Learning Processing

•••

Short Tutorial For Those Who Code

Processing

Processing is compiled into Java before it's run, but looks a lot like C++ for all those GENE 121 fans and has a very similar interface to Arduino..

Pros:

- Super awesome language for graphics and visualization
- Very easy to get a project up and running

Getting Started

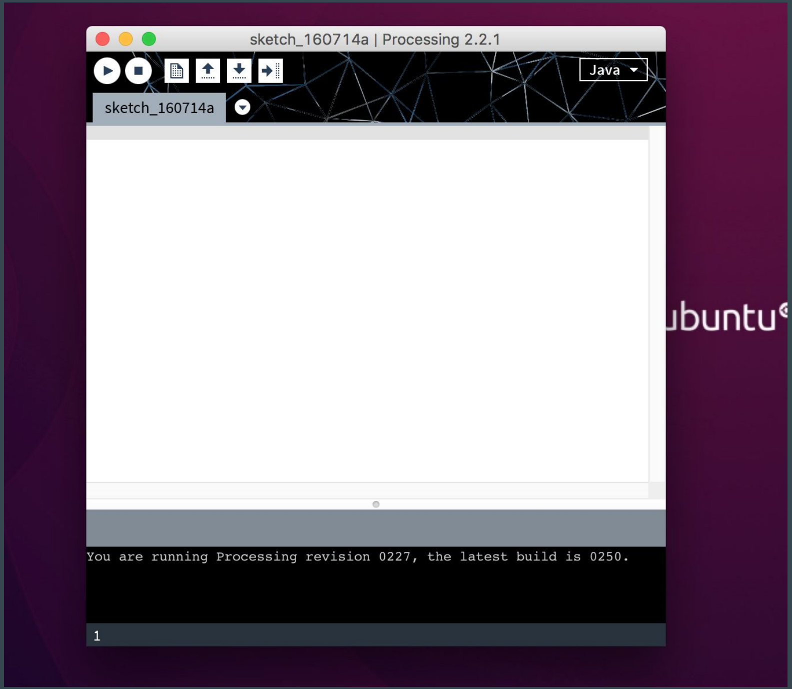
So as always, step one is installation:

Download the application: http://processing.org/download

Windows:	double click .zip to unpack, double click processing.exe
Mac:	open .zip, drag to Applications folder, run application
Linux:	download to ~, open terminal and type:

➔ `tar xvfz processing-<version number>.tgz`
➔ `cd processing-<version number>`
➔ `./processing`

Should look something like this:



```
You are running Processing revision 0227, the latest build is 0250.
```

Now we can start writing!

Bonus: what’s wrong with this picture?

Core Tenets: `Setup()` and `Draw()`

Equivalent to `main()`, these are the core functions.

Setup:

- Called at the beginning of the function
- Used to initialize data used in the program

Draw:

- Called before every frame is drawn (60Hz)
- Contains all of the graphics code of the program

Core Tenets: `Setup()` and `Draw()`

Variables often declared globally, above Setup.

Setup:

- Nothing new in terms of coding
- Just initialize data like you would

Draw:

- Define all graphical objects that would be drawn to the screen
- Consider things such as drawing order, speed/acceleration, relative positions

Hello, Processing!

So for our first program, we’re going to do something exceedingly simple (when done with processing): try to draw a square that follows the cursor on the screen.

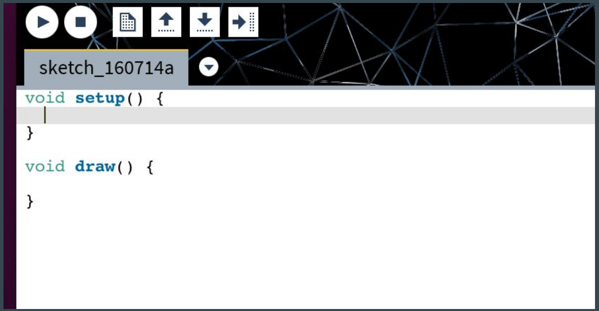
Setup:

