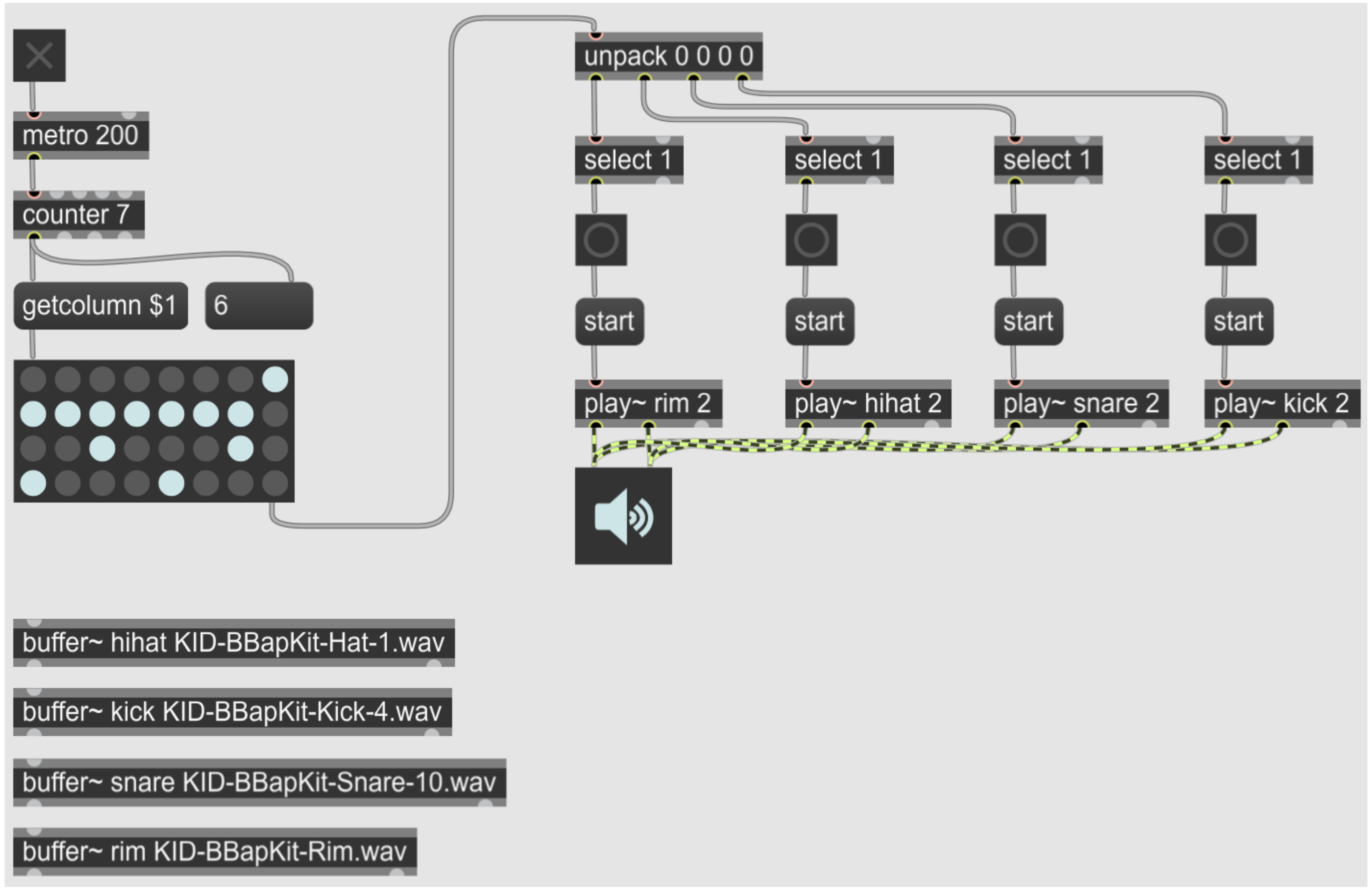


A "getcolumn k" message will return the values of the k-th column

