Animation

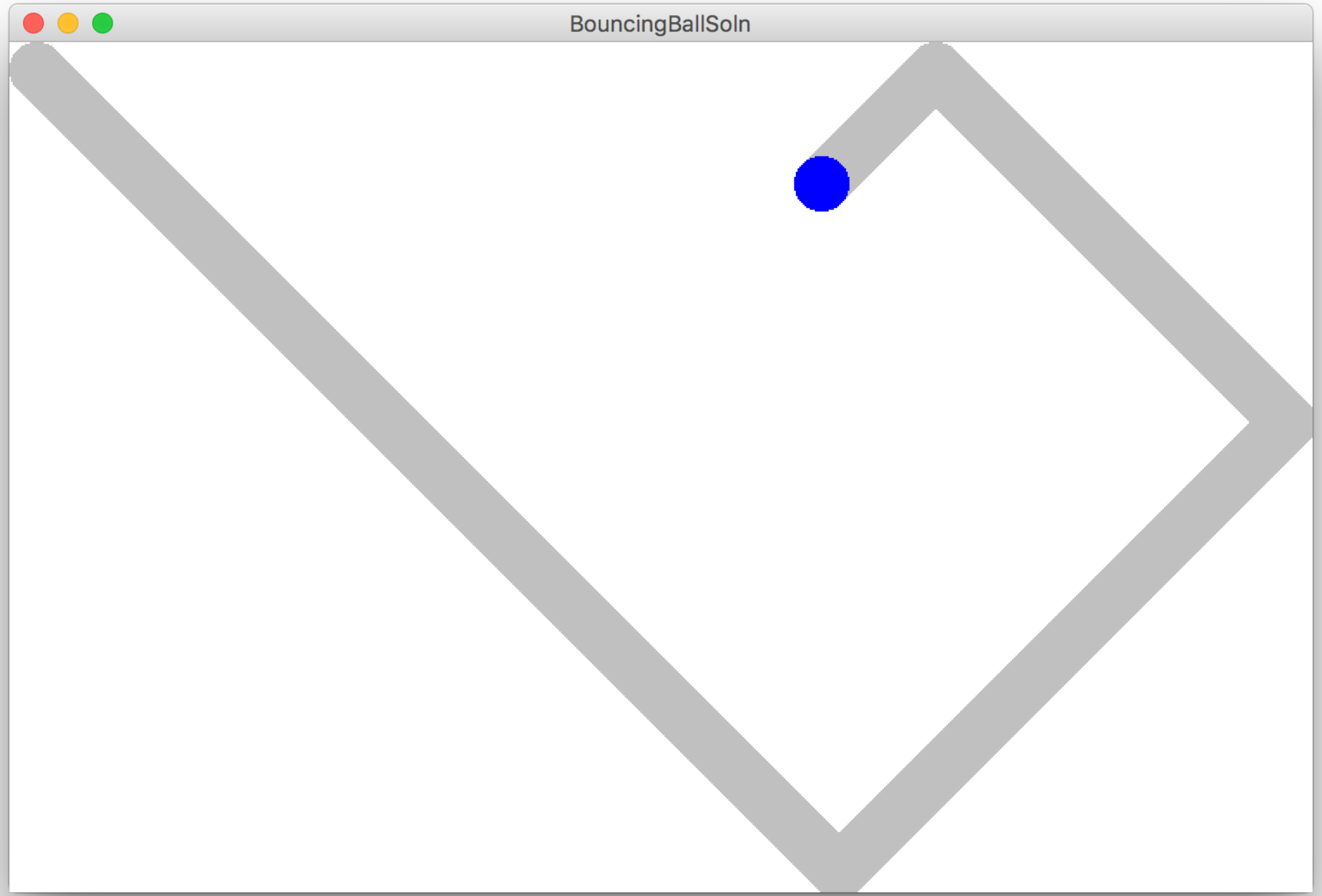
Chris Piech
CS106A, Stanford University

This is Method Man. He is part of the Wu Tang Clan. ☺

Learning Goals

1. Feel more confident writing methods
2. Write animated programs

You will be able to write Bouncing Ball



Defining a Method

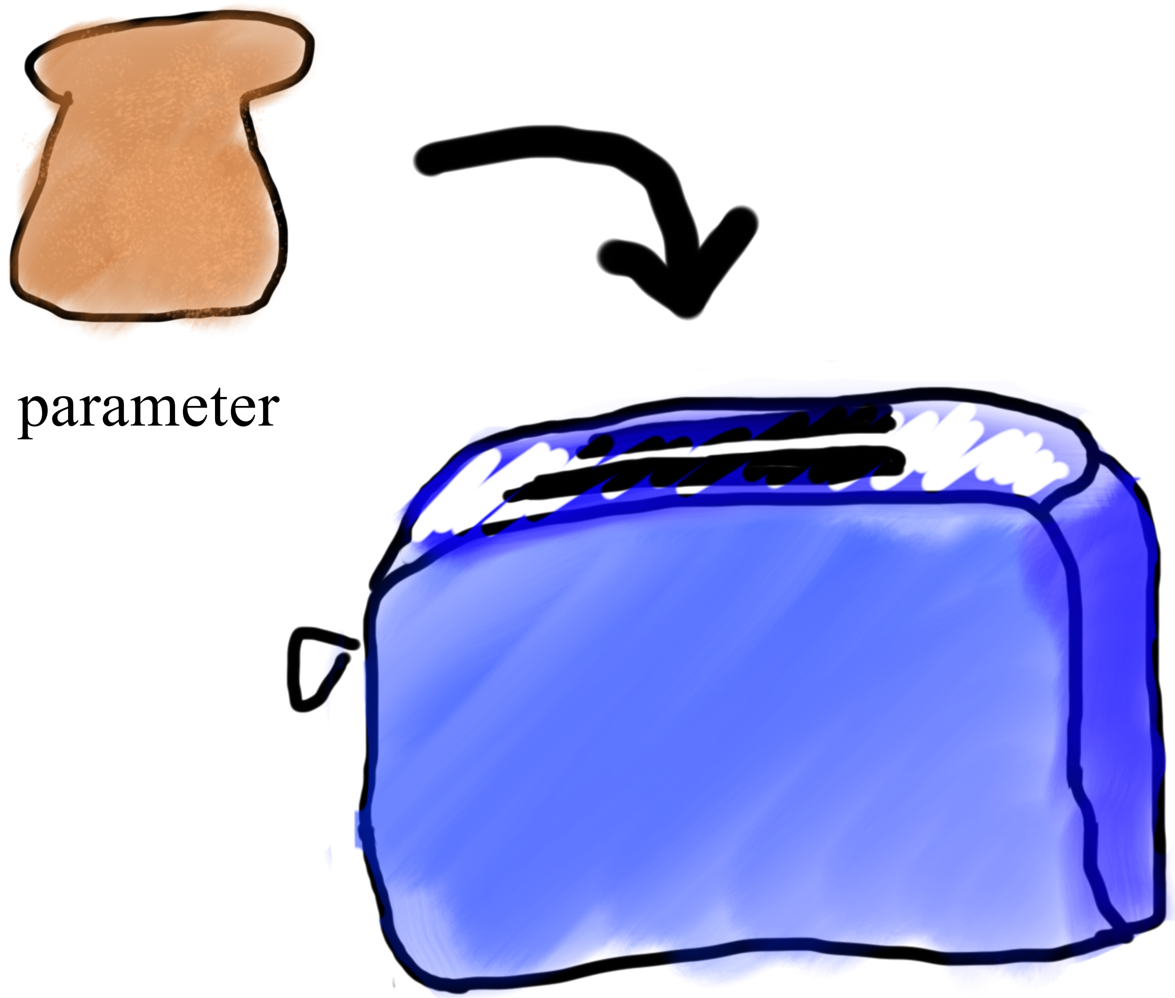
```
private void turnRight() {
    turnLeft();
    turnLeft();
    turnLeft();
}
```