- Initialize Screen
- Draw background (grey)

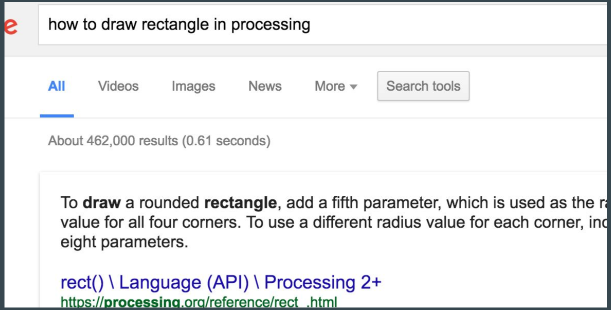
Draw:

- Draw a (red) rectangle centered around the cursor

Hello, Processing!


sketch_160714a

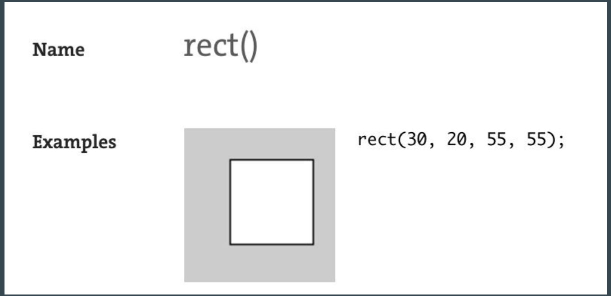
```
void setup() {
  
}

void draw() {

}
```


how to draw rectangle in processing

About 462,000 results (0.61 seconds)

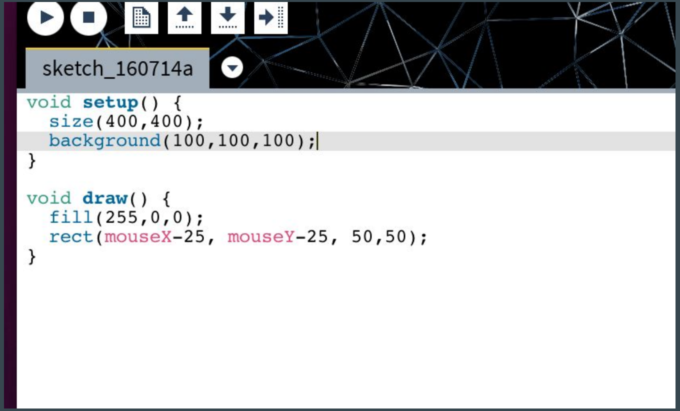
To **draw** a rounded **rectangle**, add a fifth parameter, which is used as the r
value for all four corners. To use a different radius value for each corner, inc
eight parameters.

rect() \ Language (API) \ Processing 2+
https://processing.org/reference/rect_.html


Name rect()

Examples

```
rect(30, 20, 55, 55);
```


sketch_160714a

```
void setup() {
  size(400,400);
  background(100,100,100);
}

void draw() {
  fill(255,0,0);
  rect(mouseX-25, mouseY-25, 50,50);
}
```

