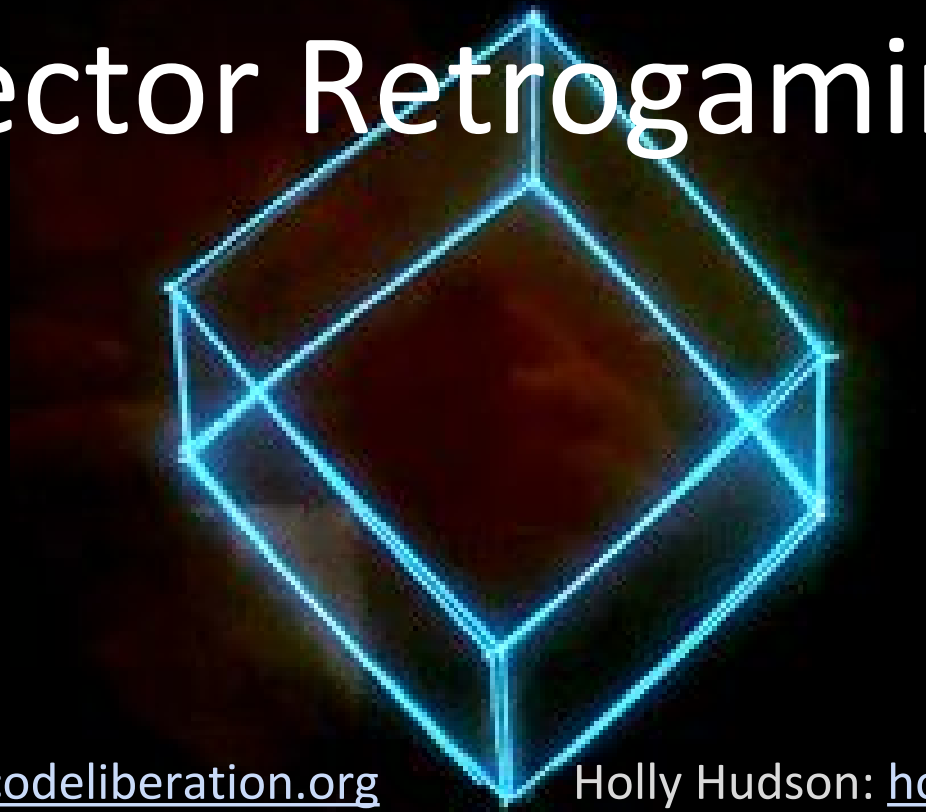
Vector Retrogaming



Adelle Lin: adelle@codeliberation.org
Holly Hudson: holly@nycresistor.com

Who are you?

Tell me about yourself!

What is Code Liberation?

A little about us.

Many popular games are made of:

- **Characters**: beings with which players interact
- **Mechanics & Rules**: systems that determine how players interact with the game
- **Goal**: what players must do to finish the game
- **Story**: narrative driving the game's goal & rules

Preparation

Get the starter code for today's game on GitHub!

The URL: https://github.com/adellelin/vectrexGame.git

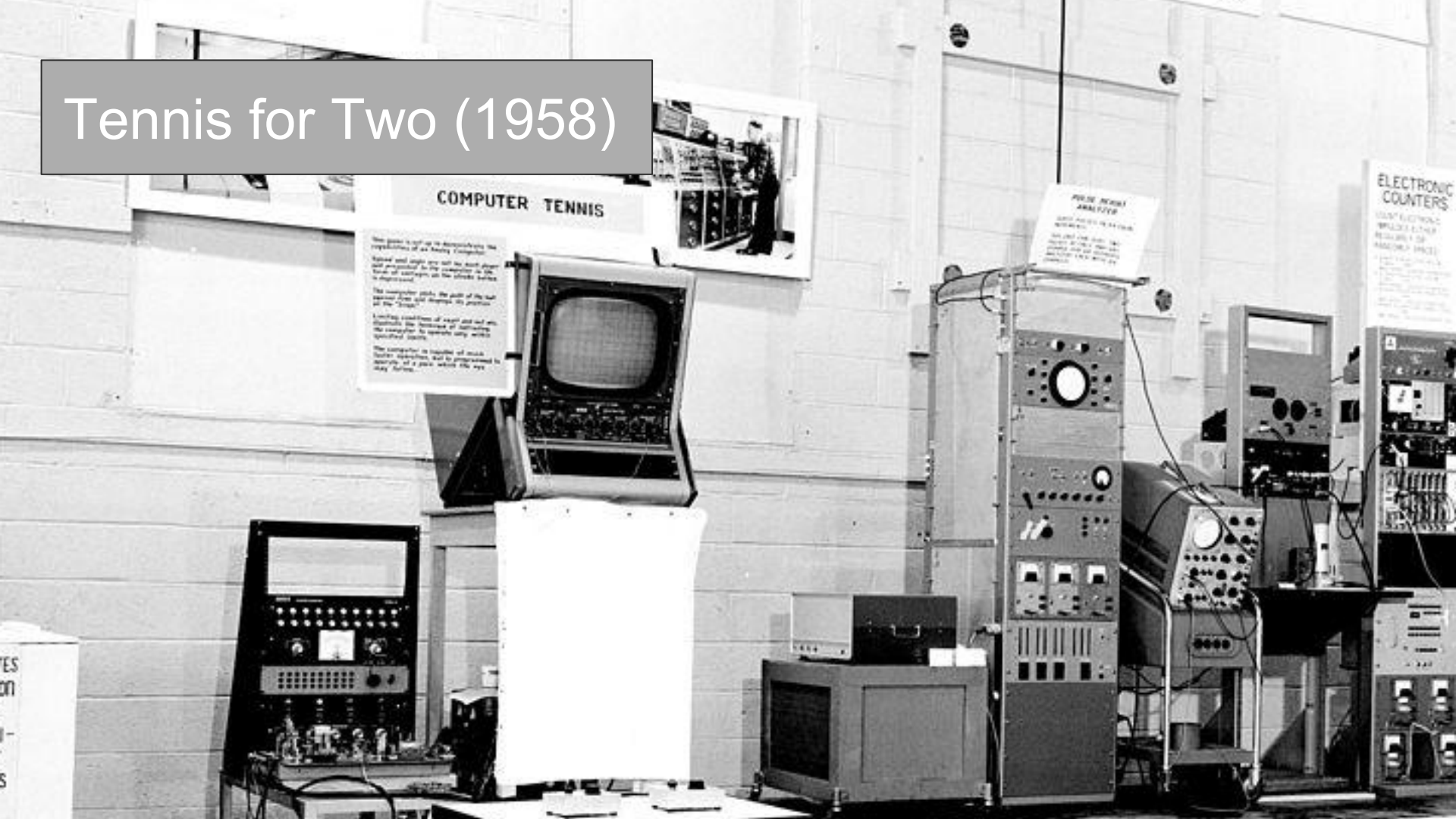
Click the **Download ZIP** button on the right side.