Methods are Like Toasters

Methods are Like Toasters

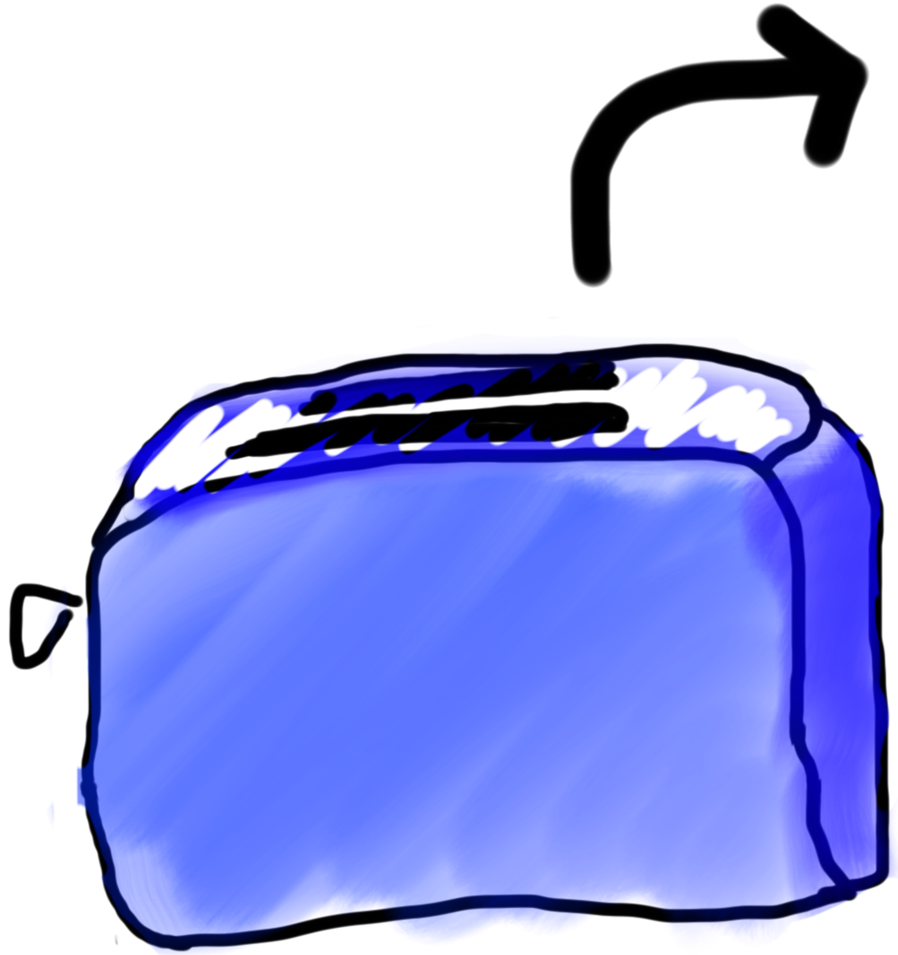
parameter

Methods are Like Toasters

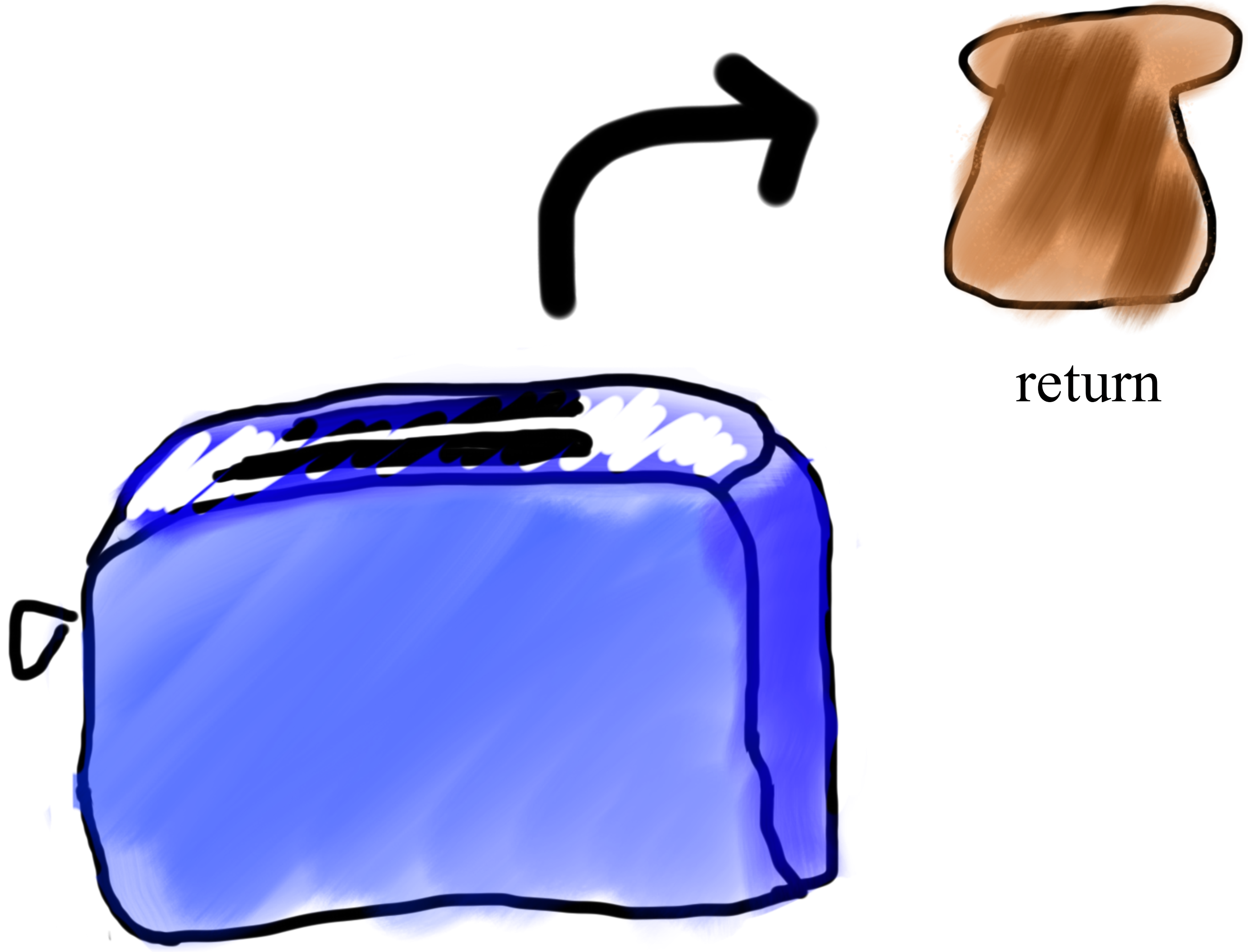


Methods are Like Toasters

Methods are Like Toasters

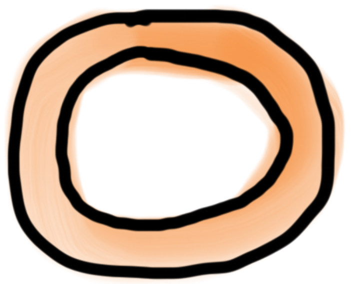


Methods are Like Toasters

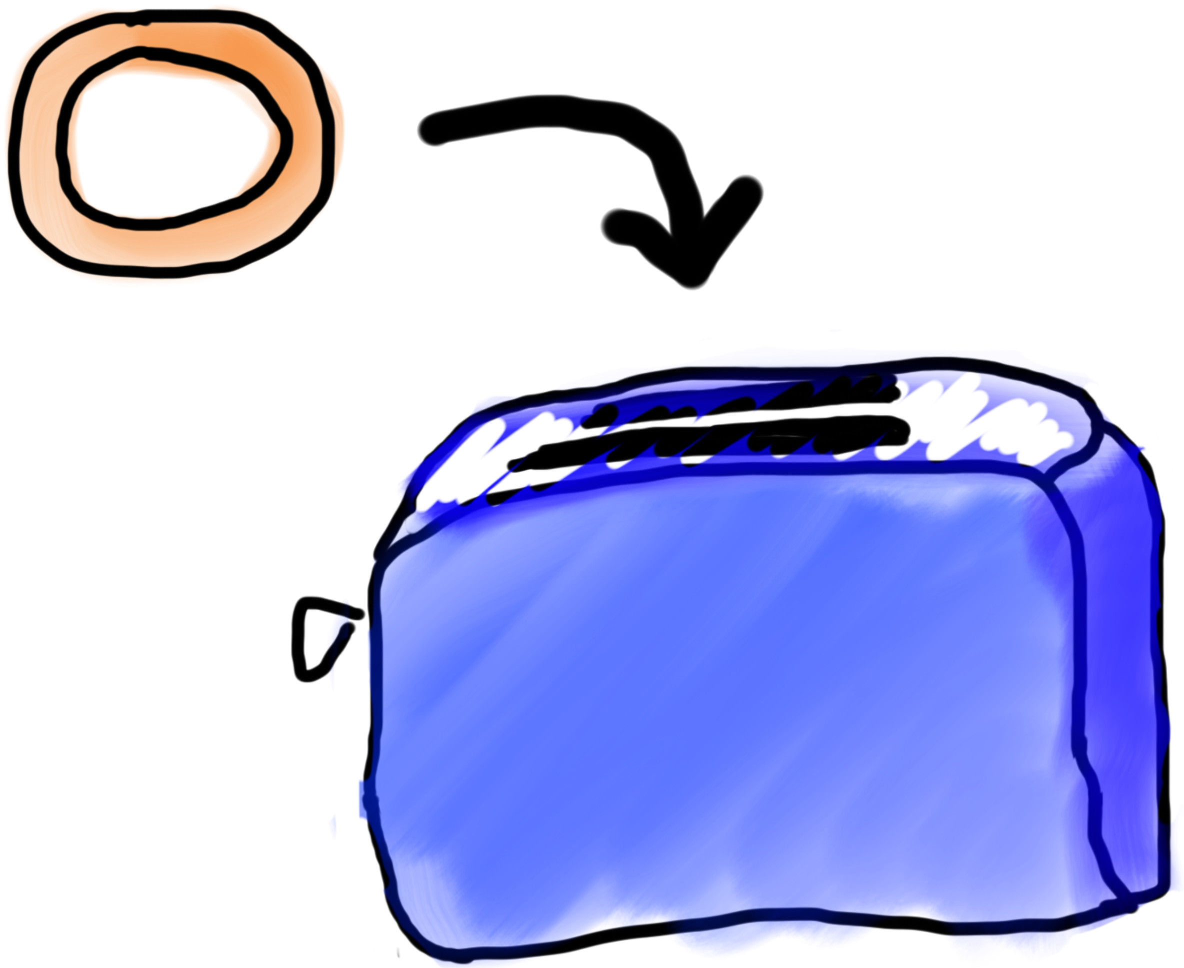


Methods are Like Toasters

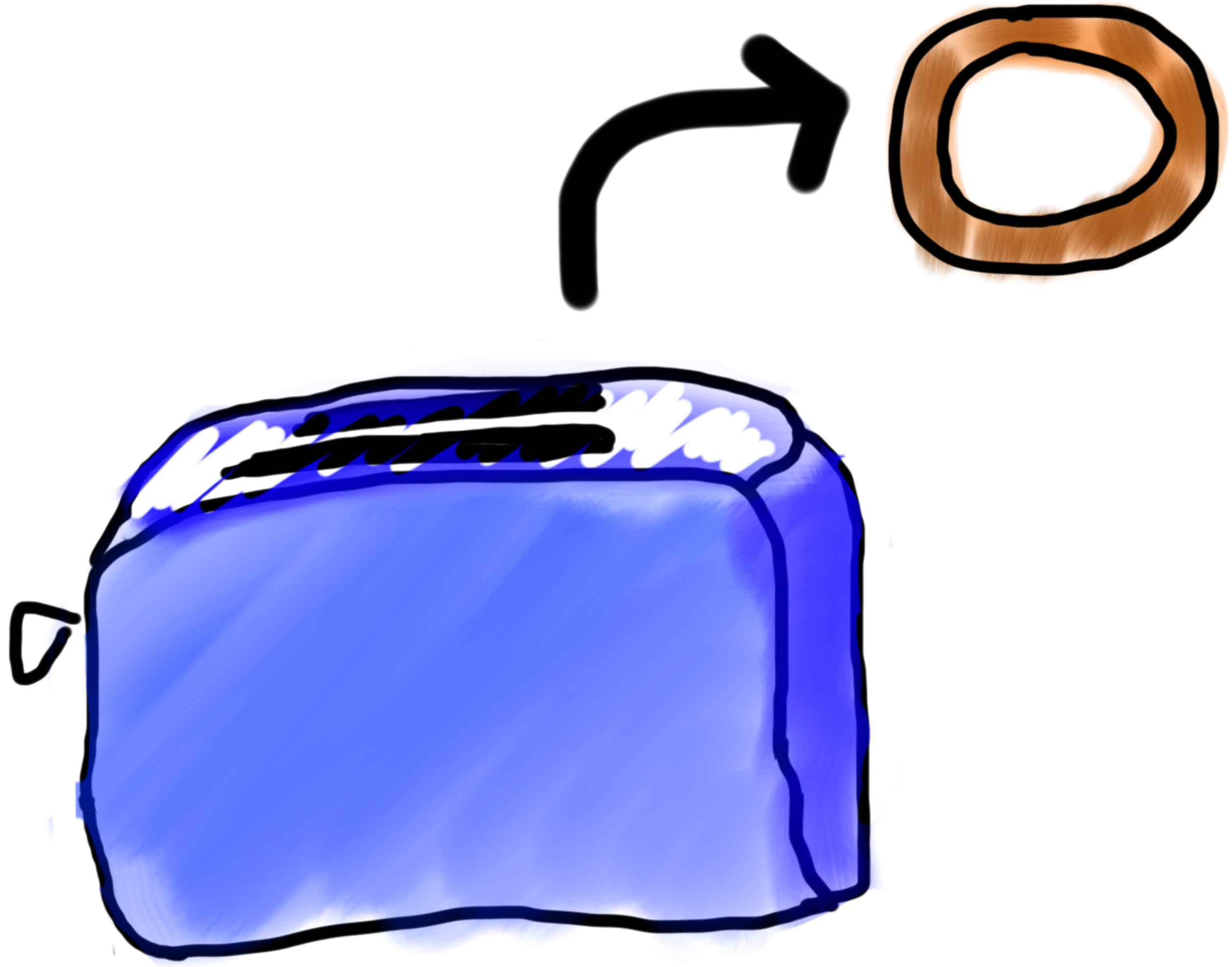
Methods are Like Toasters



Methods are Like Toasters

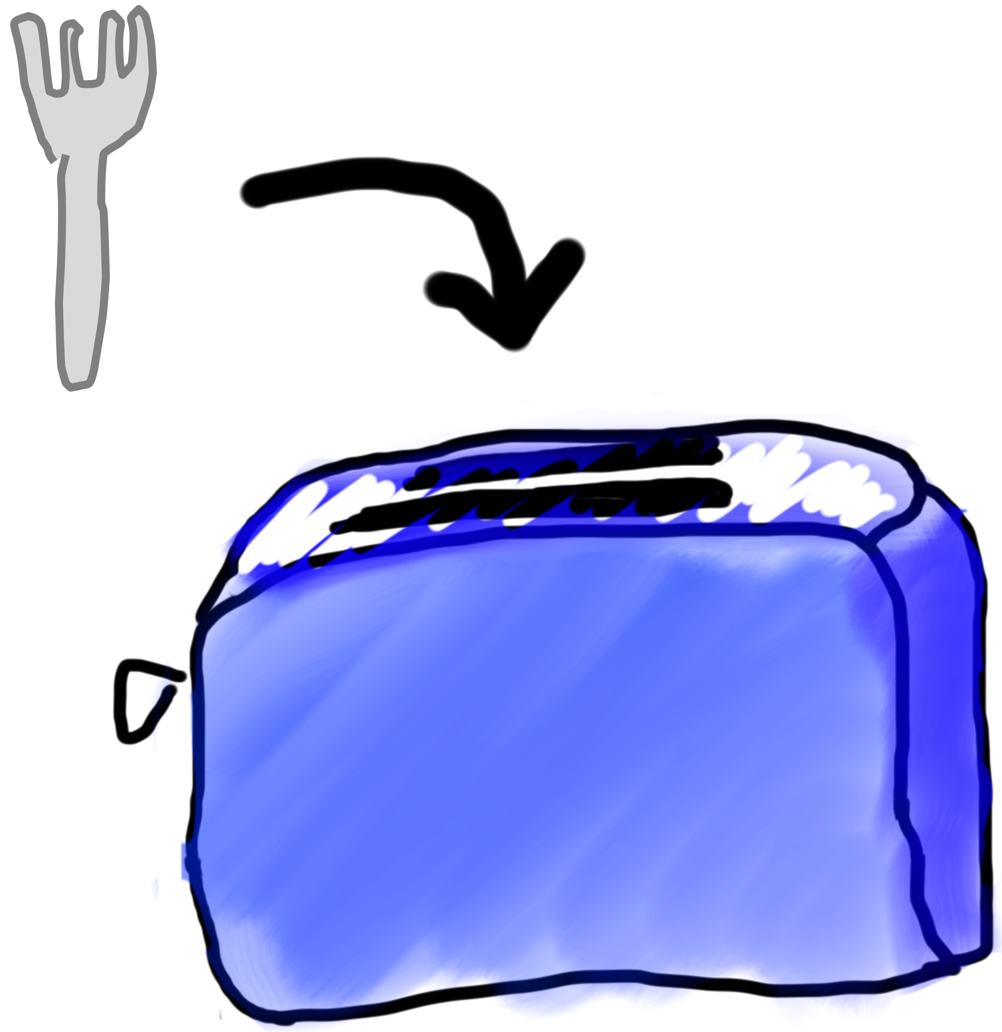


Methods are Like Toasters



Methods are Like Toasters

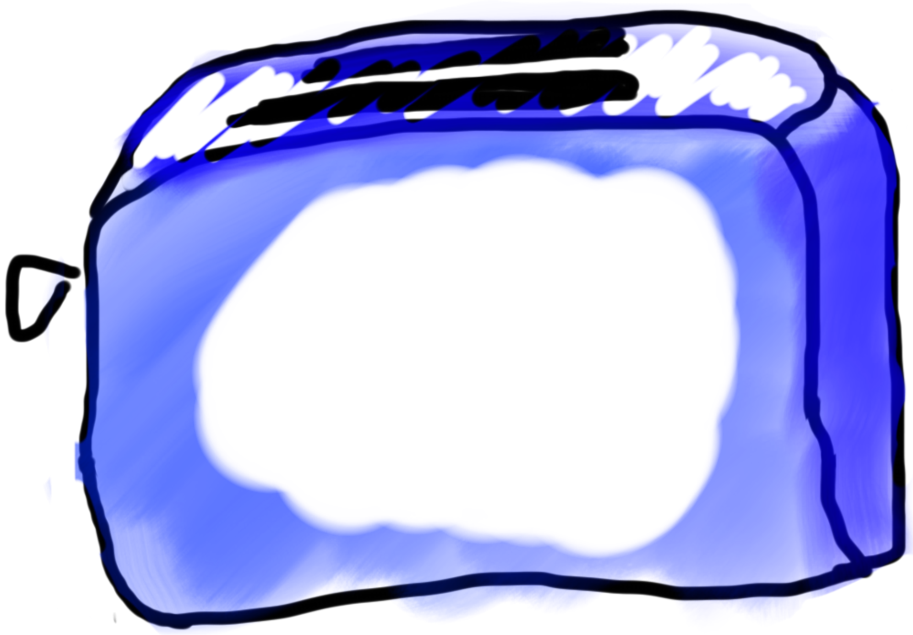
Methods are Like Toasters



Methods are Like Toasters

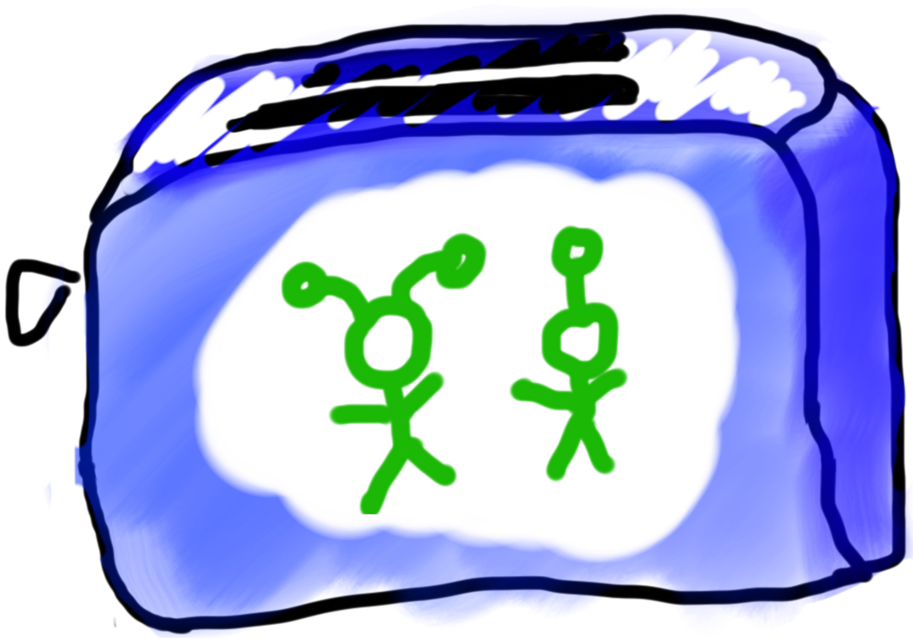
Methods are Like Toasters

Methods are Like Toasters

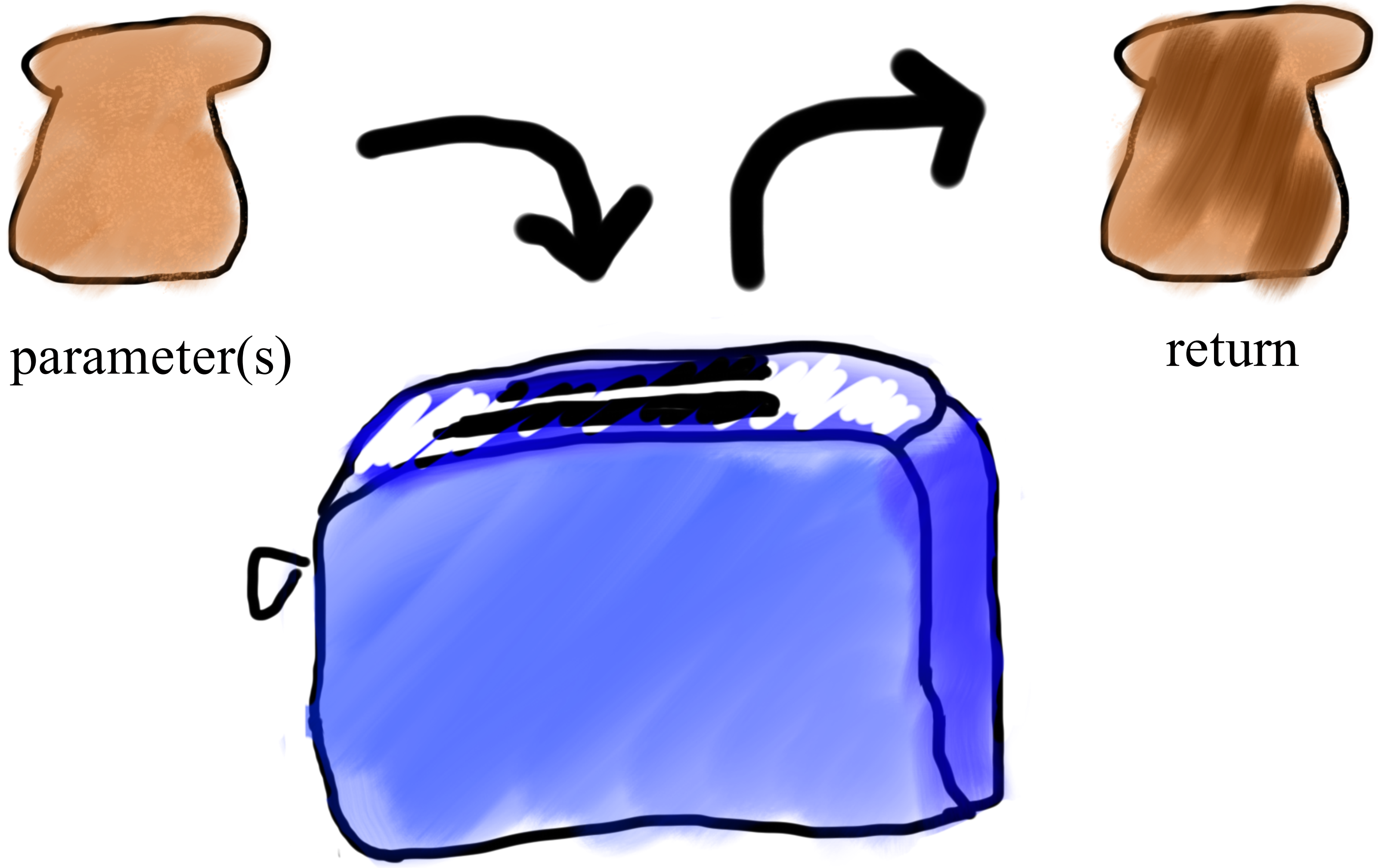


Methods are Like Toasters

Methods are Like Toasters



Methods are Like Toasters



Anatomy of a method

```
public void run() {
    double mid = average(5.0, 10.2);
    println(mid);
}


private double average(double a, double b) {
    double sum = a + b;
    return sum / 2;
}
```

Anatomy of a method

```
public void run() {
    double mid = average(5.0, 10.2);
    println(mid);
}
```

method "definition"

```
private double average(double a, double b) {
    double sum = a + b;
    return sum / 2;
}
```

Anatomy of a method

```
public void run() {
    double mid = average(5.0, 10.2);
    println(mid);
}
```

Return Type

Parameters

```
private double average(double a, double b) {
    double sum = a + b;
    return sum / 2;
}
```

Do not confuse **return type** with **`println`**. Both are “outputs” of sort.

Anatomy of a method

```
public void run() {
    double mid = average(5.0, 10.2);
    println(mid);
}
```

name

```
private double average(double a, double b) {
    double sum = a + b;
    return sum / 2;
}
```

Anatomy of a method

```
public void run() {
    double mid = average(5.0, 10.2);
    println(mid);
}


private double average(double a, double b) {
    double sum = a + b;
    return sum / 2;
}
```