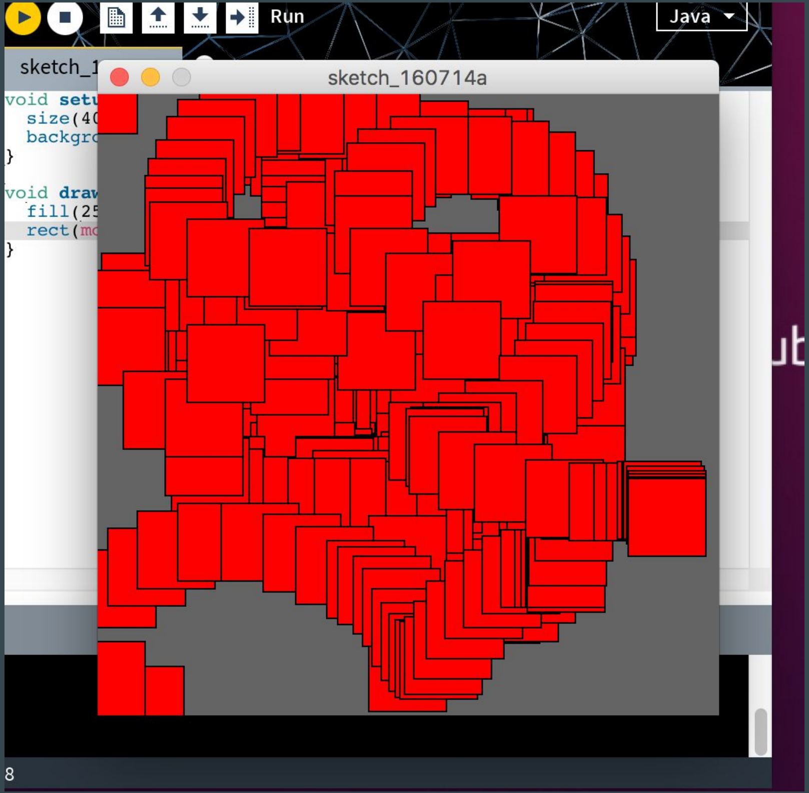
4 Lines Later...

We run it!

But it doesn’t work as we expected.

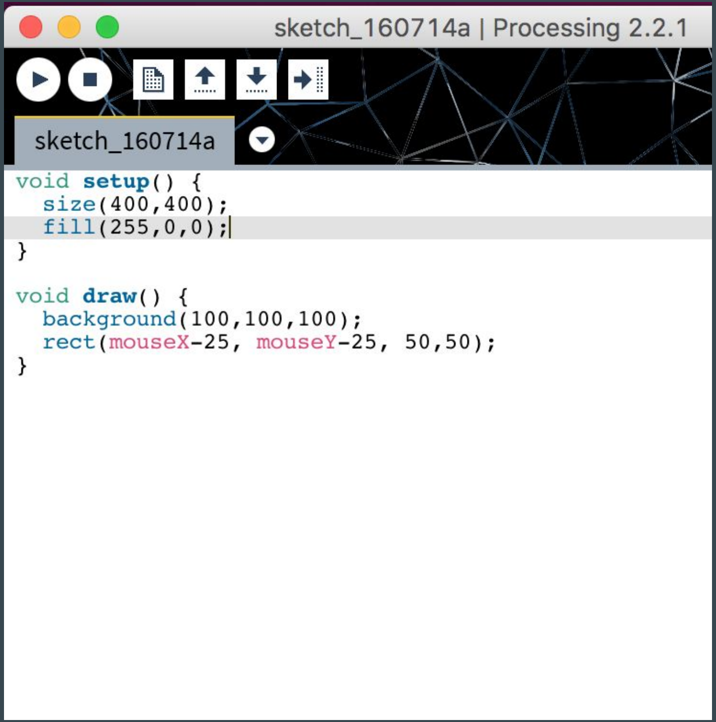
Why? Because processing draws each frame on top of the last, and all of our squares are being superimposed.

So we learn and we iterate.



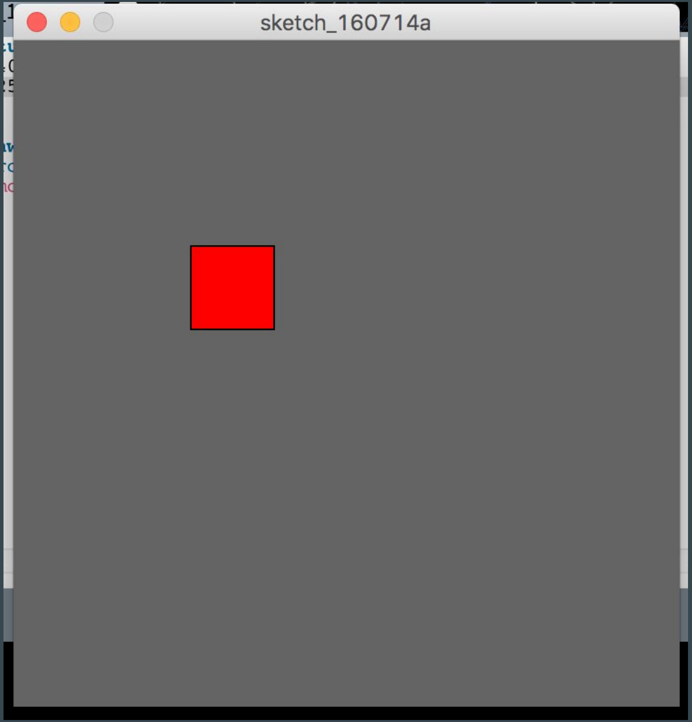
```
void setu
  size(40
  backgro
}

void draw
  fill(25
  rect(mo
}
```