What is a Vector Display?

Analog Oscilloscope

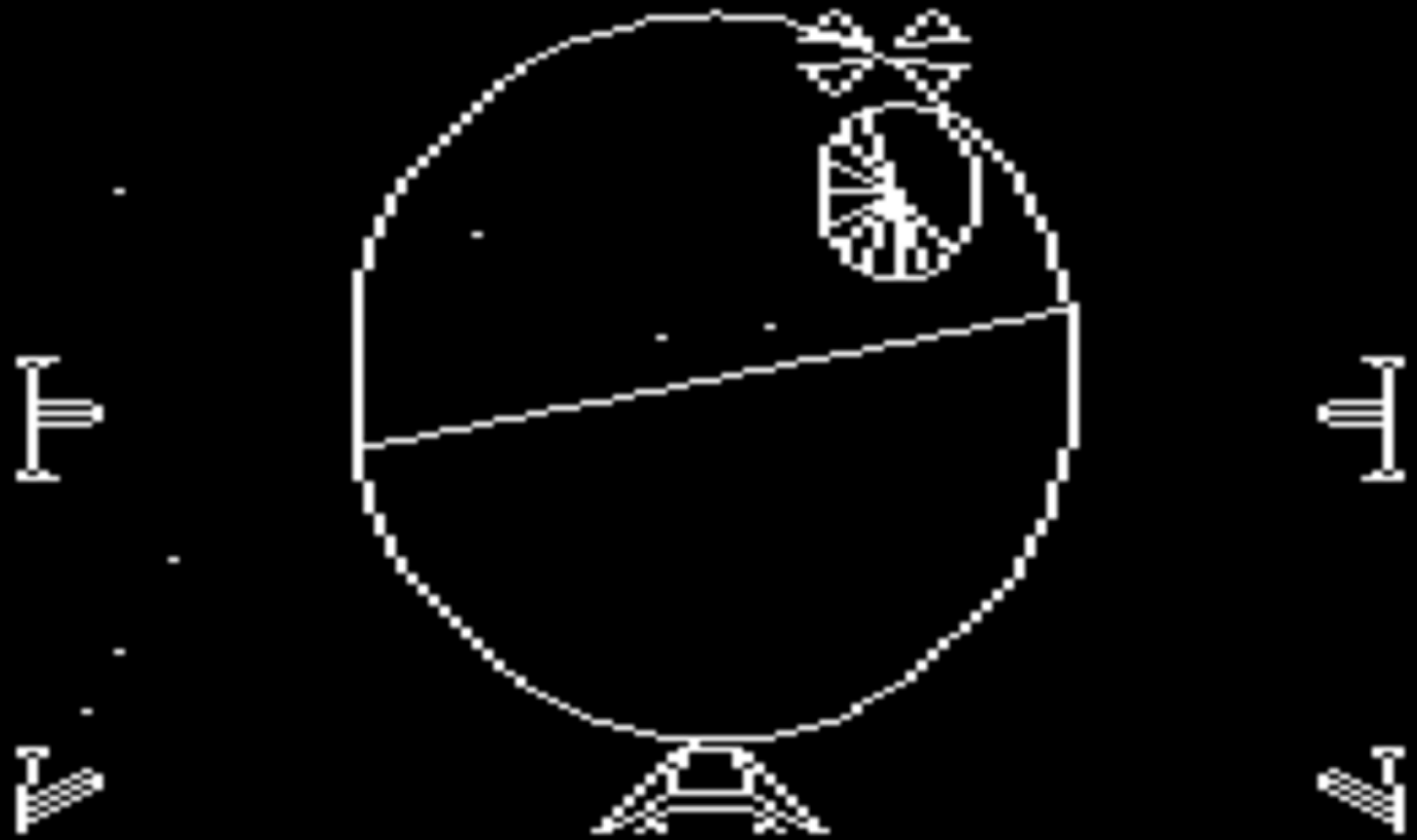
Tennis for Two (1958)

Tennis for Two (1958)