body

Anatomy of a method

```
public void run() {
    double mid = average(5.0, 10.2);
    println(mid);
}


private double average(double a, double b) {
    double sum = a + b;
    return sum / 2;
}
```

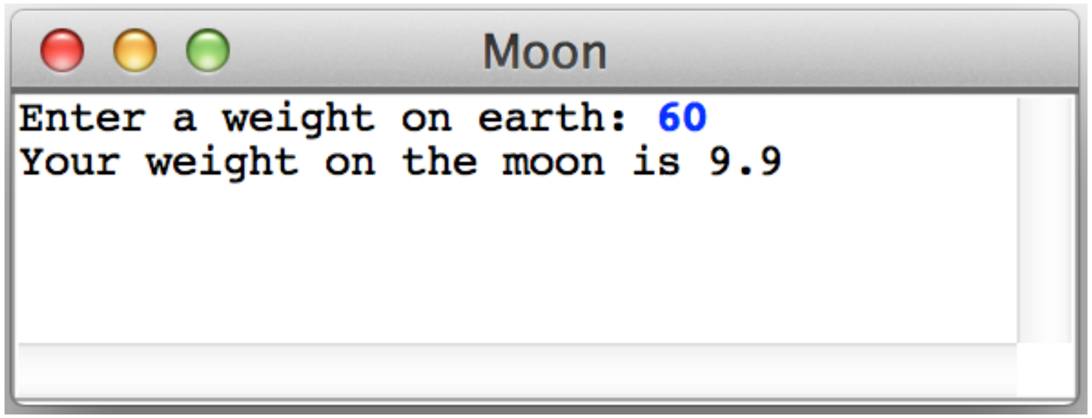
Also possible to return a value

Anatomy of a method

method "call"

```
public void run() {
    double mid = average(5.0, 10.2);
    println(mid);
}



private double average(double a, double b) {
    double sum = a + b;
    return sum / 2;
}
```

Method for Weight on Moon



```
Enter a weight on earth: 60
Your weight on the moon is 9.9
```

* Your weight on the moon is 16.5% your weight on the earth

As we left off..

Bad Times With Methods

```
// NOTE: This program is buggy!!

private void addFive(int x) {
   x += 5;
}


public void run() {
   int x = 3;
   addFive(x);
   println("x = " + x);
}
```

Good Times With Methods

```
// NOTE: This program is feeling just fine...

private int addFive(int x) {
   x += 5;
   return x;
}

public void run() {
   int x = 3;
   x = addFive(x);
   println("x = " + x);
}
```

Primitive *variables* are **not** passed! Their values are.

Pass by "Value"

More Examples

Changed Name

```
private void run() {
    int num = 5;
    cow(num);
}

private void cow(int grass) {
    println(grass);
}
```

Same Variable Name

```
private void run() {
    int money = 5;
    retireEarly();
    println(money);
}

private void retireEarly(){
    int money = 1200000;
    println(money);
}
```

No Methods in Methods

```
private void run() {
   println("hello world");
   private void sayGoodbye() {
      println("goodbye!");
   }
}
```

Illegal modifier for parameter goodbye, only final is permitted

Huh?!?

No Methods in Methods

```
private void run() {
    println(“hello world”);
    sayGoodbye();
}

private void sayGoodbye() {
    println(“goodbye!”);
}
```

Methods Called on Objects

```
GRect rect = new GRect(20, 20);
rect.setColor(Color.Blue);
```

receiver

* We will talk about how to define these later in the class

Remember Booleans?