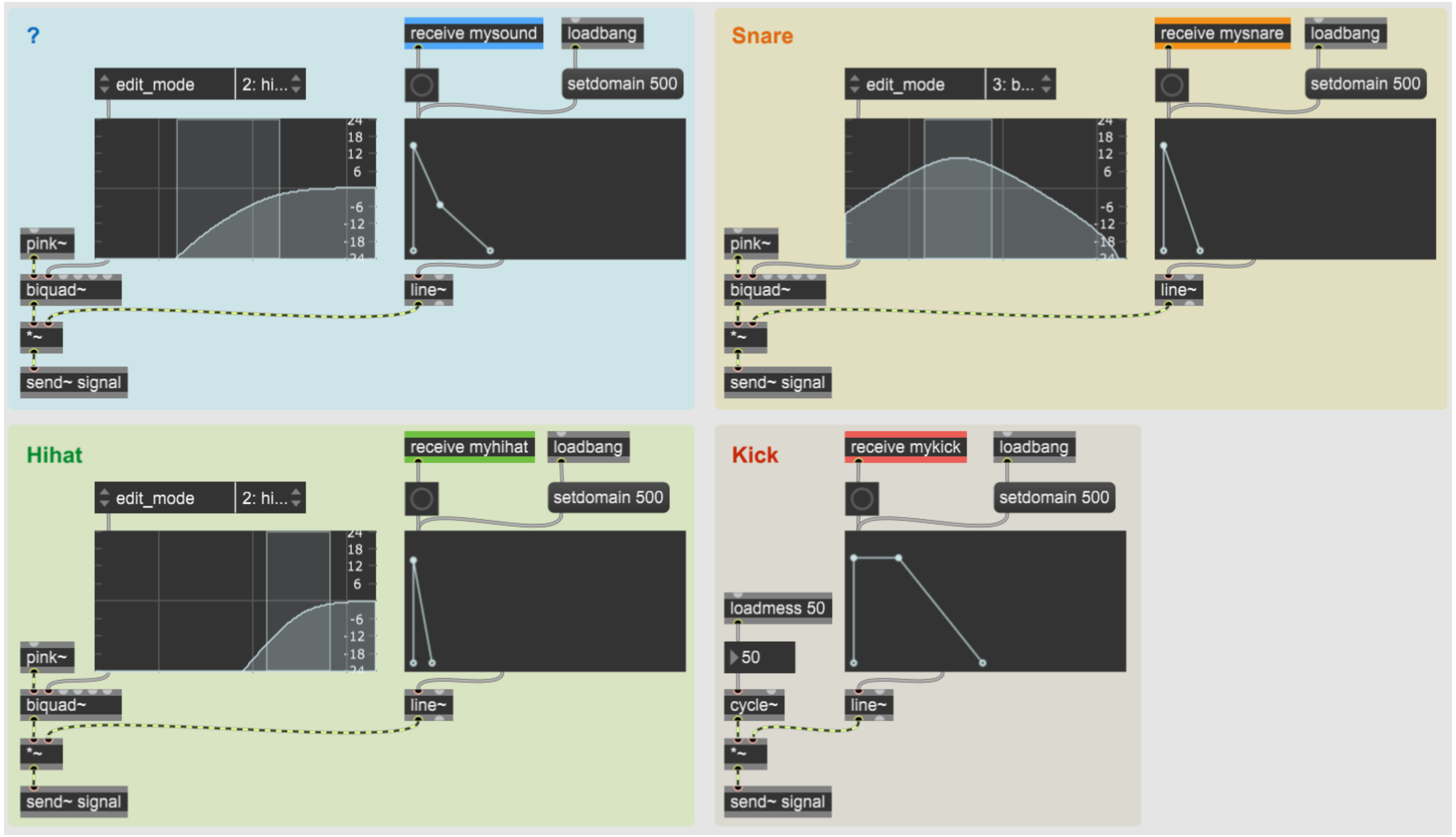
MIDI Drum Machine



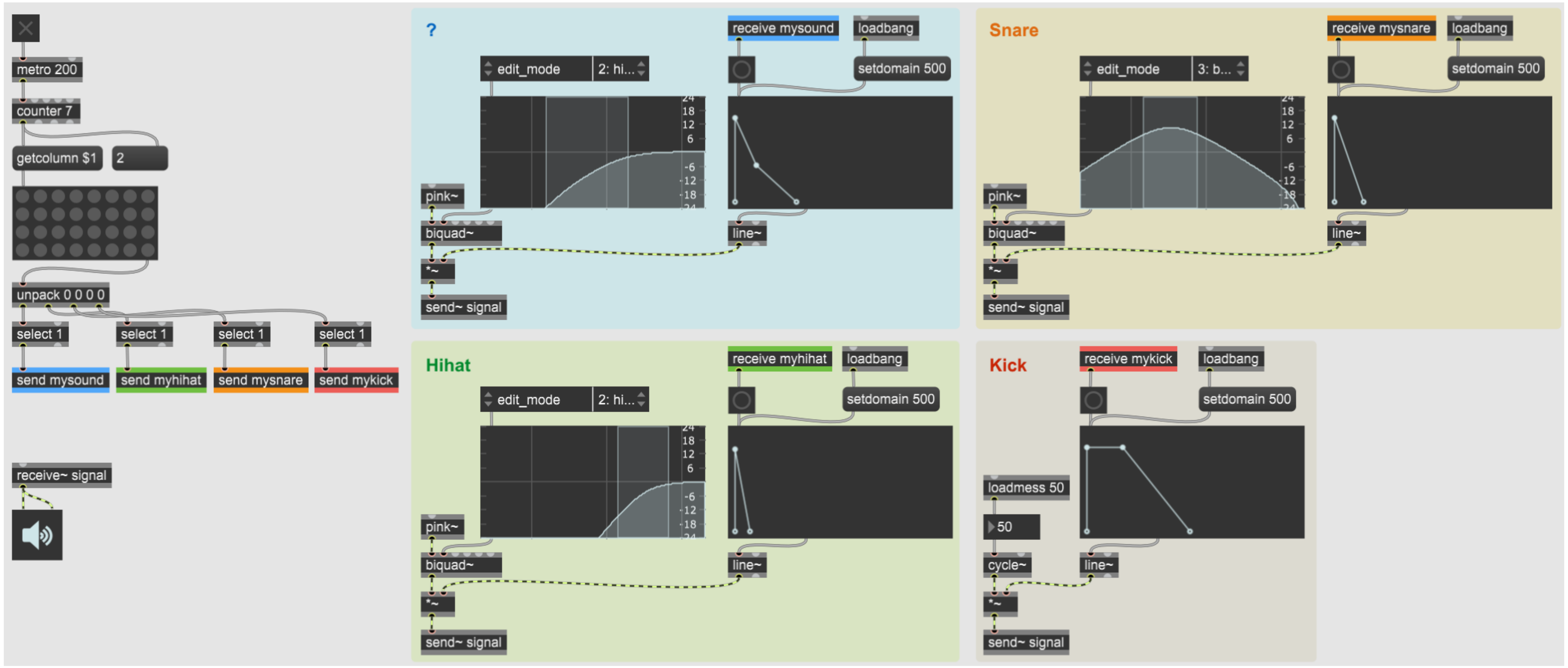
Sample-based Drum Machine



Synth Drum Sounds