SCORE 03 WAVE
00009495 SHIELDS GONE

SHOOT TIE FIGHTERS



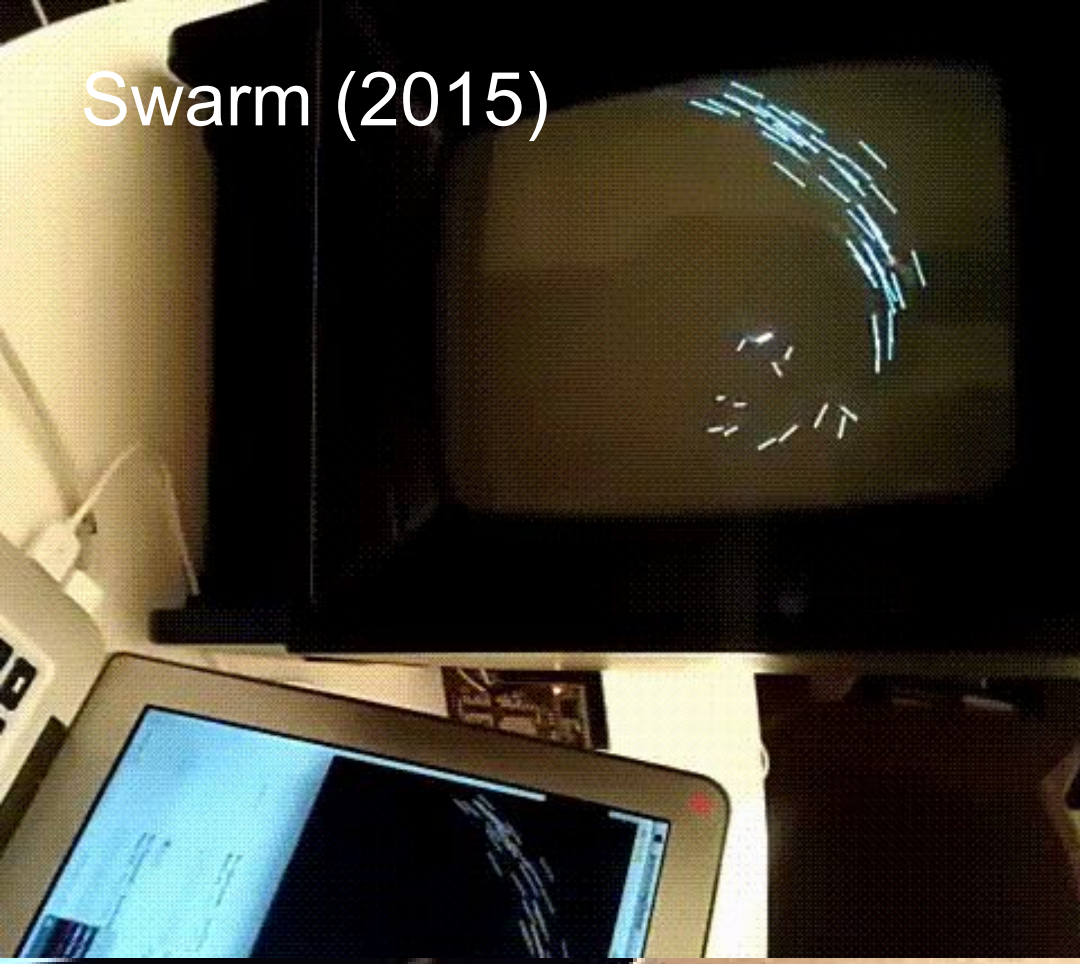

SCORE
2033
8
SHIELD
1 WAVE
MAY
THE
FORCE
BE
WITH
YOU

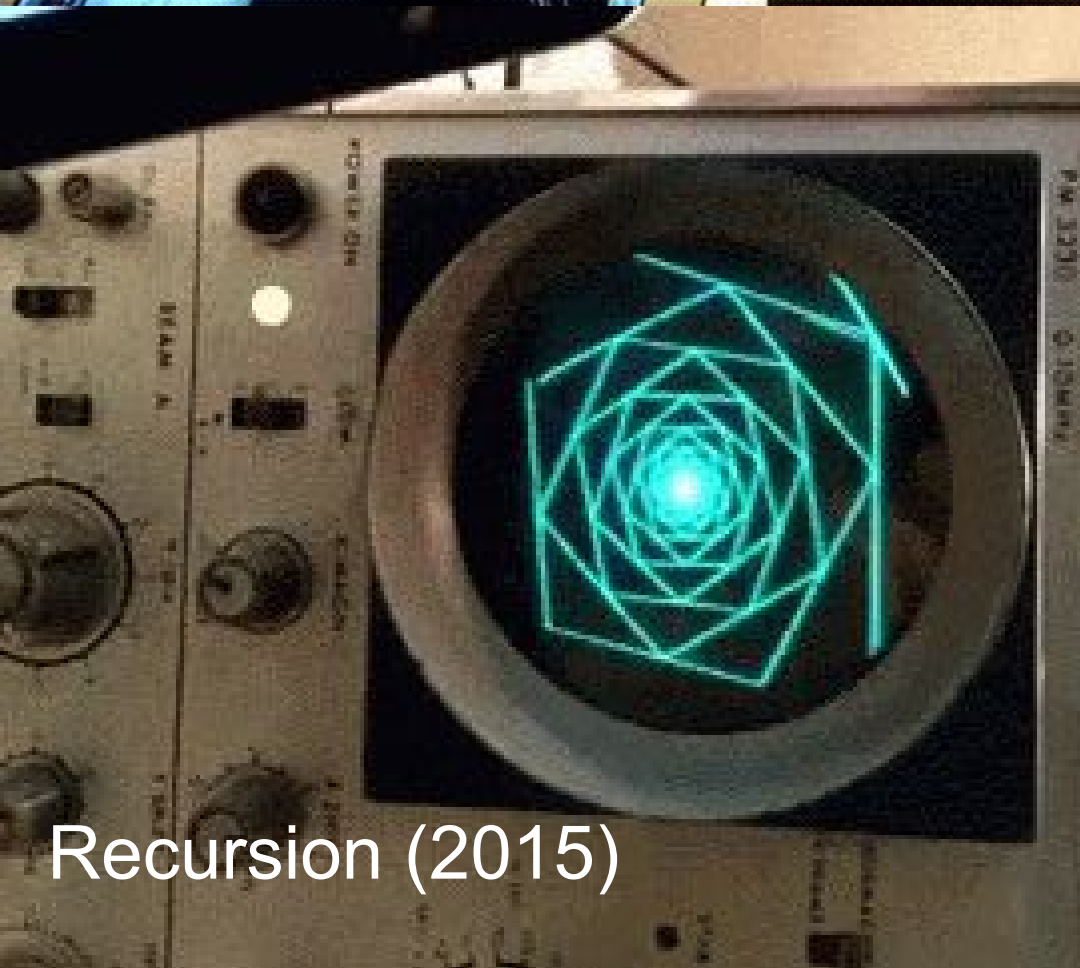

002800


SCORE
00
8
SHIELD
1 WAVE
SHOOT TIE FIGHTERS


BATTLE ZONE
ATARI 1980

“Vec9” (2015)

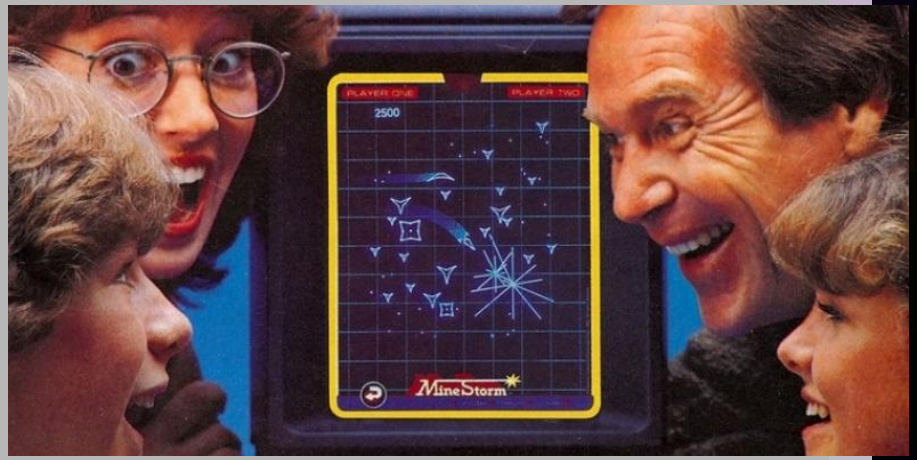

Swarm (2015)