Boolean Variable

```
boolean karelIsAwesome = true;

boolean myBool = 1 < 2;
```

Boolean Operations

```
boolean a = true;

boolean b = false;


boolean and = a && b;

boolean or = a || b;

boolean not = !a;
```

Is Divisible By

```java
private void run() {
    for(int i = 1; i <= 100; i++) {
        if(isDivisibleBy(i, 7)) {
            println(i);
        }
    }
}
```

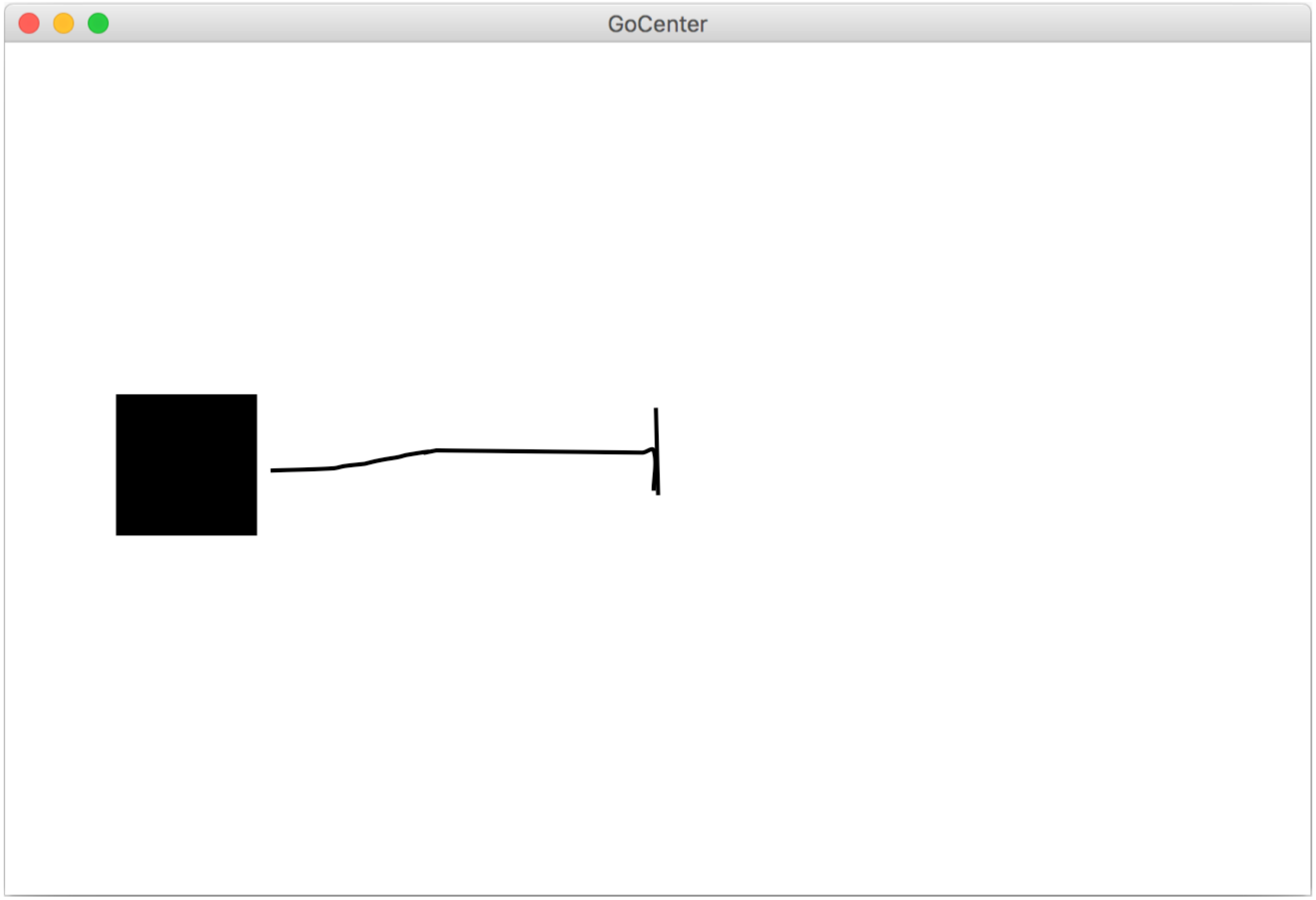
Boolean Return

```
private void run() {
    for(int i = 1; i <= 100; i++) {
        if(isDivisibleBy(i, 7)) {
            println(i);
        }
    }
}

private void isDivisibleBy(int a, int b) {
    if((a % b) == 0) {
        return true;
    } else {
        return false;
    }
}
```

Boolean Return

```
private void run() {
    for(int i = 1; i <= 100; i++) {
        if(isDivisibleBy(i, 7)) {
            println(i);
        }
    }
}

private void isDivisibleBy(int a, int b) {
    return a % b == 0;
}
```

Move to Center



Animation Loop

```
private void run() {
    // setup

    while(true) {
        // update world

        // pause
        pause(DELAY);
    }
}
```

Animation Loop

```
private void run() {
    // setup

    while(true) {
        // update world

        // pause
        pause(DELAY);
    }
}
```

Make all the variables you need. Add graphics to the screen.

Animation Loop

```
private void run() {
    // setup

    while(true) {
        // update world

        // pause
        pause(DELAY);
    }
}
```

The animation loop is a repetition of heartbeats

Animation Loop

```
private void run() {
    // setup

    while(true) {
        // update world

        // pause
        pause(DELAY);
    }
}
```

Each heart-beat, update the world forward one frame

Animation Loop

```
private void run() {
    // setup

    while(true) {
        // update world

        // pause
        pause(DELAY);
    }
}
```

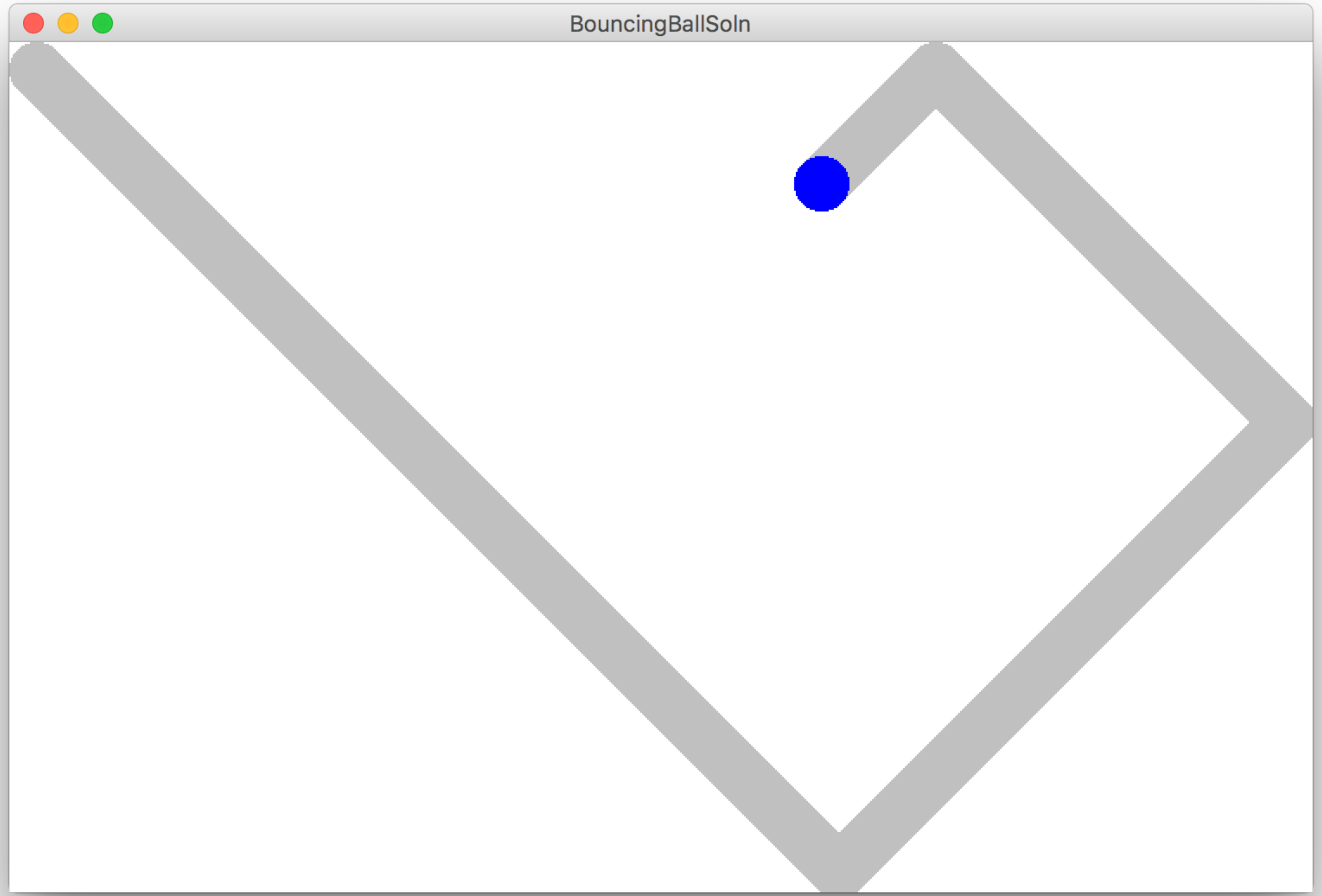
If you don't pause, humans won't be able to see it

Move To Center

```
private void run() {
    // setup
    GRect r = makeRect();
    while(!isPastCenter(r))
        // update world
        r.move(1, 0);
        // pause
        pause(DELAY);
    }
}
```