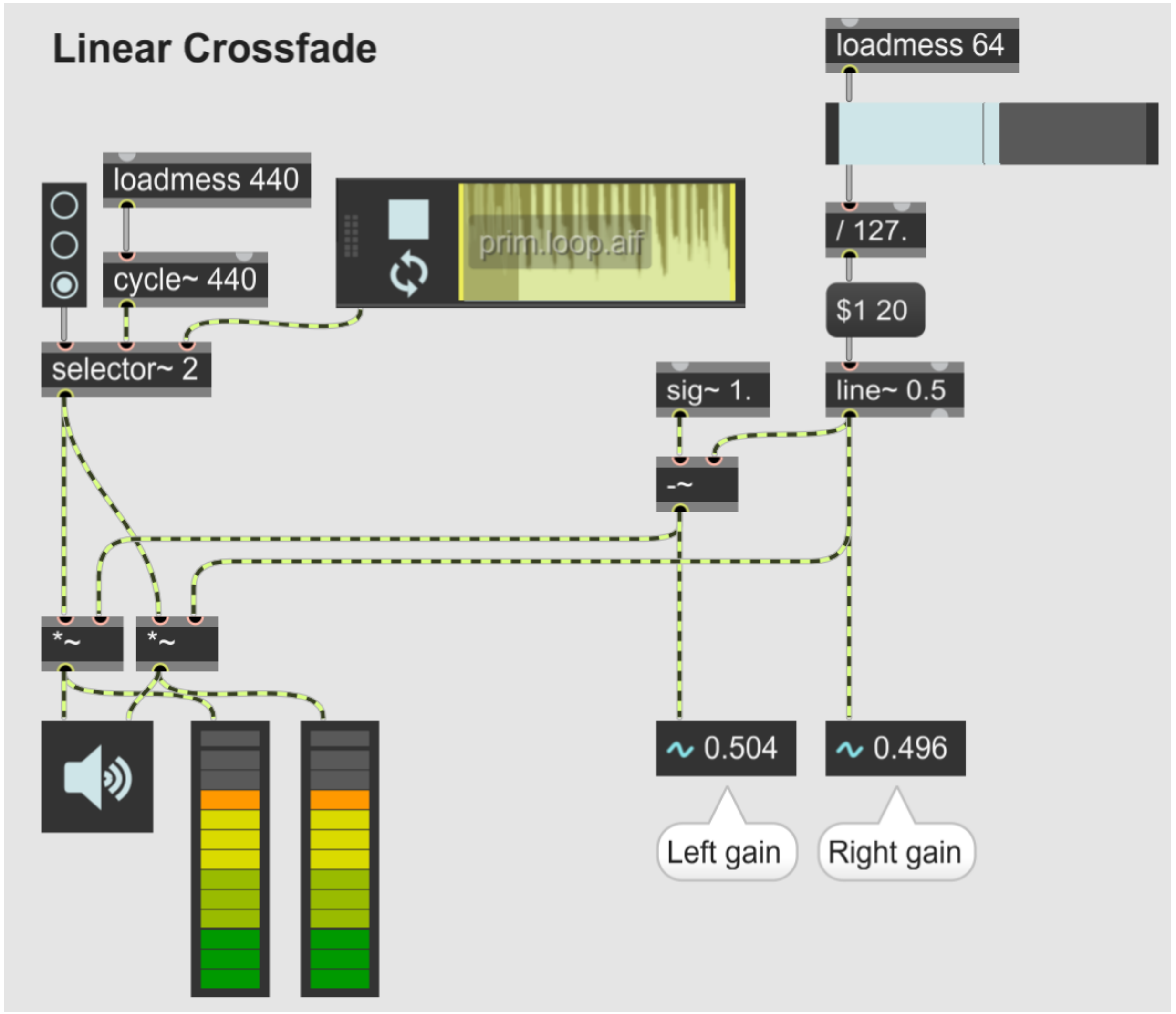


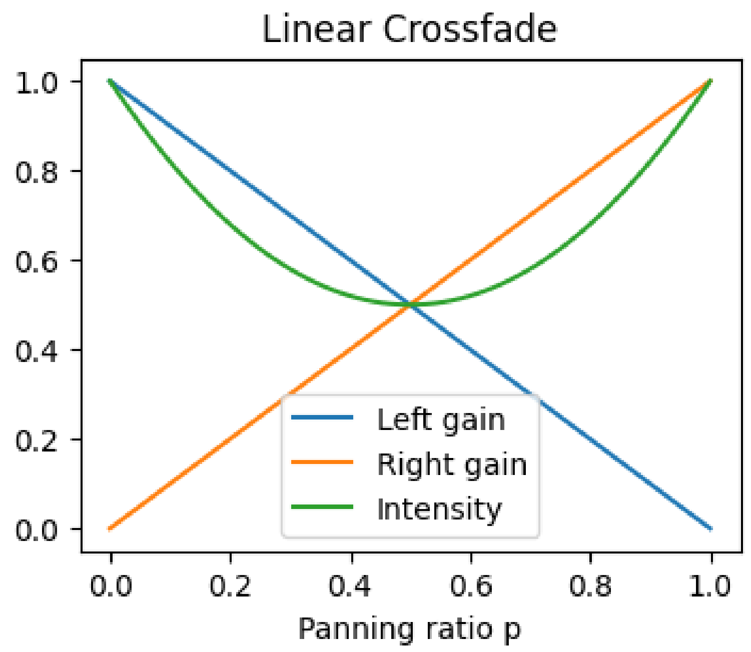
Synth Drum Machine



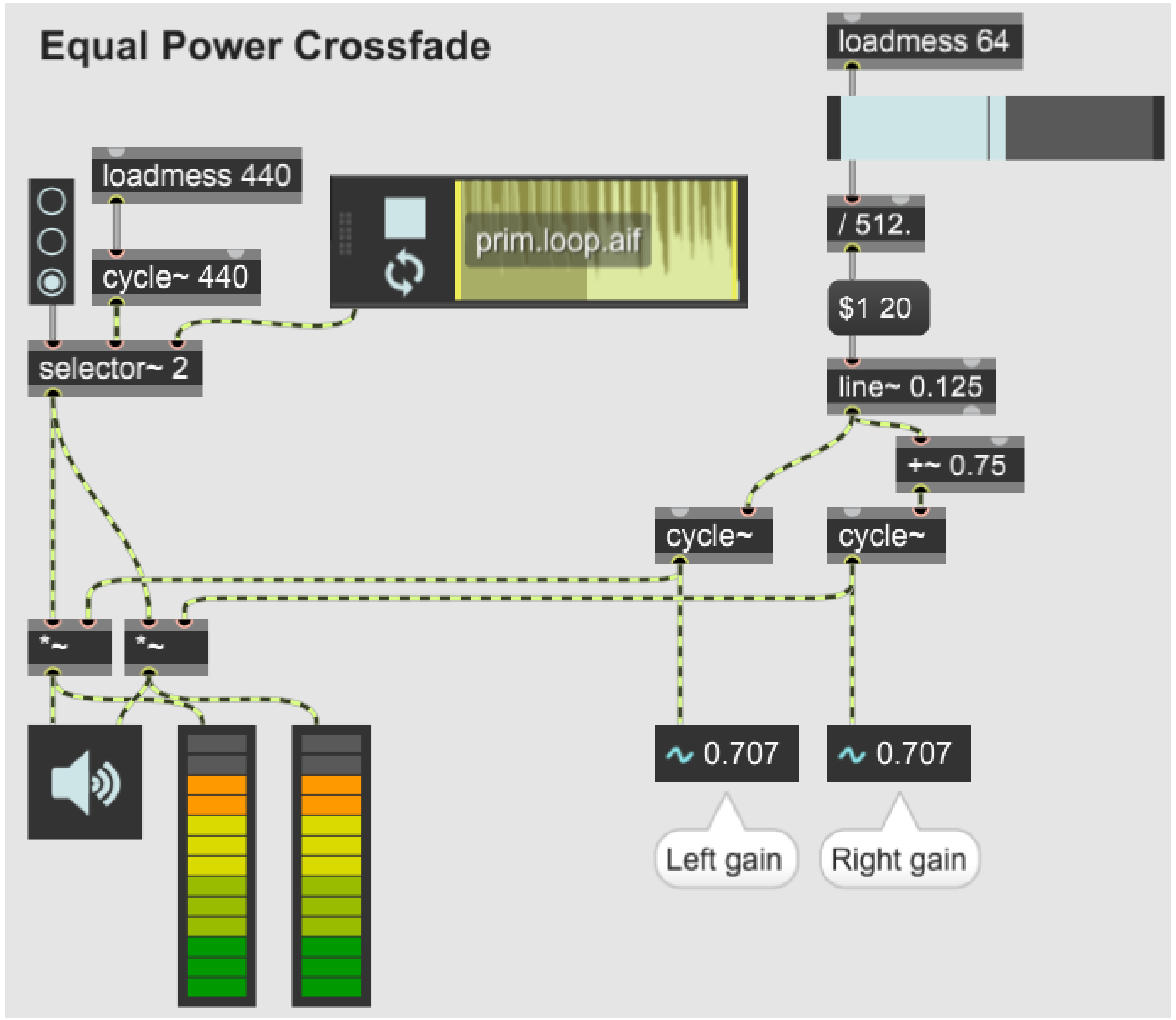