Recursion (2015)

“Space Rocks” (2013)

Vectorscope (with “audio” XY mode)

Vectrex console



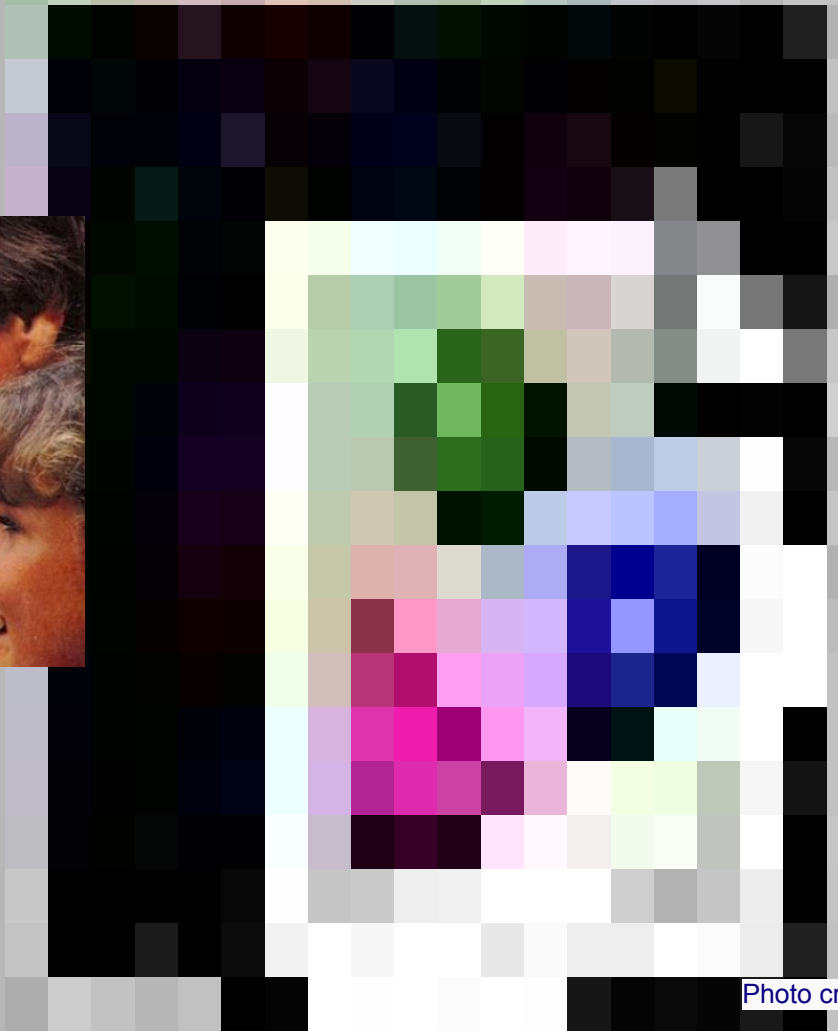
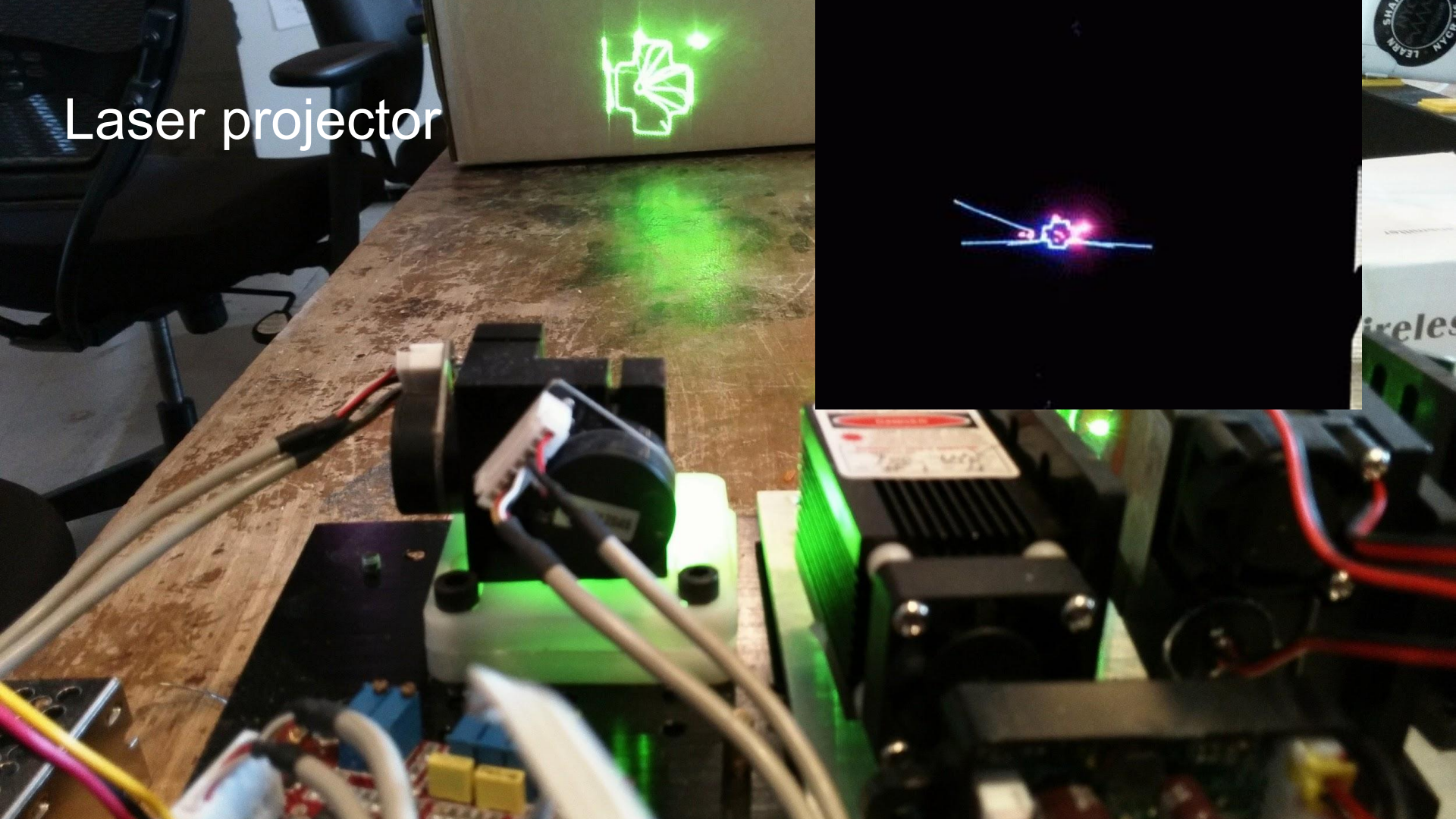