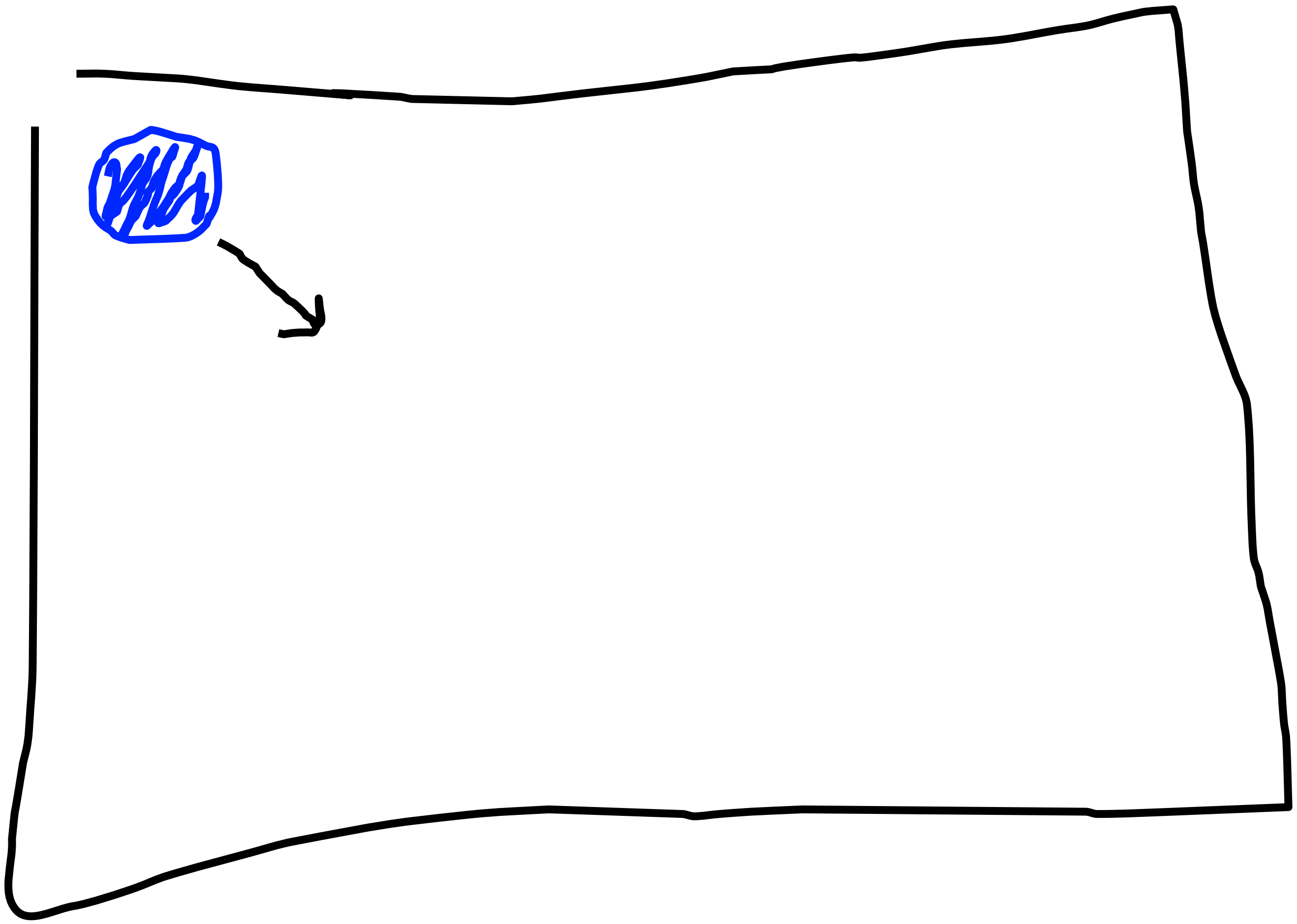
Animation in the CS Department

Bouncing Ball



Bouncing Ball

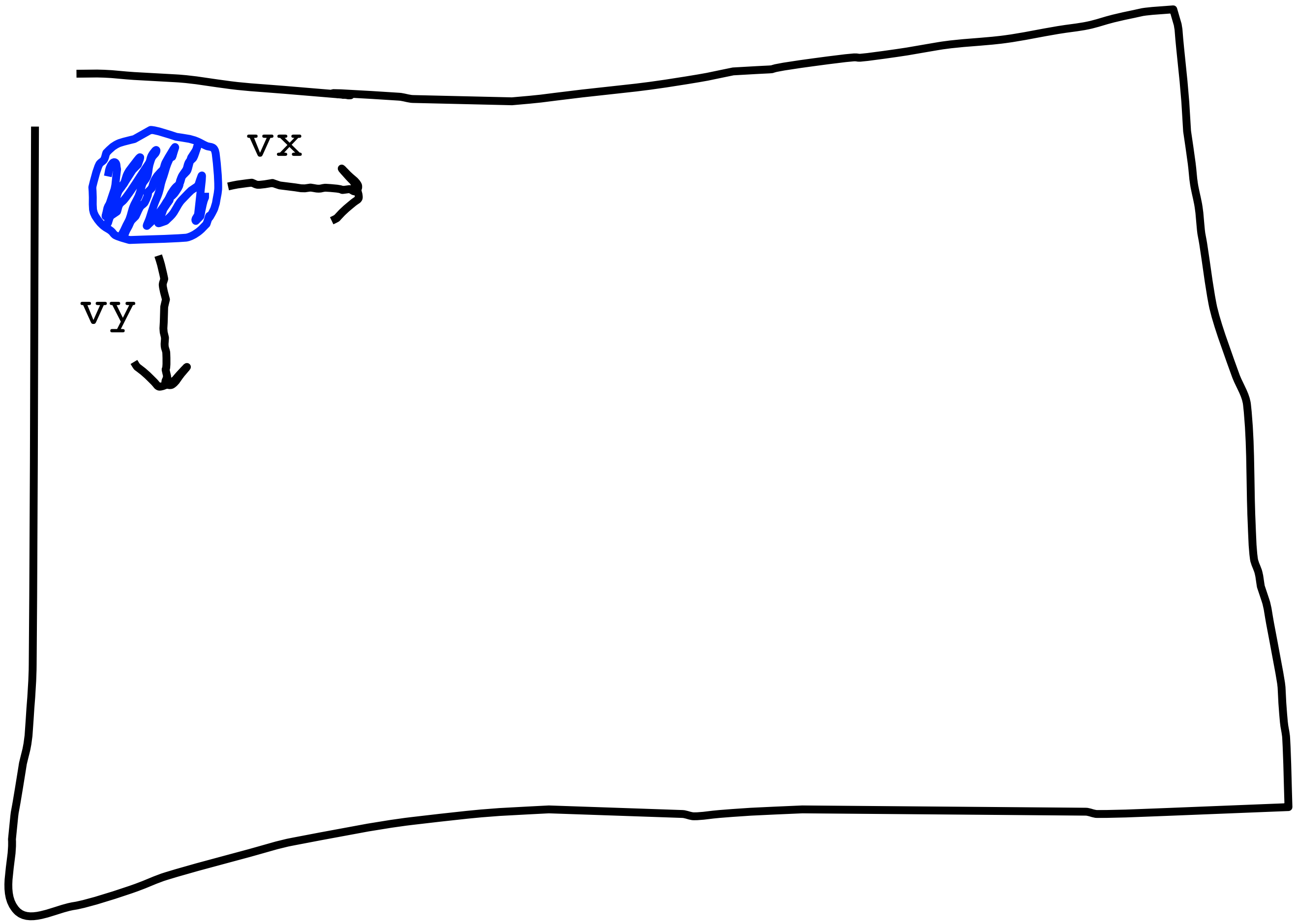
First heartbeat



Velocity: how much the ball position changes each heartbeat

Bouncing Ball

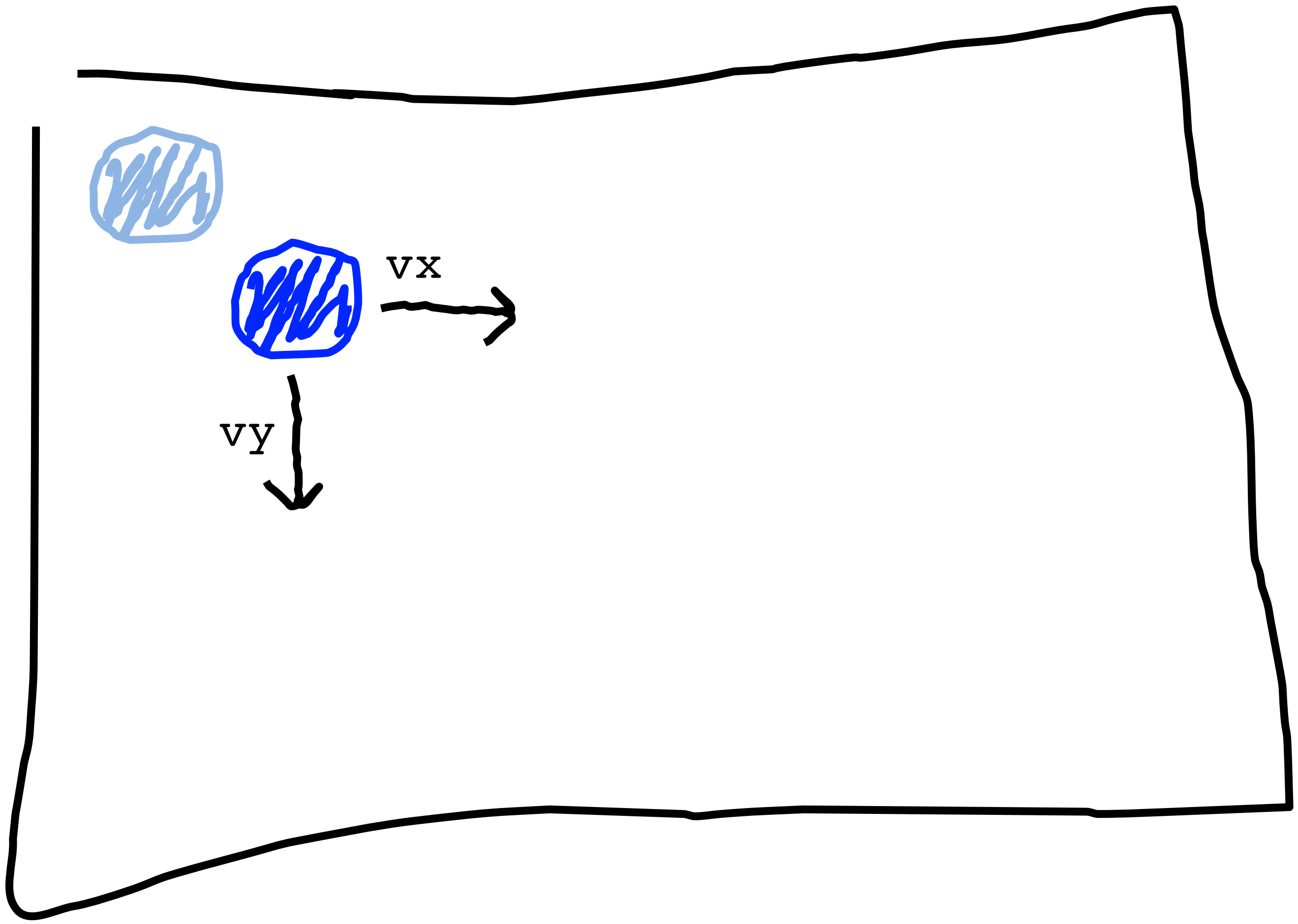
First heartbeat