Review – Panning & Balancing

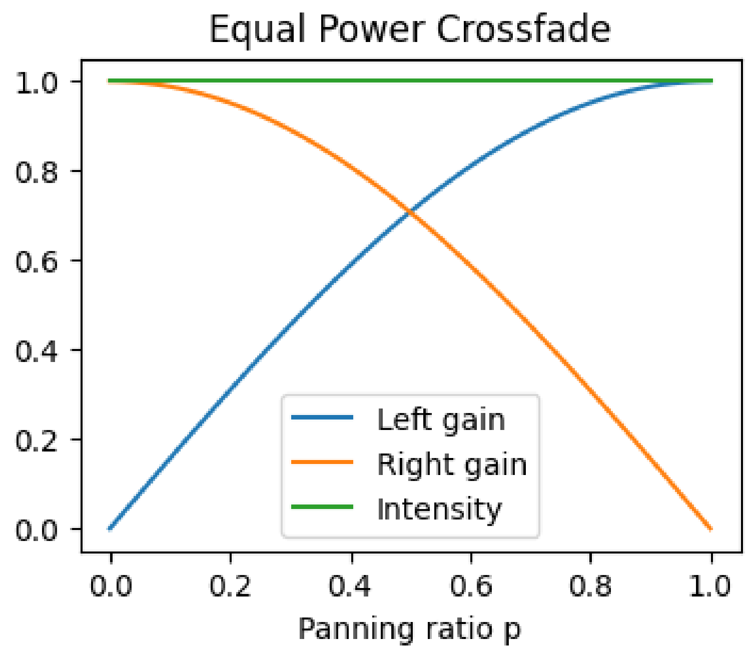
Linear Crossfade