Photo credit: Evan-Amos

Laser projector



What is processing?

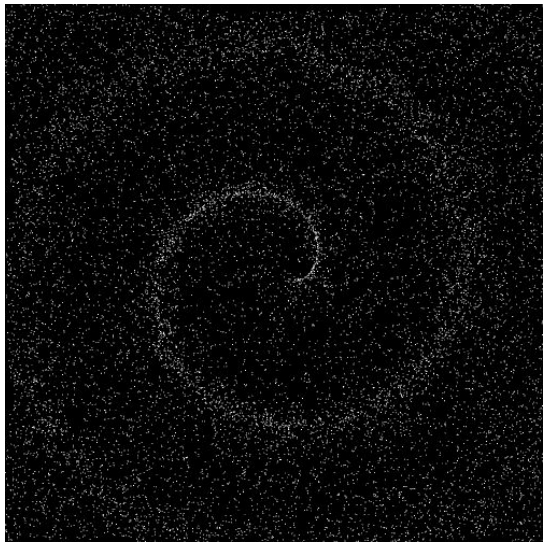
Processing is a library for Java (a programming language)

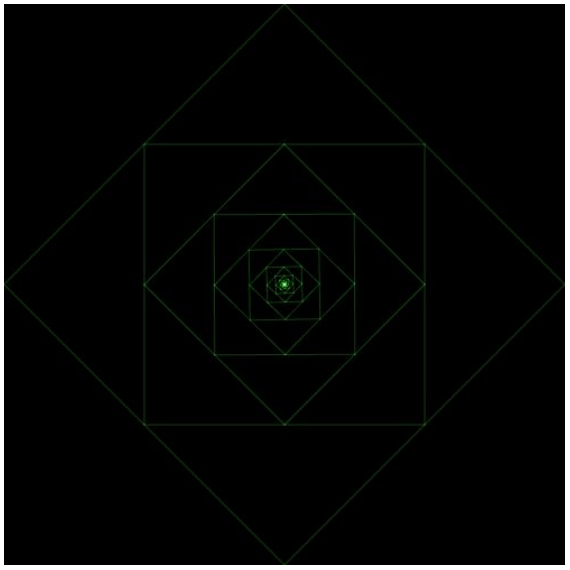
- Library: a collection of code intended to simplify coding process and/or give it more functionality

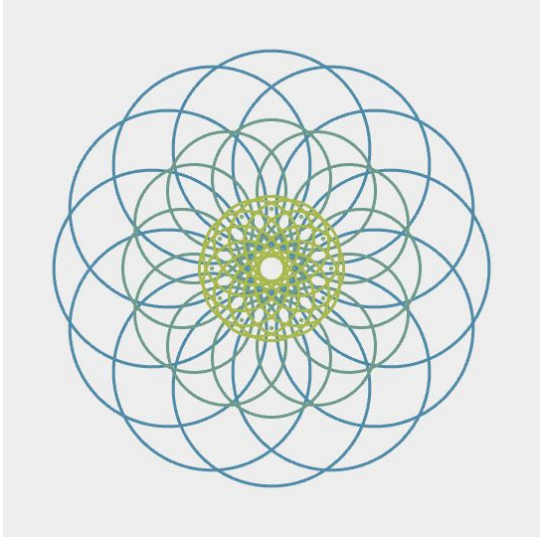
We'll be working in the Processing environment (the app), which is also referred to as an IDE

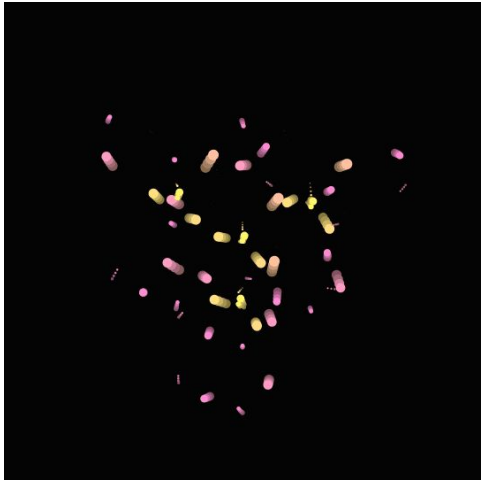
Processing

artist: http://jacobjoaquin.tumblr.com/









artist: http://www.sasj.nl/

What is processing?

An IDE is a an Integrated Development Environment

- an application that allows you to write code in this language

Processing interface consists of:

- space to write code
- compiler that lets your run the sketch
- debugger that notifies you of errors in your code

What is processing?