The `GOval` **`move`** method takes in a change in x and a change in y

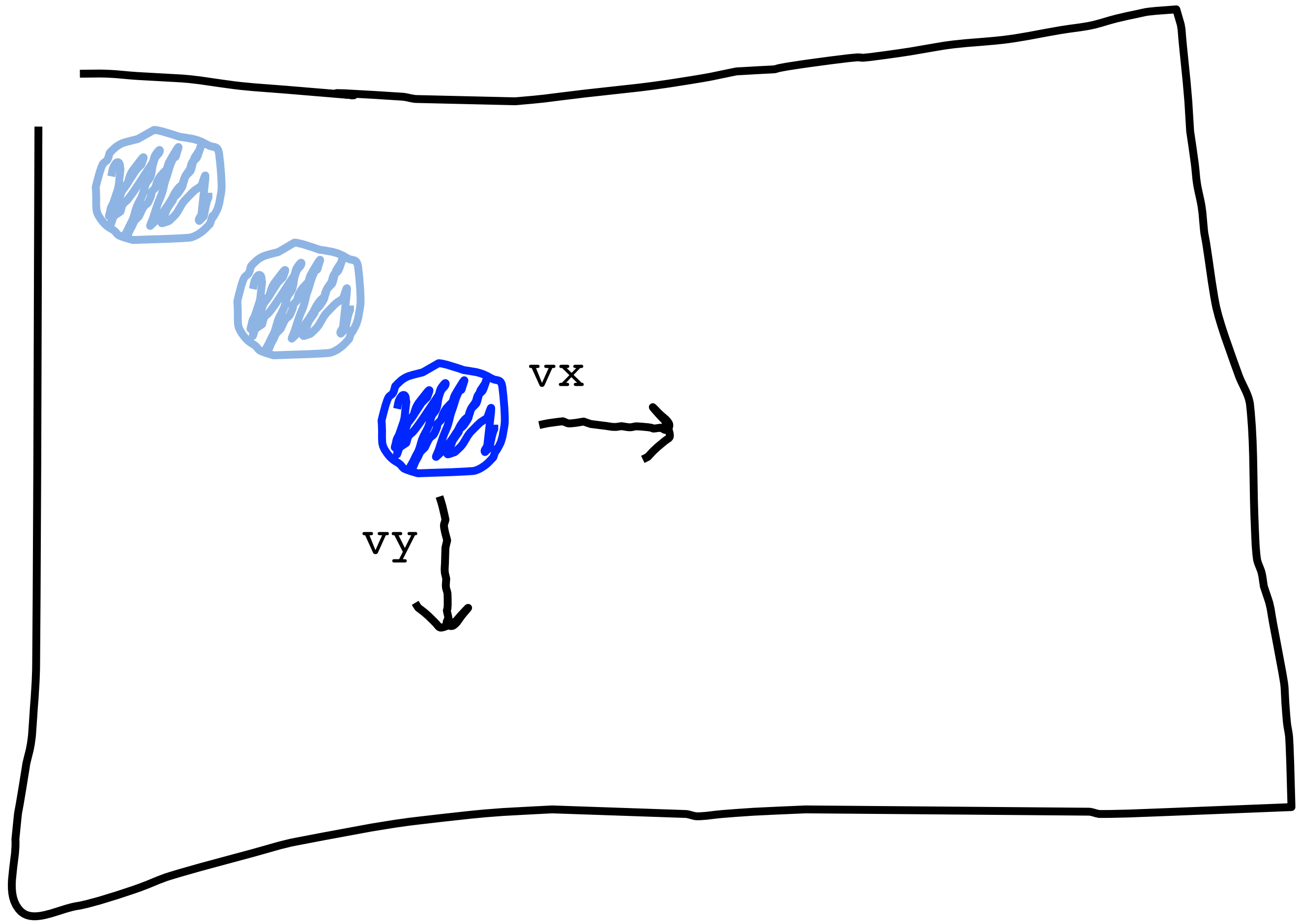
Bouncing Ball

Second heartbeat



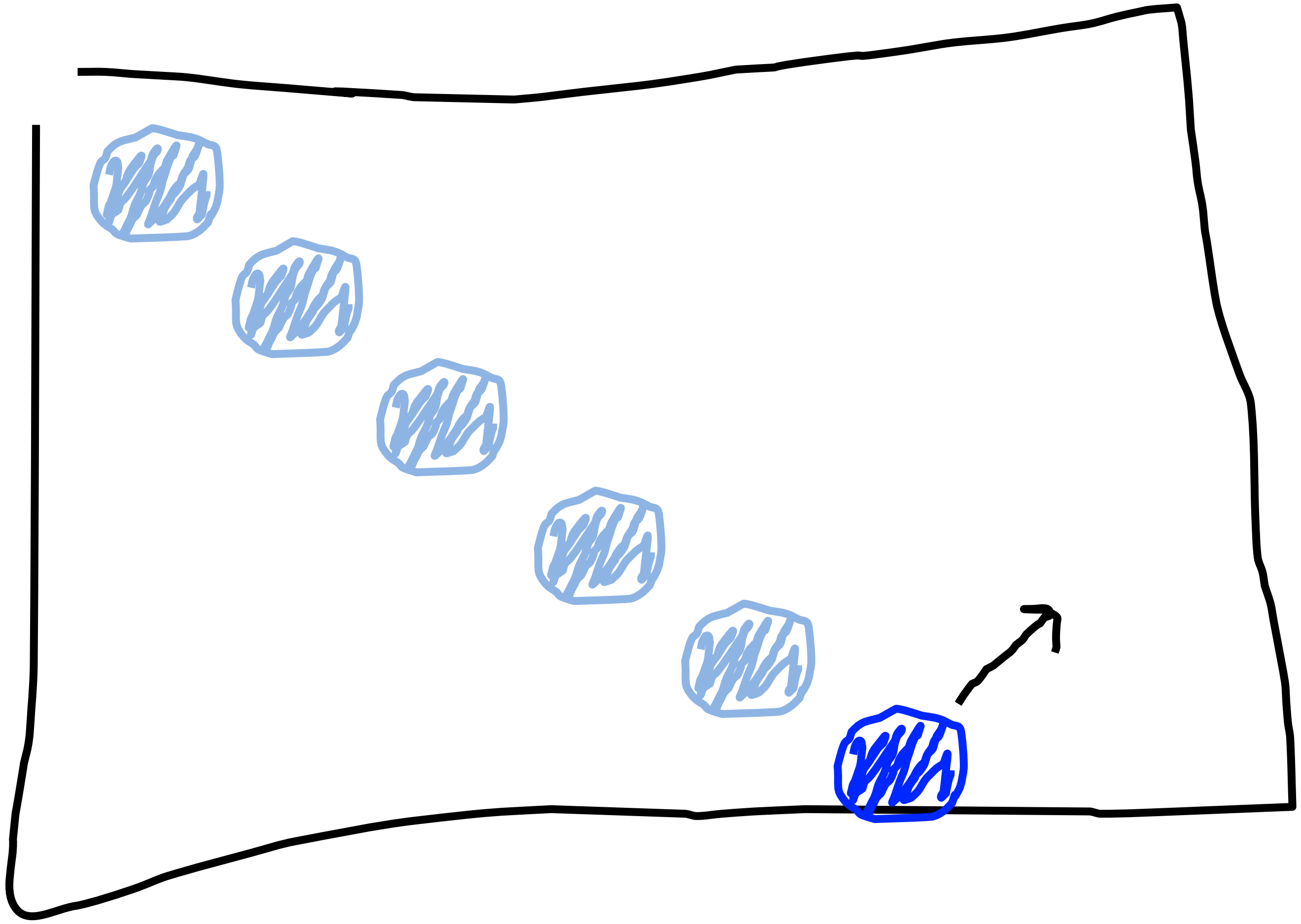
Bouncing Ball

Third heartbeat



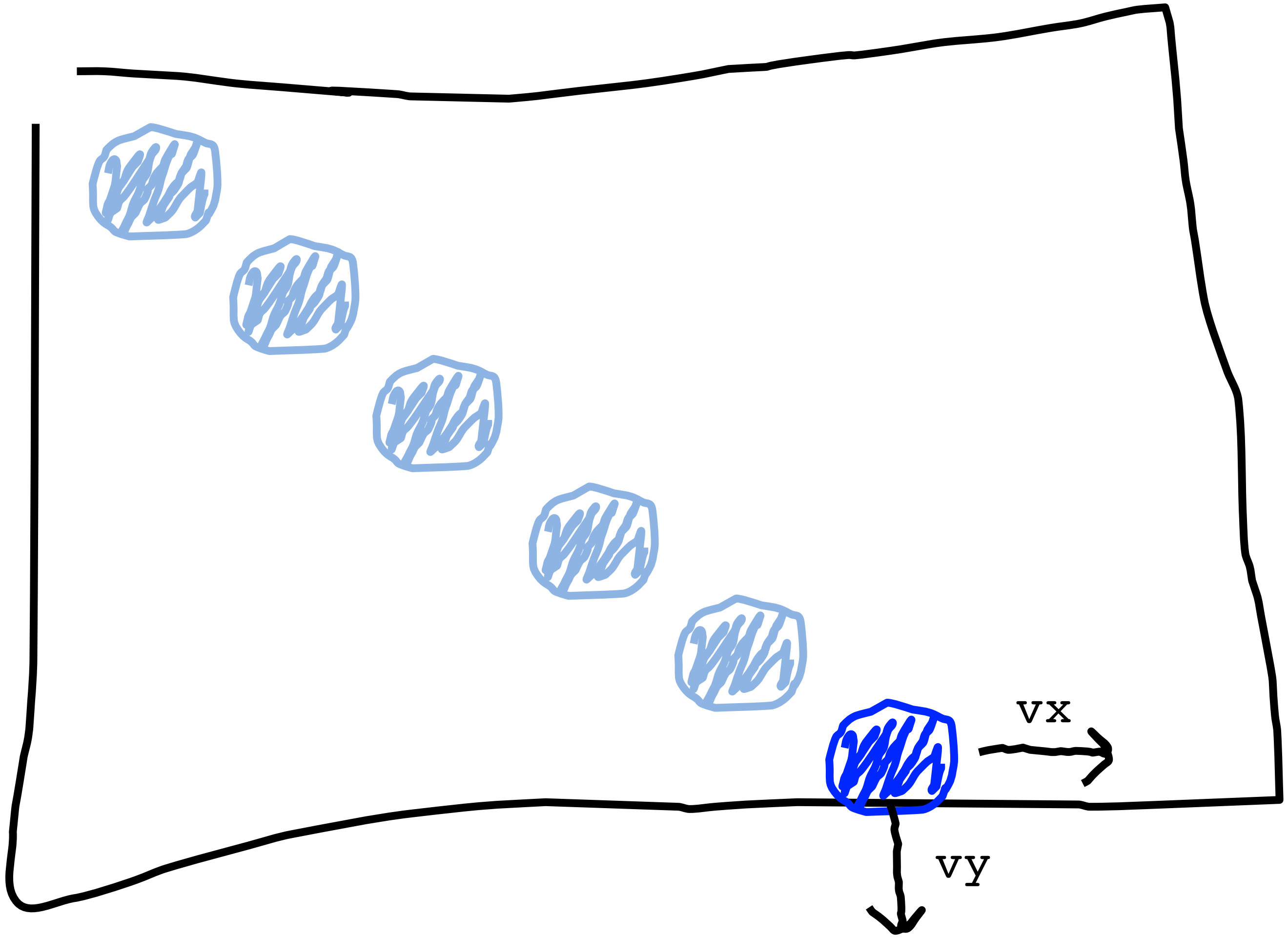
Bouncing Ball

What happens when we hit a wall?