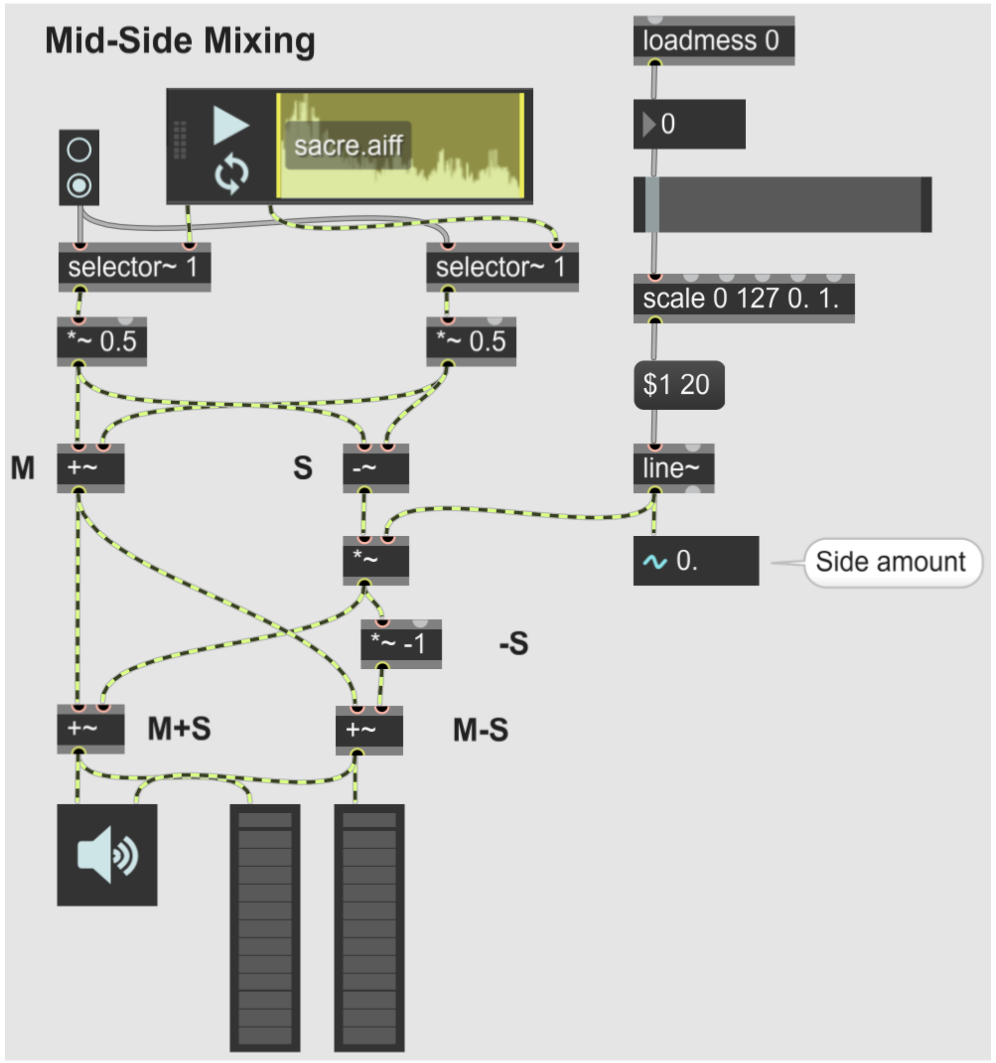


Equal Power Crossfade



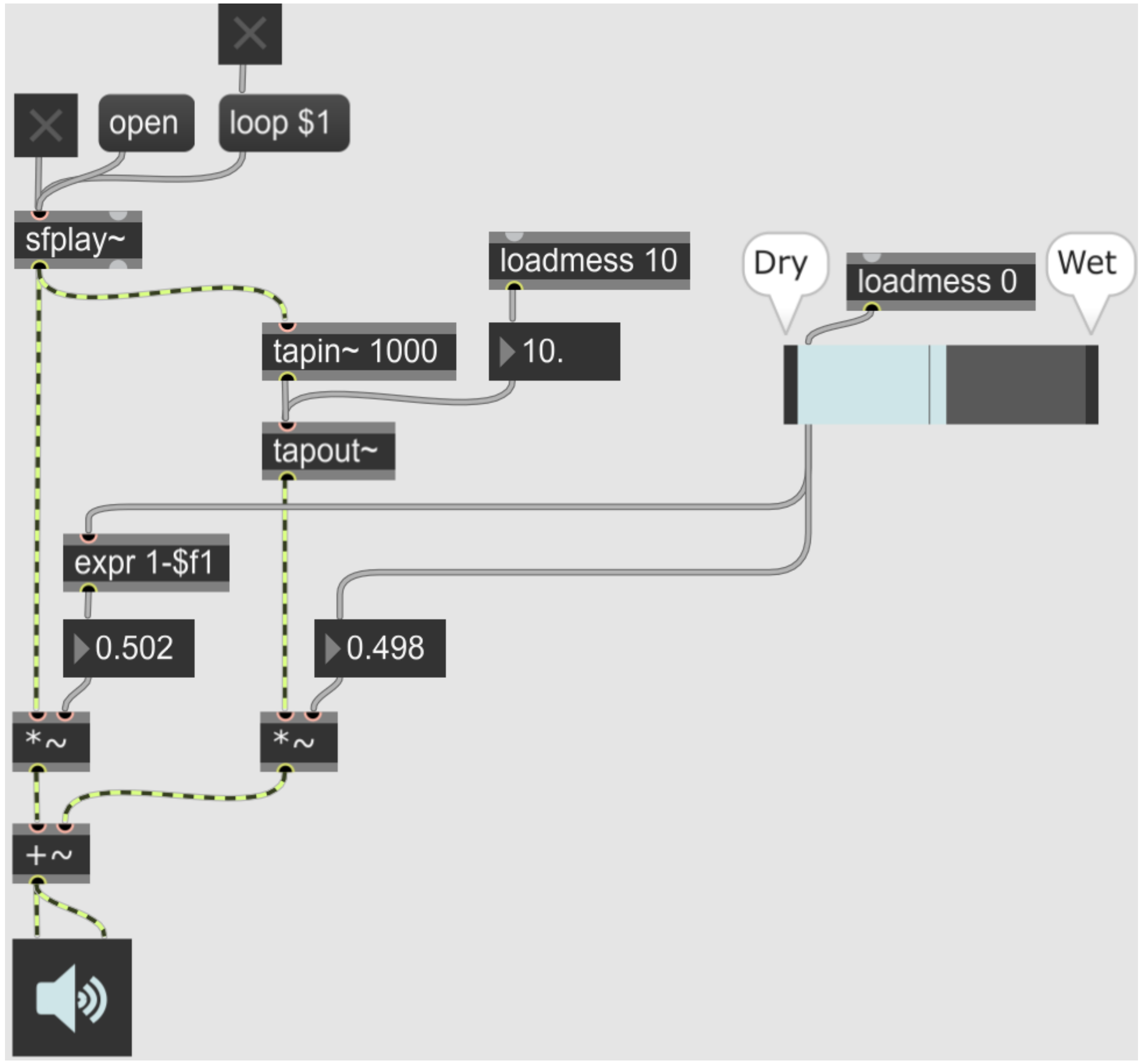


Mid-Side Mixing