Real Solution



sketch_160714a | Processing 2.2.1

```
void setup() {
  size(400,400);
  fill(255,0,0);
}

void draw() {
  background(100,100,100);
  rect(mouseX-25, mouseY-25, 50,50);
}
```



Main Takeaways:

Processing is extremely simple and iterable: it's <u>made</u> for playing around.

Processing also has an amazing reference.

So starting off with what we know about the framework, I'll give you guys a problem and you can just play around with that and read up on the documentation until you have it figured out.

My First Program

Draw “Hello World” in big letters on the screen, but have the colors change (however you want) every second.

Note: there are a bunch of ways to do this. Whatever works is good - the main point of the exercise is to familiarize with the language and docs.

Steps (or maybe hints)

Declare variables globally:

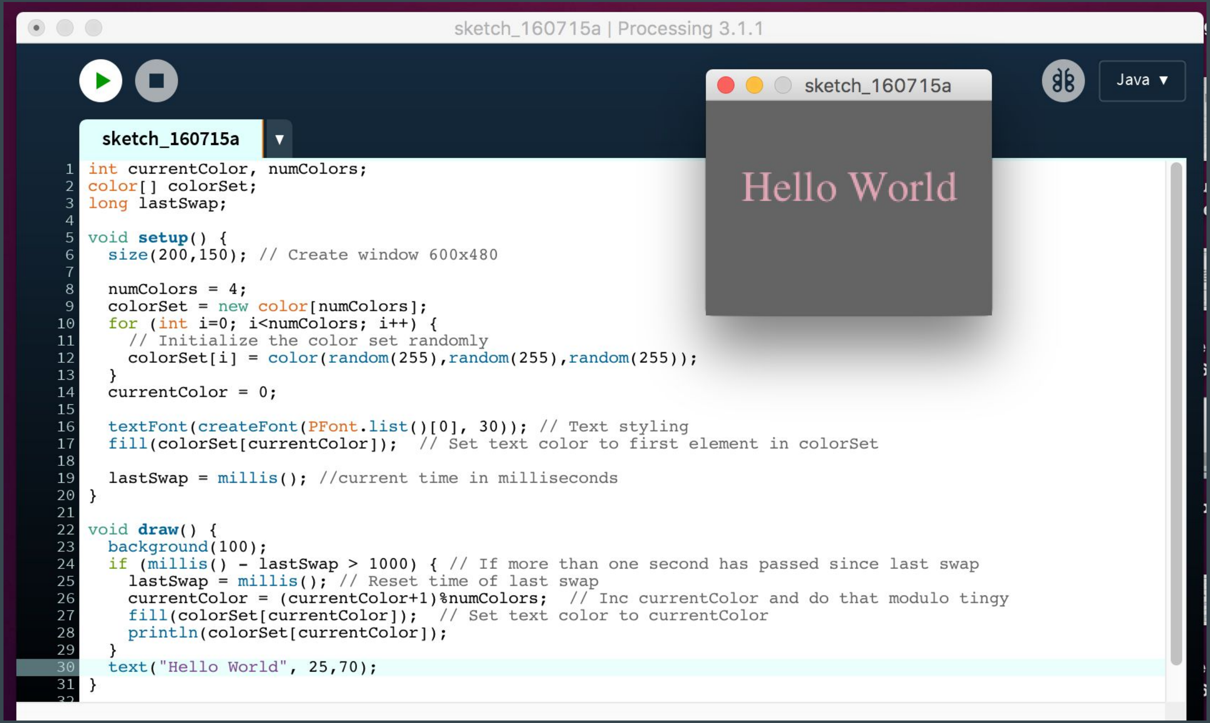
- Set of colors
- Current color
- Time of last swap (in milliseconds)