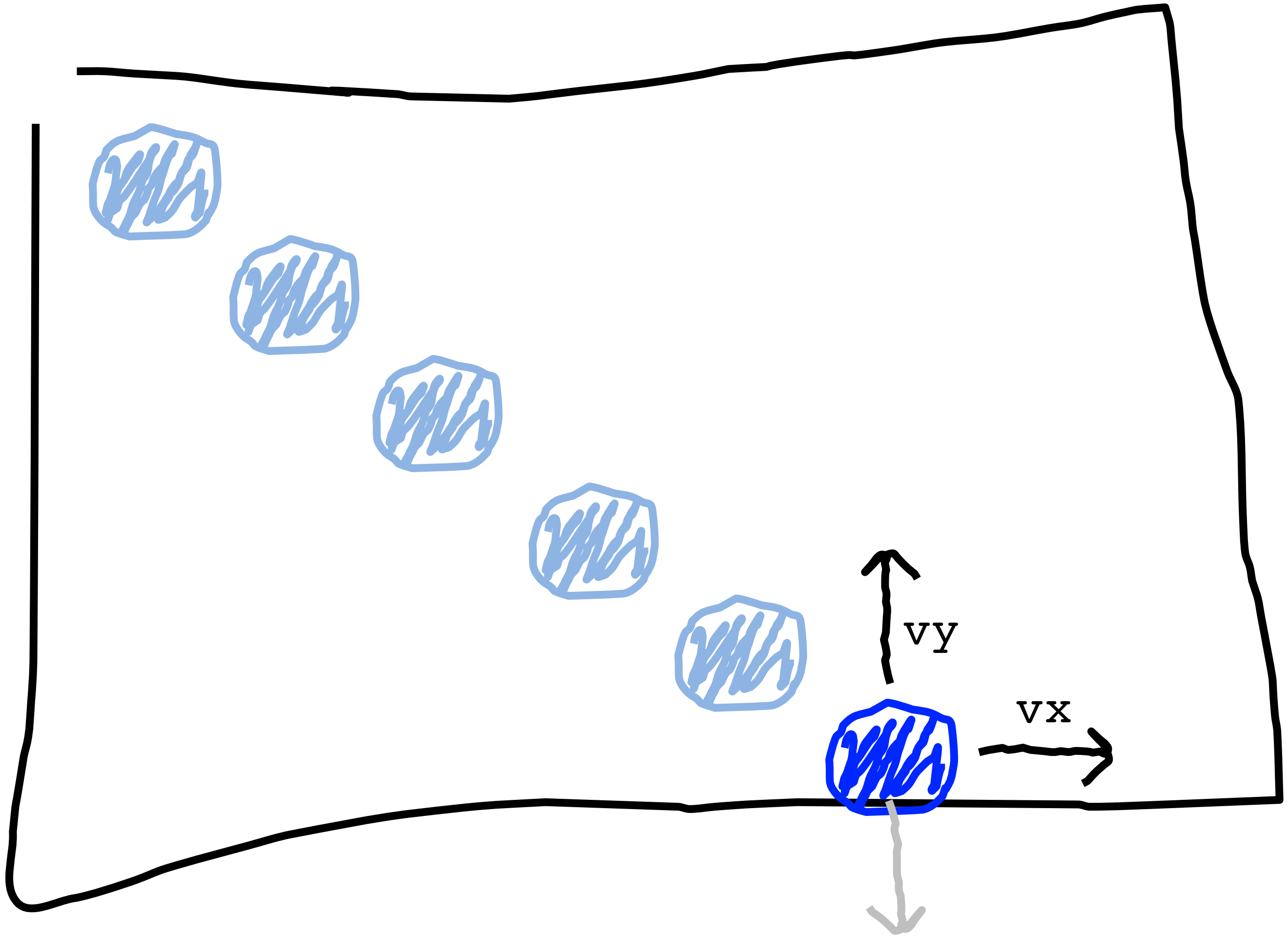
Bouncing Ball

We have this velocity



Bouncing Ball

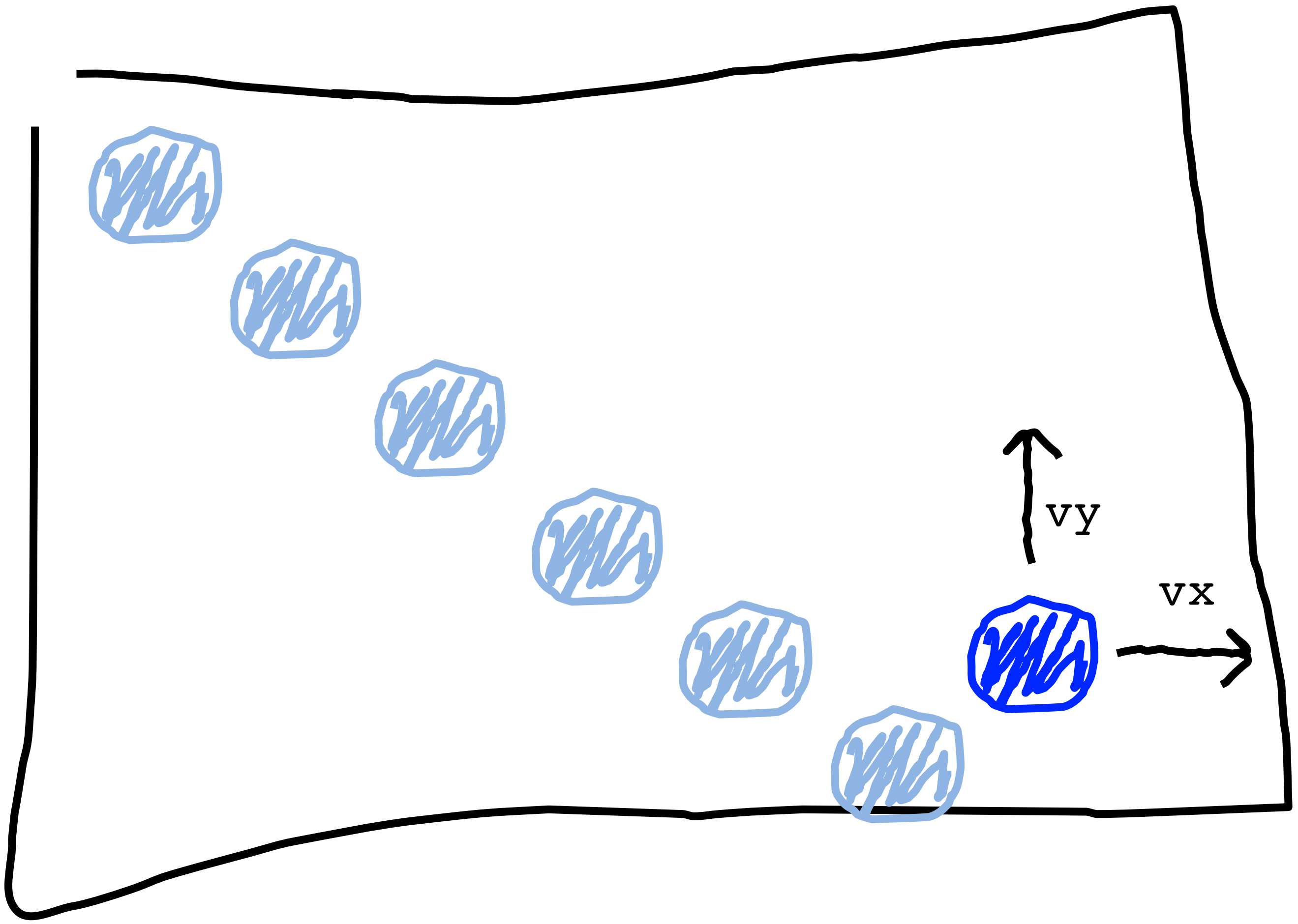
Our new velocity



When reflecting vertically: `vy = -vy`

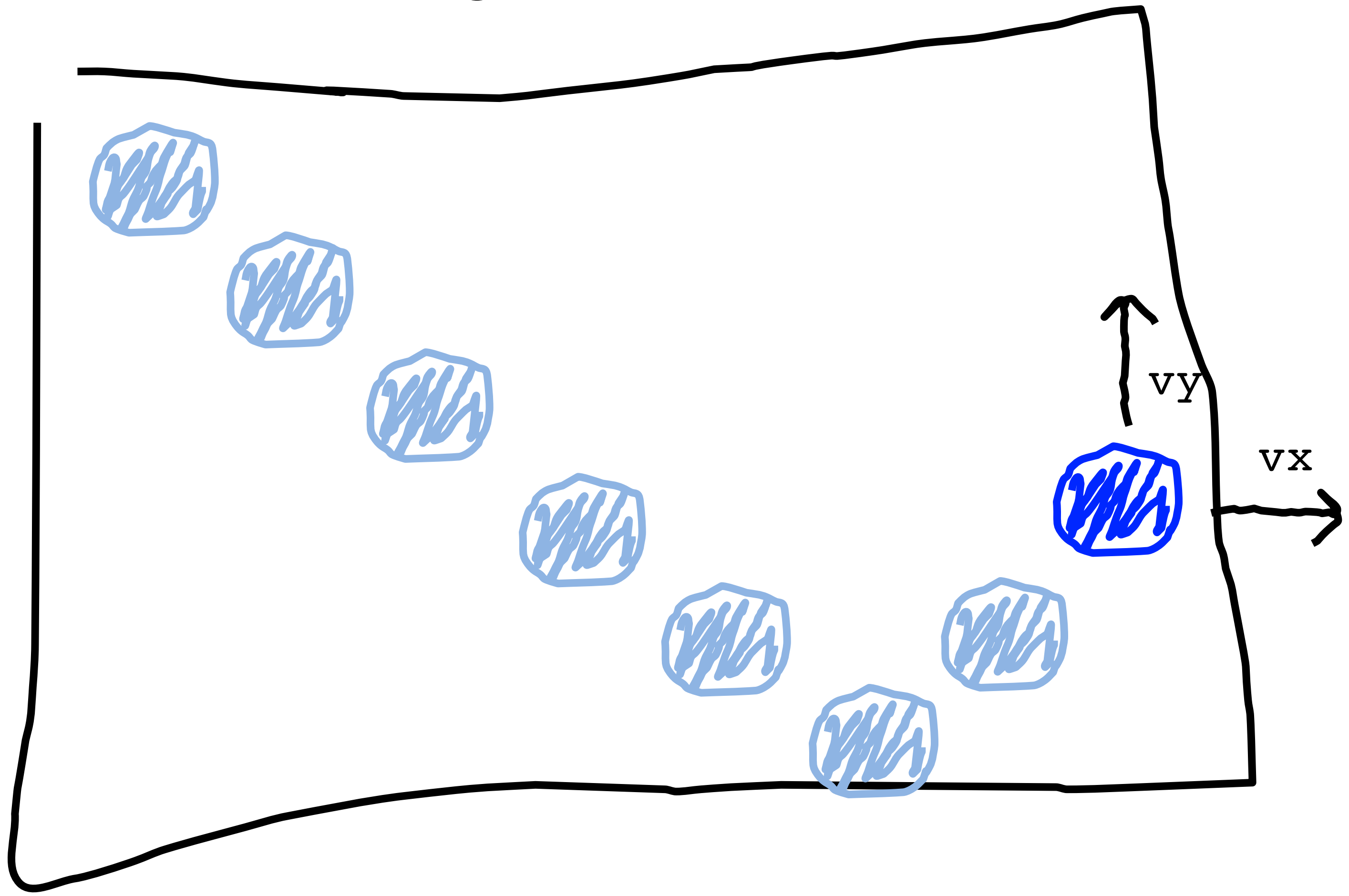
Bouncing Ball

Seventh heartbeat



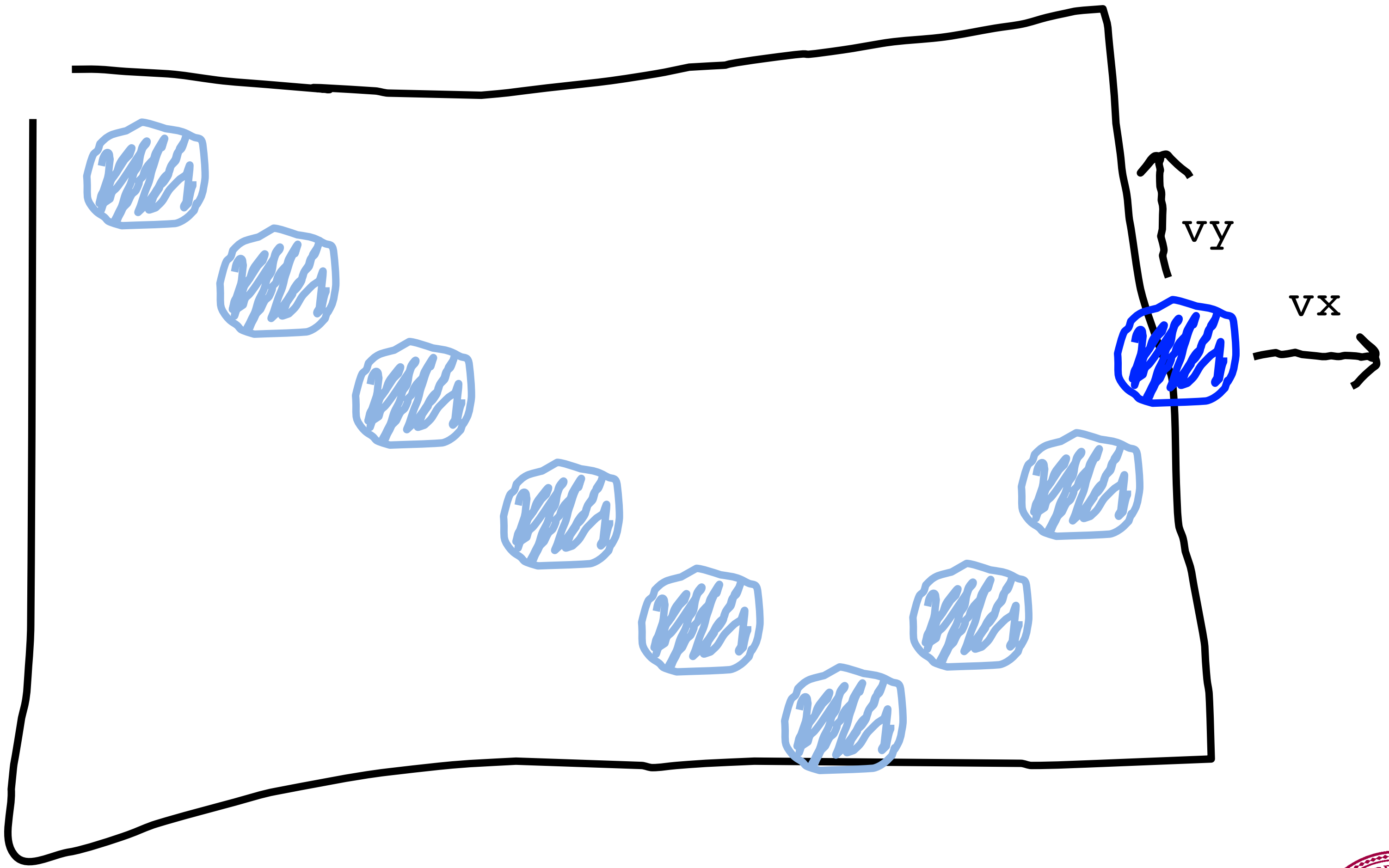
Bouncing Ball

Eighth heartbeat



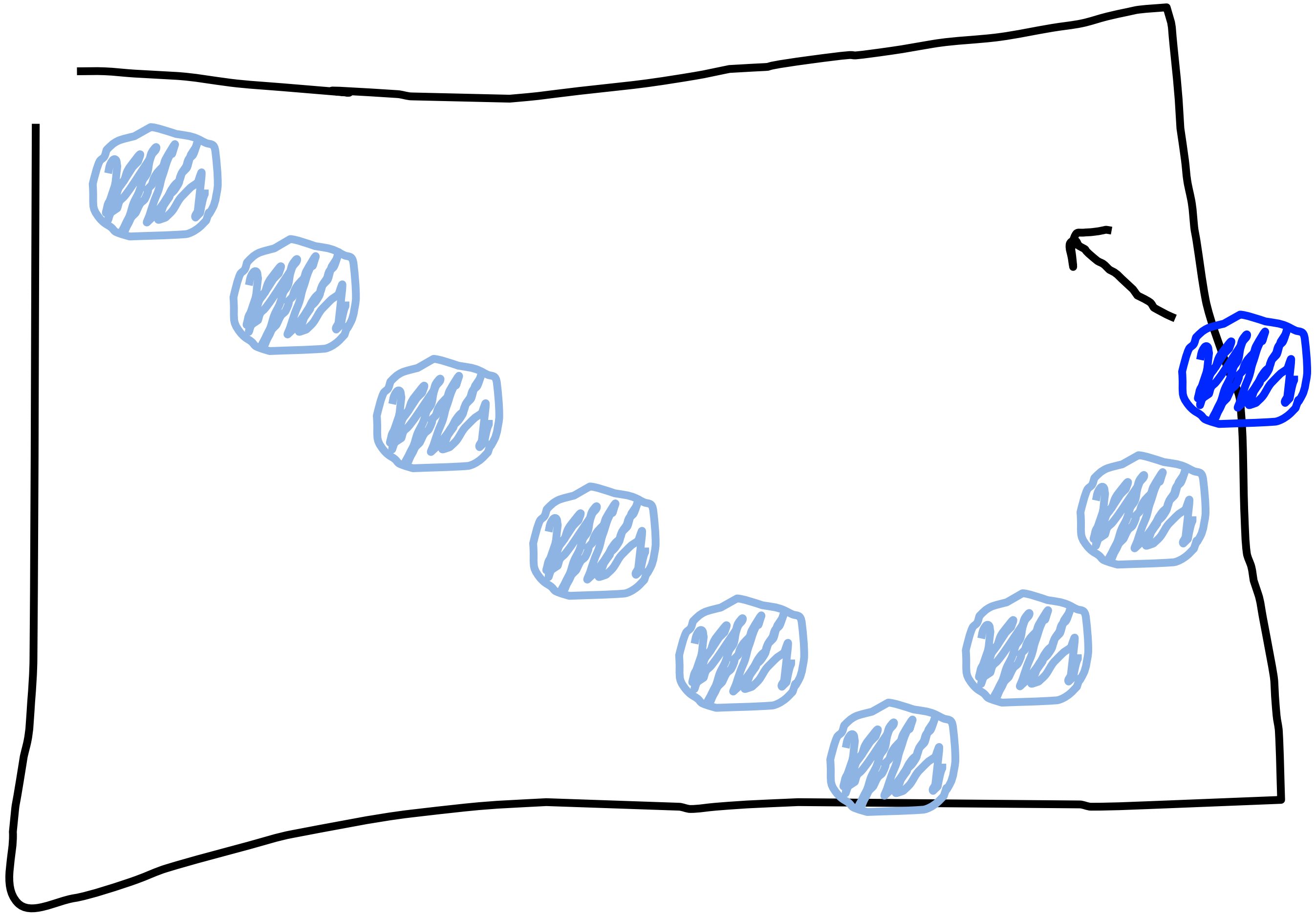
Bouncing Ball

Ninth heartbeat



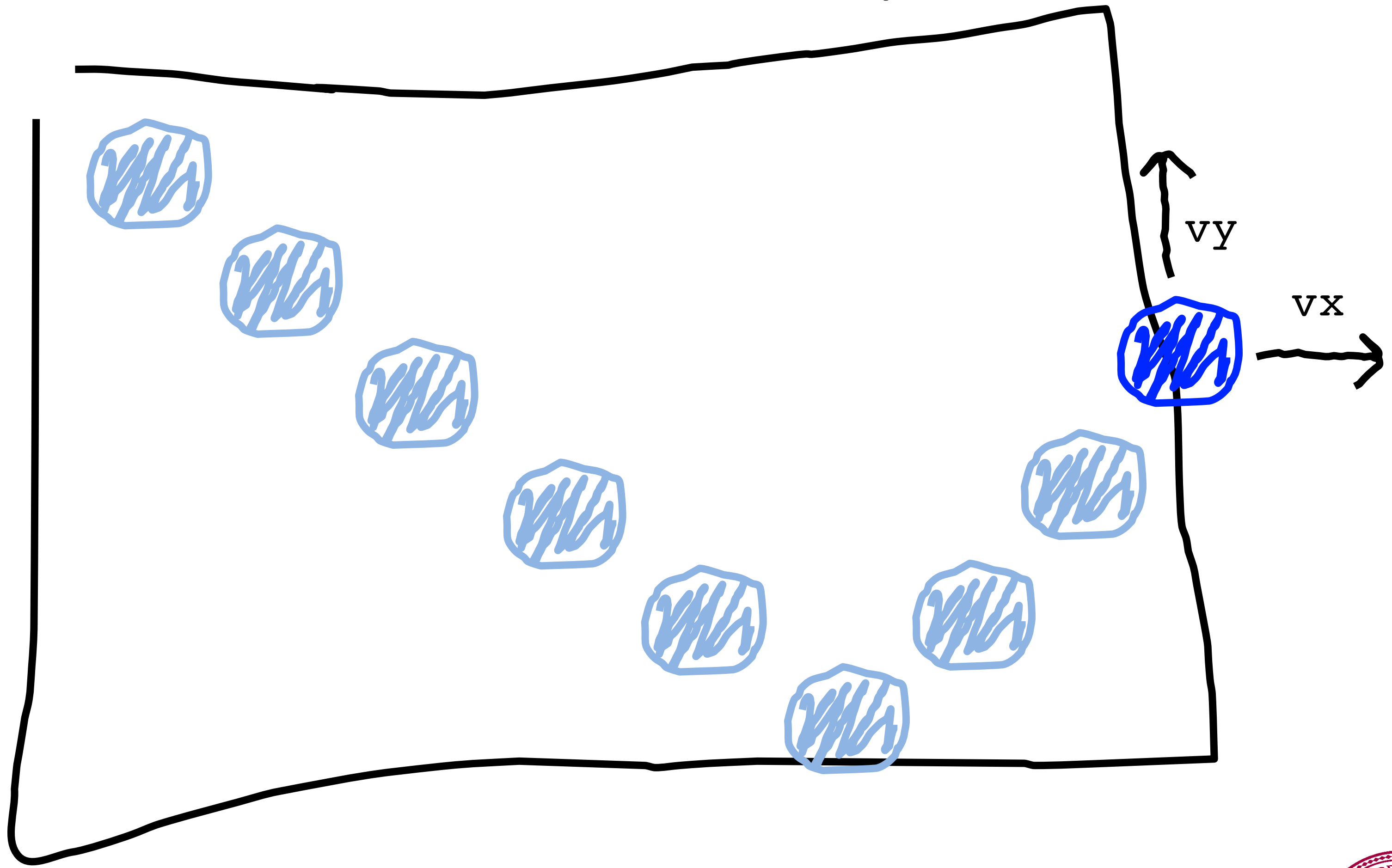
Bouncing Ball

We want this!