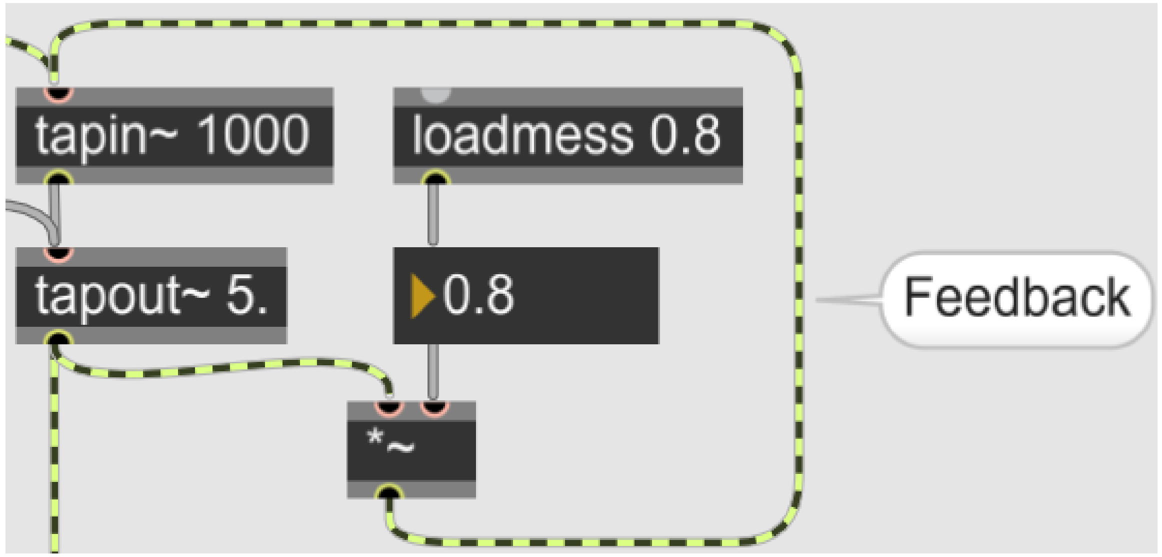


Review – Delay

Delay Effect

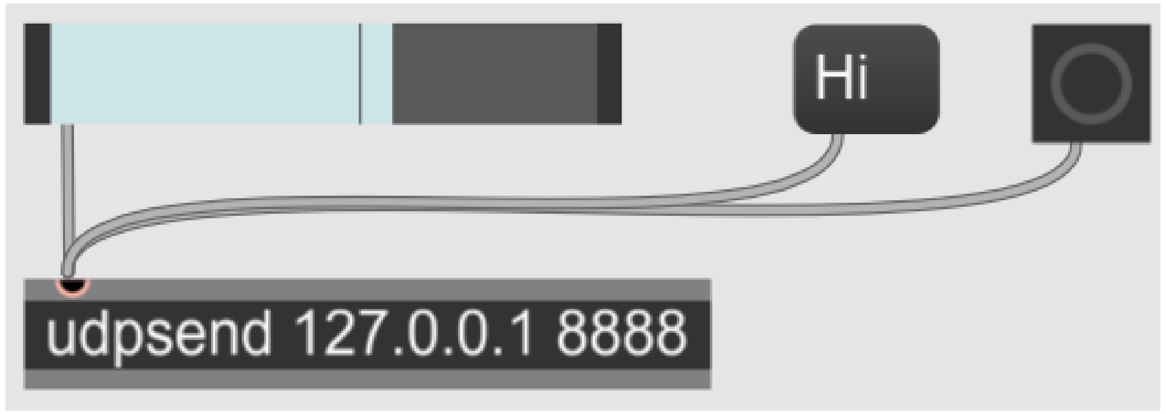


Echo Effect

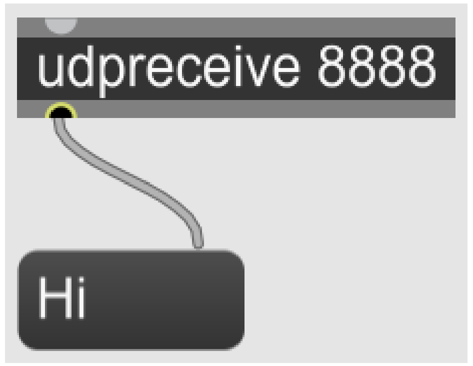


$$\begin{tikzcd}\text{Processing} \arrow[r, shift left] & \text{Max} \arrow[l, shift left]\end{tikzcd}$$