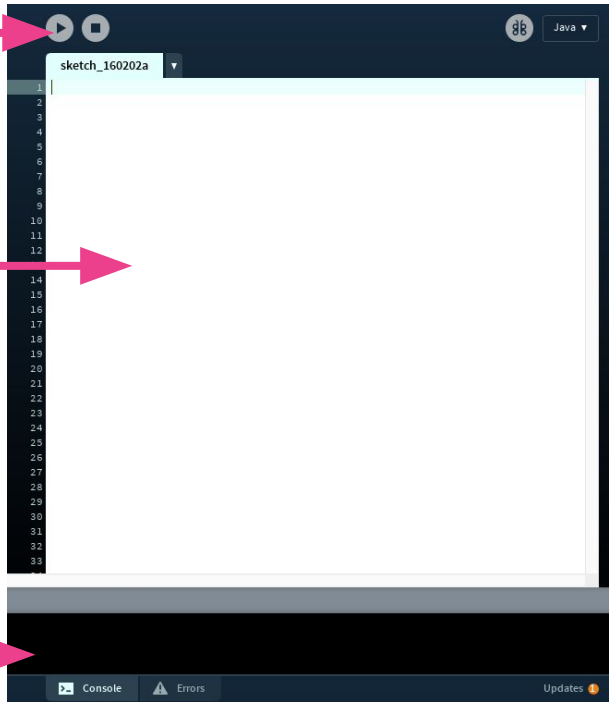
Baking a cake

1. recipe
2. bake
3. taste-test

2. compile

1. write code

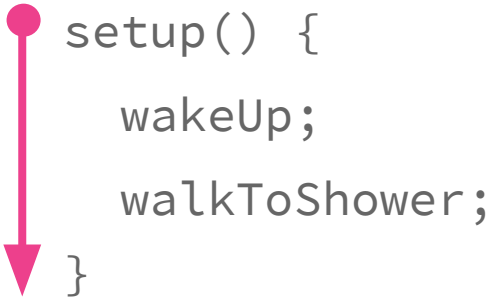
3. debug



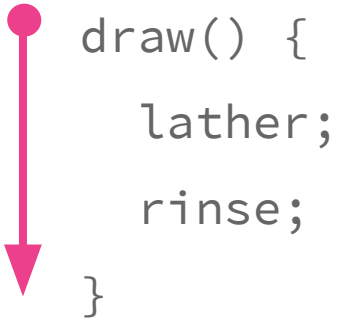
Structure of Processing

Commands **usually** require ; at the end.

Code in `setup()` runs once from top to bottom, and code in `draw()` runs in a loop.



```
setup() {
  wakeUp;
  walkToShower;
}
```



```
draw() {
  lather;
  rinse;
}
```

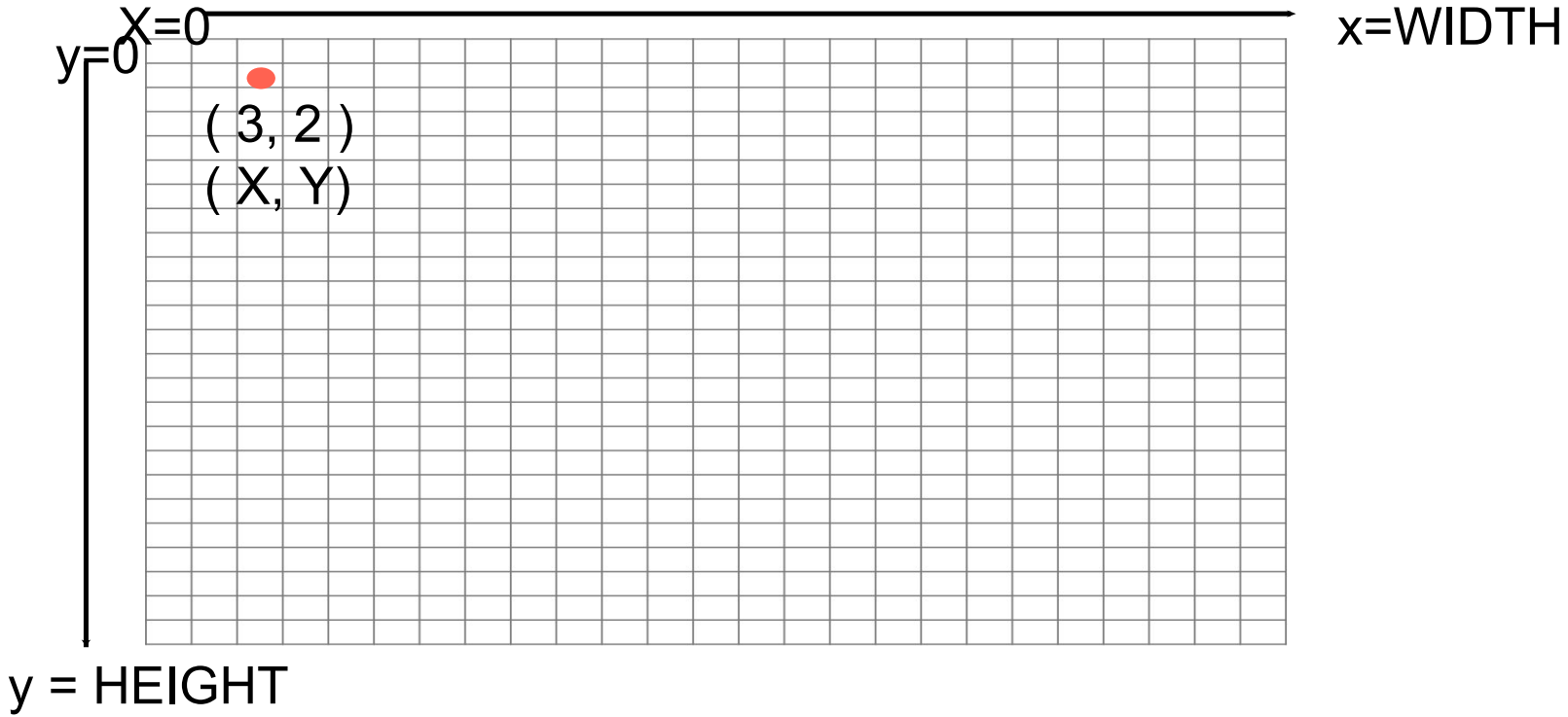
Let’s code!

Structure of Processing

Setup your drawing canvas and beginning conditions

```
setup () {
    size(500, 500);
    fill(0);
    stroke(255);
}
```

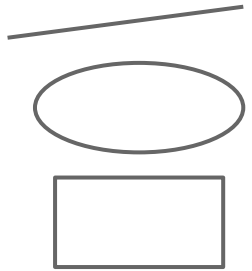
Positioning within the canvas



Make some shapes

Drawing basics

```
draw () {
    line(x1, y1, x2, y2);
    ellipse(x, y, width, height);
    rect(x, y, width, height);
}
```



Variables

In programming, the concept of variables is important.

A variable is like a bucket -- it holds a value. The = sign does not mean “equals”, it’s used to put a value in the bucket, or *assign* a value to a variable. Read it “=” as “holds”.