Setup: Set last swap to current time

Draw: If millis since last swap >1000, inc current color

Set text color to current color and draw text “Hello World”

Tadaaaa



```
int currentColor, numColors;
color[] colorSet;
long lastSwap;

void setup() {
  size(200,150); // Create window 600x480

  numColors = 4;
  colorSet = new color[numColors];
  for (int i=0; i<numColors; i++) {
    // Initialize the color set randomly
    colorSet[i] = color(random(255),random(255),random(255));
  }
  currentColor = 0;

  textFont(createFont(PFont.list()[0], 30)); // Text styling
  fill(colorSet[currentColor]);  // Set text color to first element in colorSet

  lastSwap = millis(); //current time in milliseconds
}

void draw() {
  background(100);
  if (millis() - lastSwap > 1000) { // If more than one second has passed since last swap
    lastSwap = millis(); // Reset time of last swap
    currentColor = (currentColor+1)%numColors;  // Inc currentColor and do that modulo tingy
    fill(colorSet[currentColor]);  // Set text color to currentColor
    println(colorSet[currentColor]);
  }
  text("Hello World", 25,70);
}
```

Now that we've mastered the basics:

Some things to play with:

- Graphing data (especially in real time)
- Interactivity (mousePressed( ), mouseX, mouseY…)
- Multi-screen applications (int currentScreen;)
- Interfacing with HW

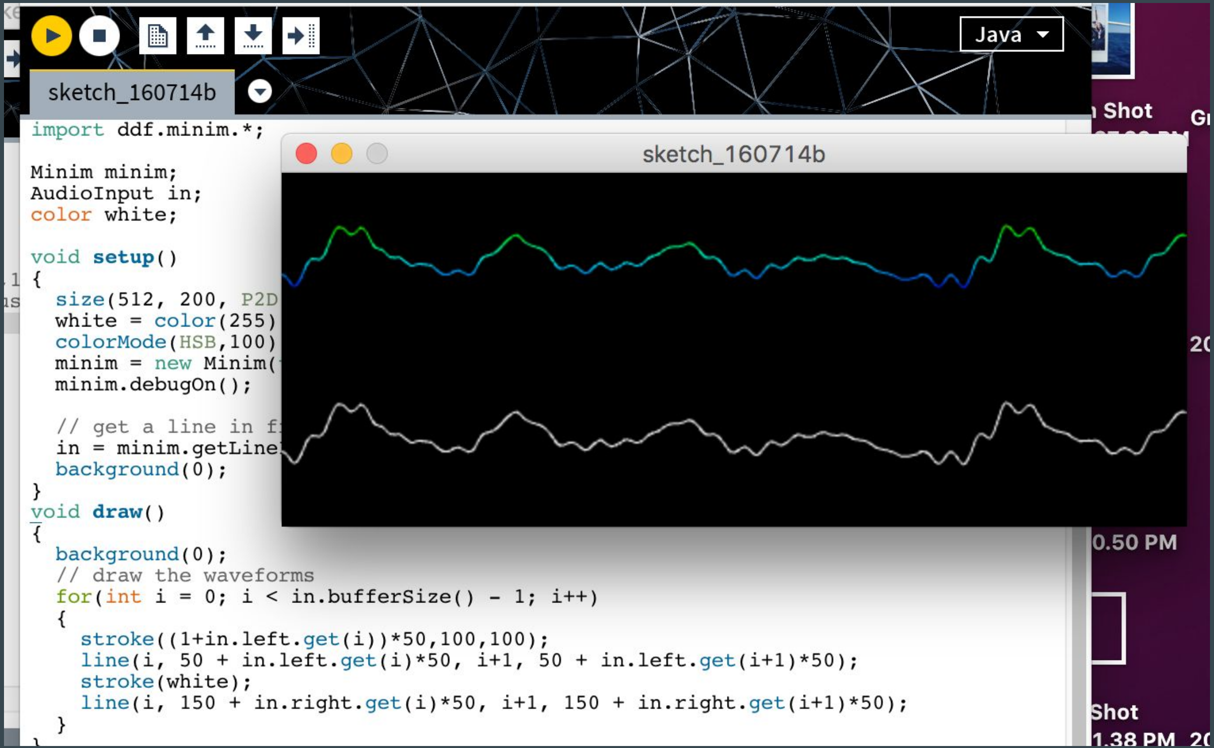
Processing's libraries are extremely helpful and just so easy to use. Interfacing with HW such as serial port communication is just a few simple lines.