Simple UDP

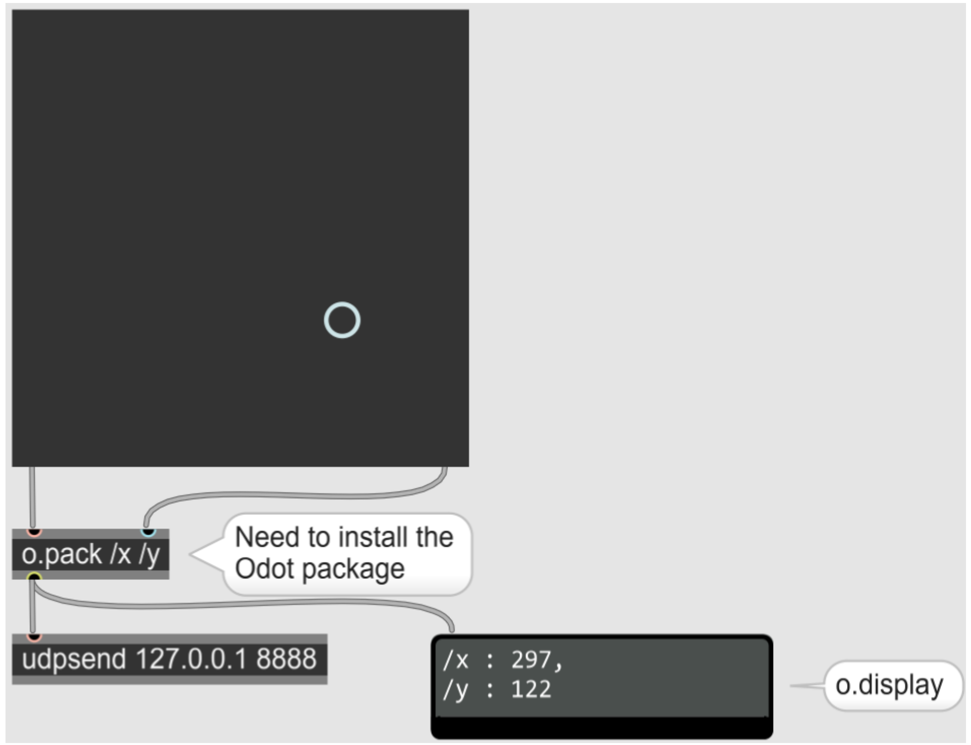


```
udpsend 127.0.0.1 8888
```



```
udpreceive 8888
```

Open Sound Control: Max → Processing

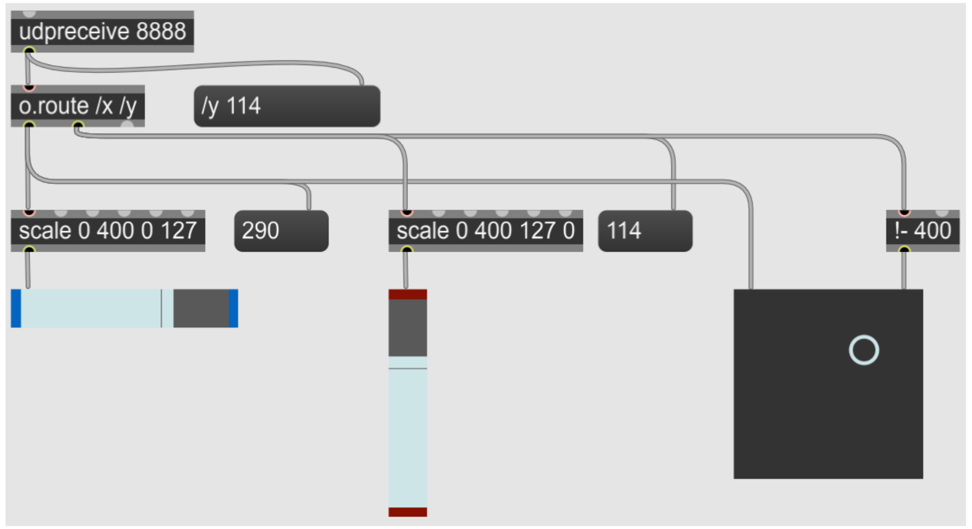


```
/x : 297,
/y : 122
```

```
import oscP5.*;
import netP5.*;

OscP5 osc;
NetAddress addr;
int x = 0, y = 0;

void setup() {
  size(400, 400);
  addr = new NetAddress("127.0.0.1", 8888);
  osc = new OscP5(this, 8888);
  osc.plug(this, "setX", "/x");
  osc.plug(this, "setY", "/y");
}

void setX(int data) {
  x = data;
}

void setY(int data) {
  y = height - data;
}

void draw() {
  background(0);
  circle(x, y, 50);
}
```

Open Sound Control: Processing → Max

```
import oscP5.*;
import netP5.*;

OscP5 osc;
NetAddress addr;
int x = 0, y = 0;

void setup() {
  size(400, 400);
  addr = new NetAddress("127.0.0.1", 8888);
  osc = new OscP5(this, 8888);
}

void mouseMoved() {
  OscMessage mesX = new OscMessage("/x");
  OscMessage mesY = new OscMessage("/y");

  mesX.add(mouseX);
  mesY.add(mouseY);

  osc.send(mesX, addr);
  osc.send(mesY, addr);
}

void draw() {
  background(0);
  circle(mouseX, mouseY, 50);
}
```


udpreceive 8888
o.route /x /y
/y 114
scale 0 400 0 127
290
scale 0 400 127 0
114
!- 400