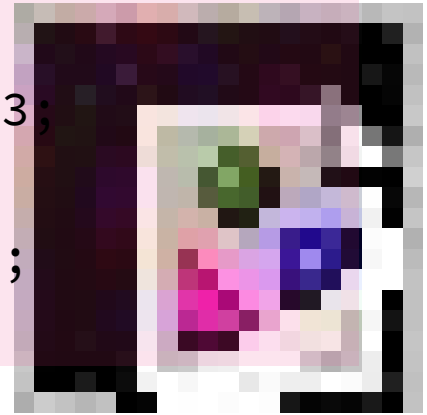
```
x = 2 + 4;
```
x holds the value 6.

```
y = x + 1;
```
y holds the value 7.

Variables

Variables hold many **data types**: int, float, arrays, objects, booleans and more. So you can declare most types as variables:

```
int kittens = 18;
float kittenWeight = 17.3;

boolean isKitten = false;
```



Variables

What do you do once you have put something in your bucket?

Conditional Statements

You may want to test / check for a certain value.

We can do this in a conditional statement. Such as an If statement. Conditions we can test (>, <, >=, <=, !, ==)

```
if (x == 6) {
    do that dance;
}
```

```
if (garden.size() < 5) {
    garden.add(new Flower());
}
```

Booleans

Booleans is a data type that returns two values --> true or false.

Booleans can be used like switches.

```
    boolean dancing = false;

void musicStarts() {
    if (!dancing) { // ! denotes not
        moveToTheBeat();
        dancing = true;
    } else {
        tired();
        dancing = false;
    }
```

for Loops

for loops are extremely useful. They let you run a block of code a certain number of times allowing you to go through a list of items in a collection.

```
order: initialize; test; increment
```

```
for (invitesSent = 0; invitesSent < numFriends; invitesSent++) {
    create invite;
    send invite to friend;
}
```

Commenting and Print to console

Always comment your code!

// to comment one line

/* for longer comments,
use a slash and asterisk, then
mirror to end the comment */
Shortcut: highlight a block of code then hit cmd + /

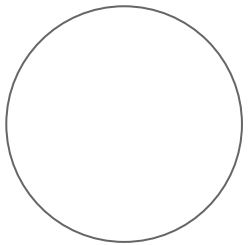
```
println(invitesSent);  // this prints to the debugger
```

Sprites

Drawing sprites - objects

In the case of the vector display. We need to draw all our own sprites. We do this using object classes, which stores the object so we can reuse it later within our sketch.

Shape



Object

Constructing an Object Class

Name your Class

Declare the properties

Write a constructor for building objects

Write the functions (things it does)
- display
- update

Constructing an Object Class

```
class Flower {
  float xpos;
  float ypos;
  int num_petals;
  float scale_size = 1;

  // The constructor
  Flower(float tempxpos, float tempypos, int tempnum_petals, ..) {
     this.xpos = tempxpos;
     this.ypos = tempypos;
     this.num_petals = tempnum_petals;
  }

  void display() {
     pushMatrix();
     // ...more code
     popMatrix();
  }
}
```

Set your parameters

Store as local variables

Draw your object

Arrays

Arrays hold several things: usually we start by defining the type

```
int[] scores;
```

100	50	88	92	100	97	51	30	100	100	90	72
0	1	2	3	4	5	6	7	8	9	10	11

The value of `scores[7]` is 30.

```
int[] scores = {100, 50, 88, 92, ..}; // make an array of scores
```

We'll learn:

- Setting up processing for the Vectrex
- Drawing our sprites
- Using ArrayLists
- Basic collision
- Adding sound to your game

ArrayList <>

We'll use a special type of array called an ArrayList, which is part of the collections framework. It has the benefits of auto-resizing, and you can copy or remove objects easily.

Declare it:

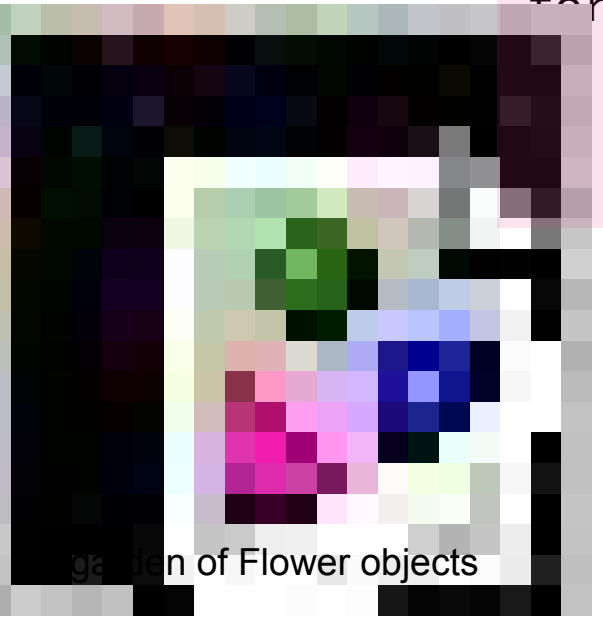
```
ArrayList<type> variable_name = new ArrayList<Object>();
ArrayList<Flower> garden = new ArrayList<Flower>();
```

Use it:

```
garden.add(new Flower());
garden.remove(oneSpecificFlower);
```

Let's display all the objects in an ArrayList

```
for (int i = 0; i < garden.size(); i++) {
    Flower blossom = garden.get(i);
    blossom.
```



garden of Flower objects

display one blossom of type Flower

the VST.pde library

- sends processing code over serial to the vector displays
- brightness
- clipping
- custom drawings commands that overwrites processings native drawing library