ALWAYS CHECK THE DOCS.

And now we play

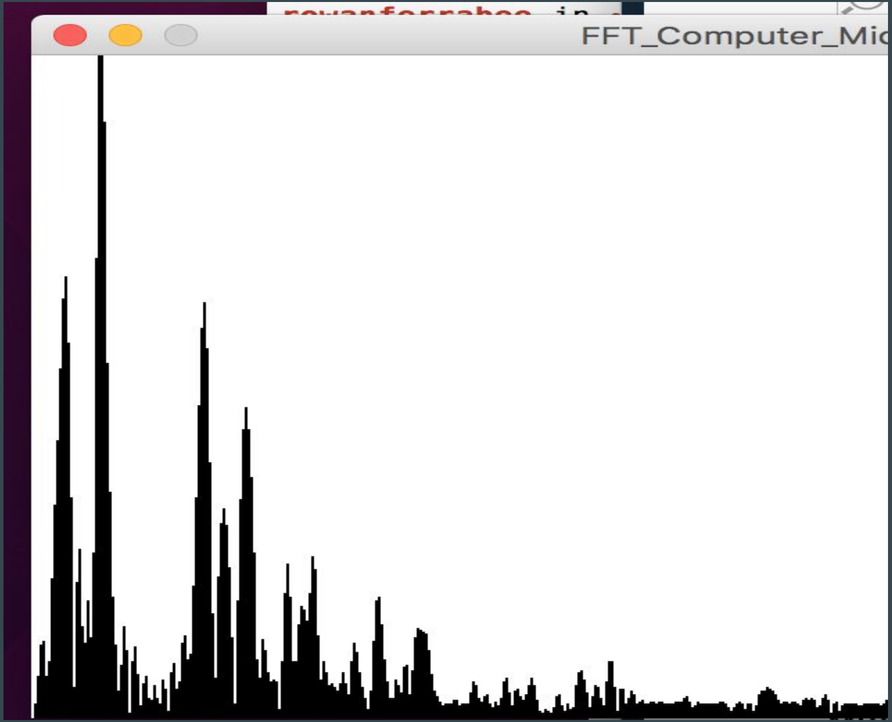
The following are a couple examples of good first projects in Processing as it relates to what we’re doing here at BioTron.

Examples: Displaying Audio Signal from Mic

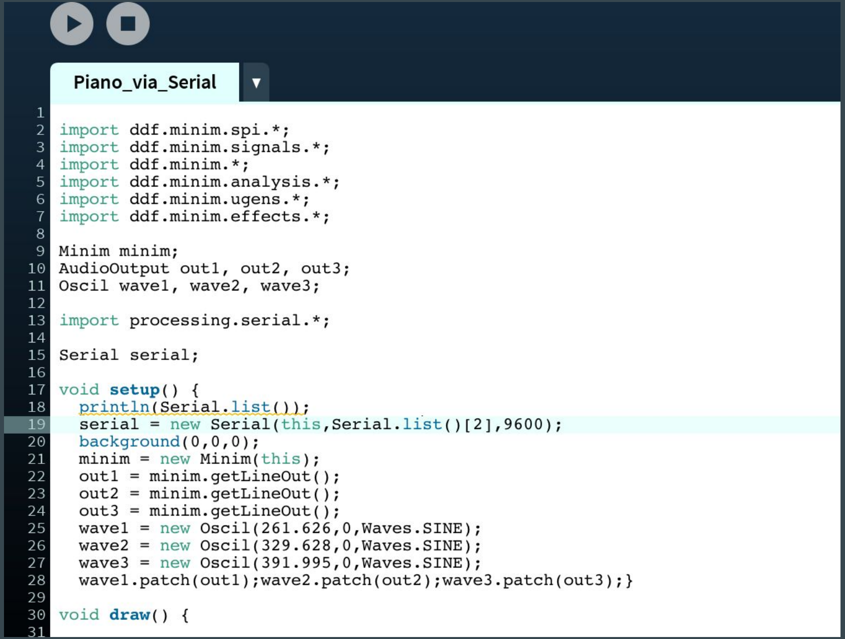


```
import ddf.minim.*;

Minim minim;
AudioInput in;
color white;

void setup()
{
  size(512, 200, P2D
  white = color(255)
  colorMode(HSB,100)
  minim = new Minim(
  minim.debugOn();

  // get a line in f
  in = minim.getLine
  background(0);
}
void draw()
{
  background(0);
  // draw the waveforms
  for(int i = 0; i < in.bufferSize() - 1; i++)
  {
    stroke((1+in.left.get(i))*50,100,100);
    line(i, 50 + in.left.get(i)*50, i+1, 50 + in.left.get(i+1)*50);
    stroke(white);
    line(i, 150 + in.right.get(i)*50, i+1, 150 + in.right.get(i+1)*50);
  }
```

Examples: Spectral Analysis of the Same



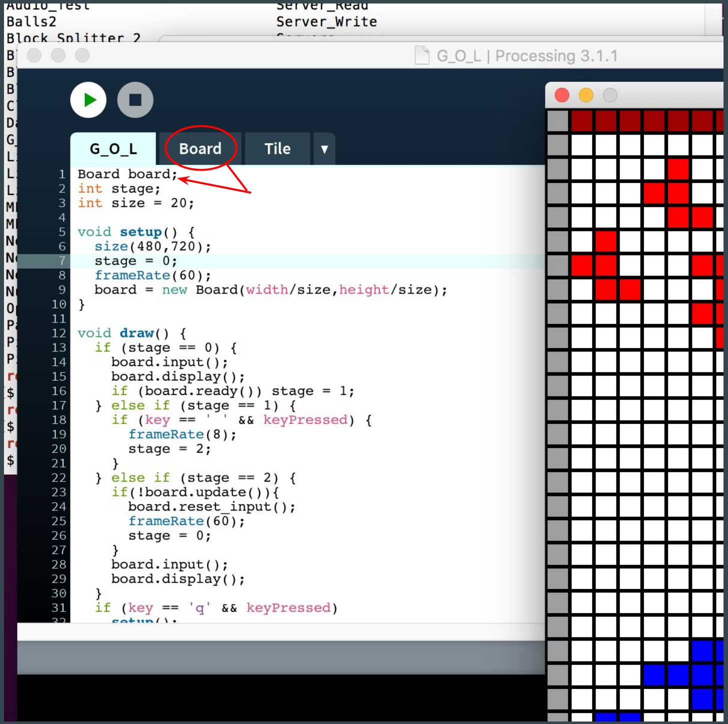
Examples: Serial communication



```

import ddf.minim.spi.*;
import ddf.minim.signals.*;
import ddf.minim.*;
import ddf.minim.analysis.*;
import ddf.minim.ugens.*;
import ddf.minim.effects.*;

Minim minim;
AudioOutput out1, out2, out3;
Oscil wave1, wave2, wave3;

import processing.serial.*;

Serial serial;

void setup() {
  println(Serial.list());
  serial = new Serial(this,Serial.list()[2],9600);
  background(0,0,0);
  minim = new Minim(this);
  out1 = minim.getLineOut();
  out2 = minim.getLineOut();
  out3 = minim.getLineOut();
  wave1 = new Oscil(261.626,0,Waves.SINE);
  wave2 = new Oscil(329.628,0,Waves.SINE);
  wave3 = new Oscil(391.995,0,Waves.SINE);
  wave1.patch(out1);wave2.patch(out2);wave3.patch(out3);}

void draw() {
```