Jake Joaquin

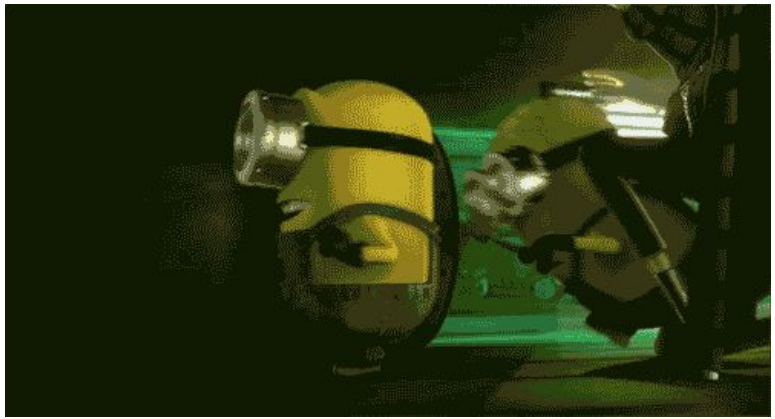
Adelle Lin

Trammell Hudson

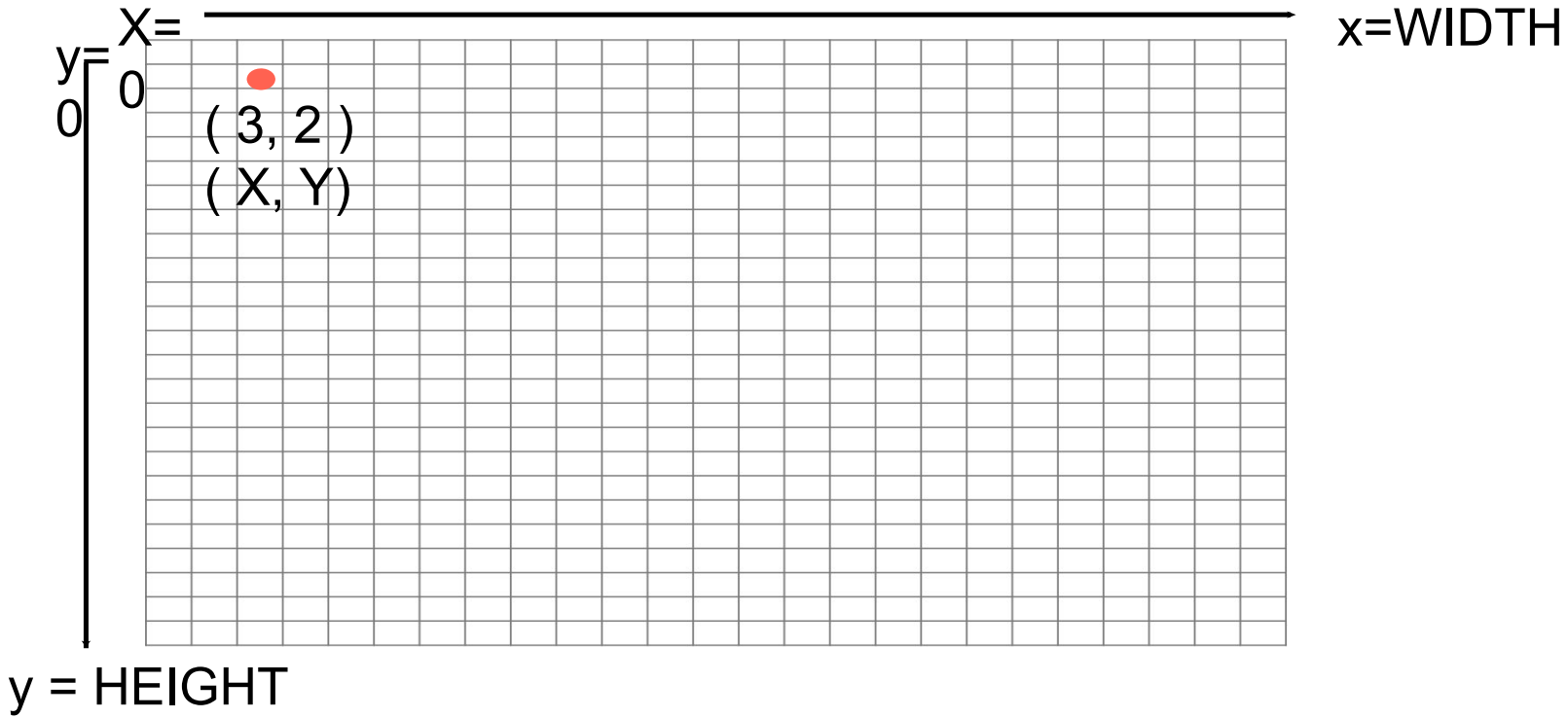
Setting up your processing file

```
Vst vst;

void setup() {
  size(512, 600, P2D);
  vst = new Vst(this, createSerial());
  blendMode(ADD);
  noFill();
  stroke(212, 128, 32, 128);
  frameRate(25);
}

void draw(){
     vst.display()
}
```

Movement and Collisions!



Let's talk about Vectors

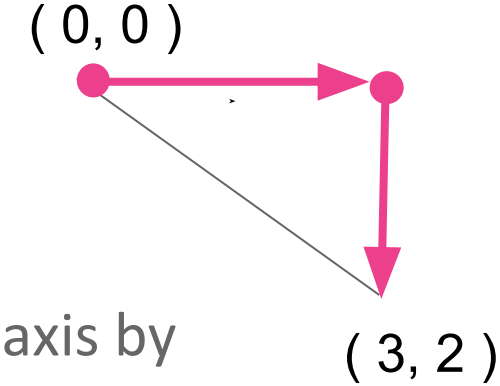


Let's talk about Vectors

Vectors are a handy way to store your x and y position. They are also a way to think about movement.

```
PVector boogie = new PVector (3, 2);
```

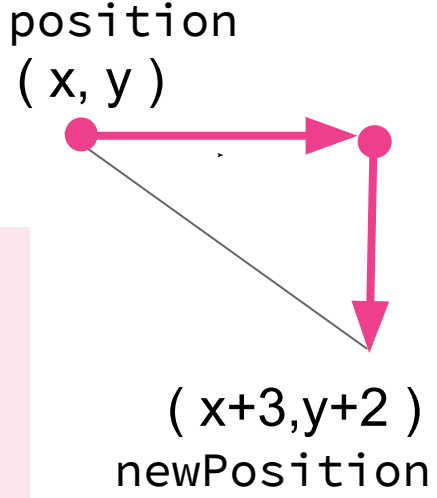
X, Y positions



In this case, we would have moved along the x-axis by 3 steps, and moved along the y-axis by 2 steps

Do some vector motion

A simple way to give your object movement is to use vector additions. We can do this within an update function in the object class




```
void update(){
    PVector velocity = new PVector (3, 2);
    newPosition = position.add(velocity);
}
```

Collisions

A collision is a test to see if two objects have made contact.

We can test for this by checking the distance between two moving objects.

```
float closeness = dist(obj1.xpos,
    obj1.ypos, obj2.xpos, obj2.
xpos);
if(closeness < 5) {
    kiss;}
```