← This thing played audio signals corresponding to buttons pressed on an Arduino jig, but I took apart the jig and it won’t run without it.

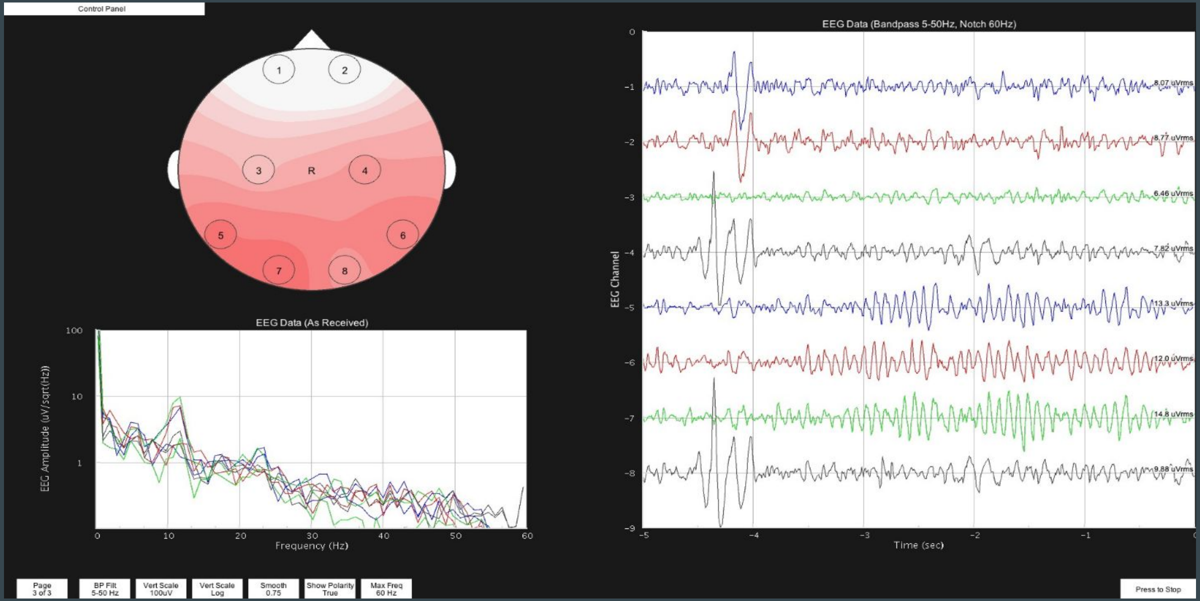
Examples: Basic Encapsulation



```
Board board;
int stage;
int size = 20;

void setup() {
  size(480,720);
  stage = 0;
  frameRate(60);
  board = new Board(width/size,height/size);
}

void draw() {
  if (stage == 0) {
    board.input();
    board.display();
    if (board.ready()) stage = 1;
  } else if (stage == 1) {
    if (key == ' ' && keyPressed) {
      frameRate(8);
      stage = 2;
    }
  } else if (stage == 2) {
    if(!board.update()){
      board.reset_input();
      frameRate(60);
      stage = 0;
    }
    board.input();
    board.display();
  }
  if (key == 'q' && keyPressed)
    setup();
```

And of course, the stock OpenBCI GUI

Playing around with this is both a lot of fun and super helpful to us



Page 3 of 3 | BP Filt 5-50 Hz | Vert Scale 100uV | Vert Scale Log | Smooth 0.75 | Show Polarity True | Max Freq 60 Hz