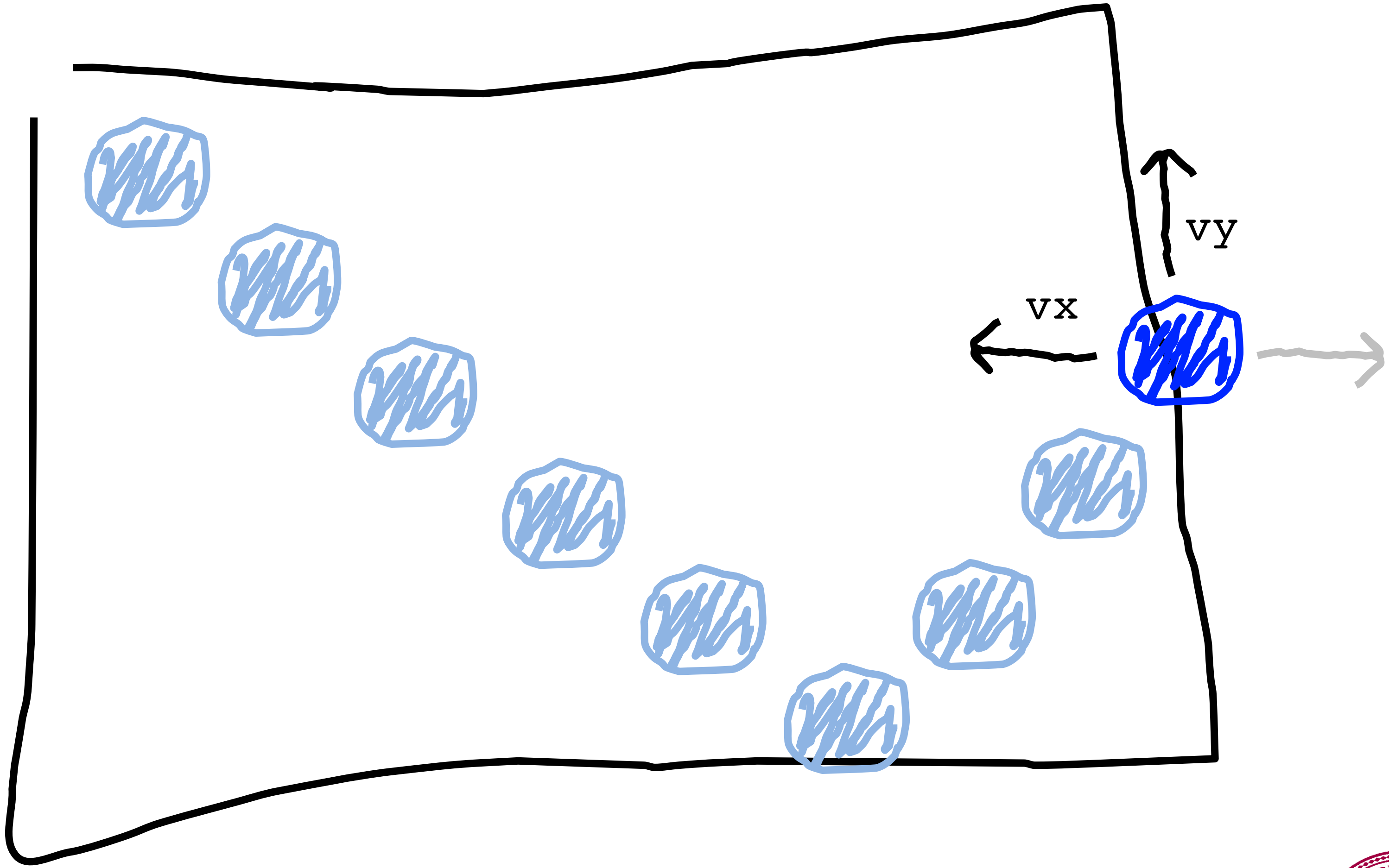
Bouncing Ball

This was our old velocity



Bouncing Ball

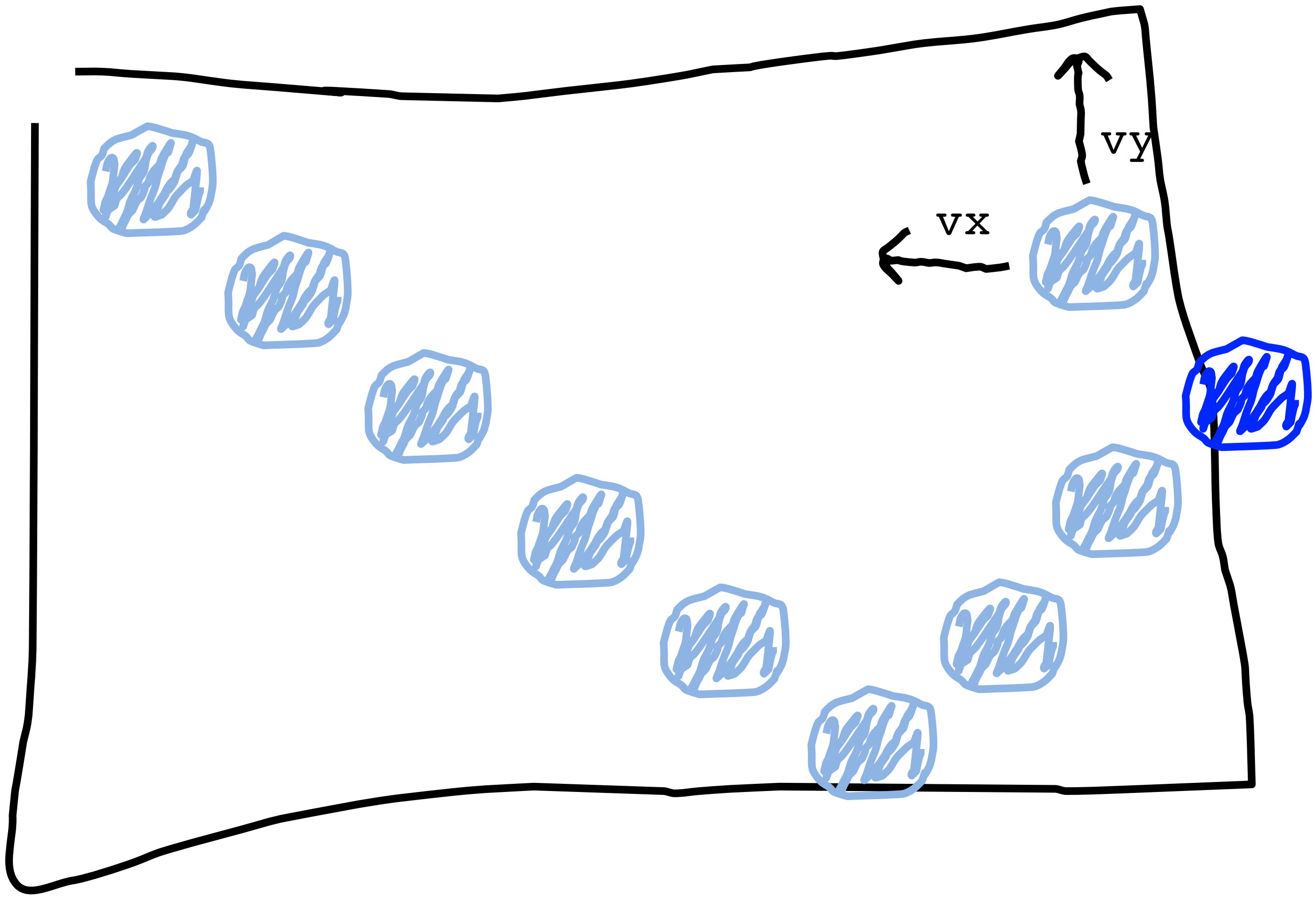
This is our new velocity



When reflecting horizontally: `vx = -vx`

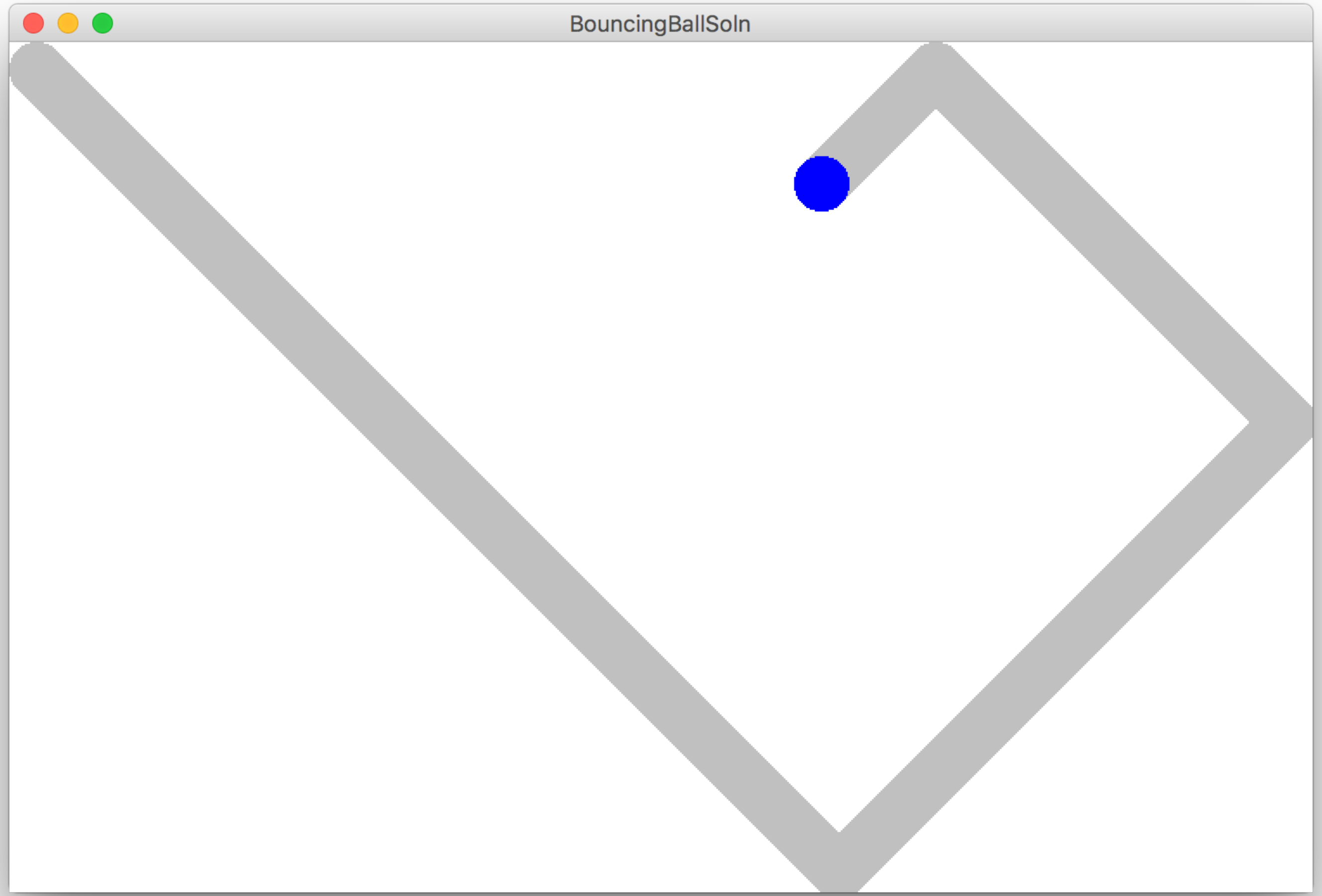
Bouncing Ball

Tenth heartbeat

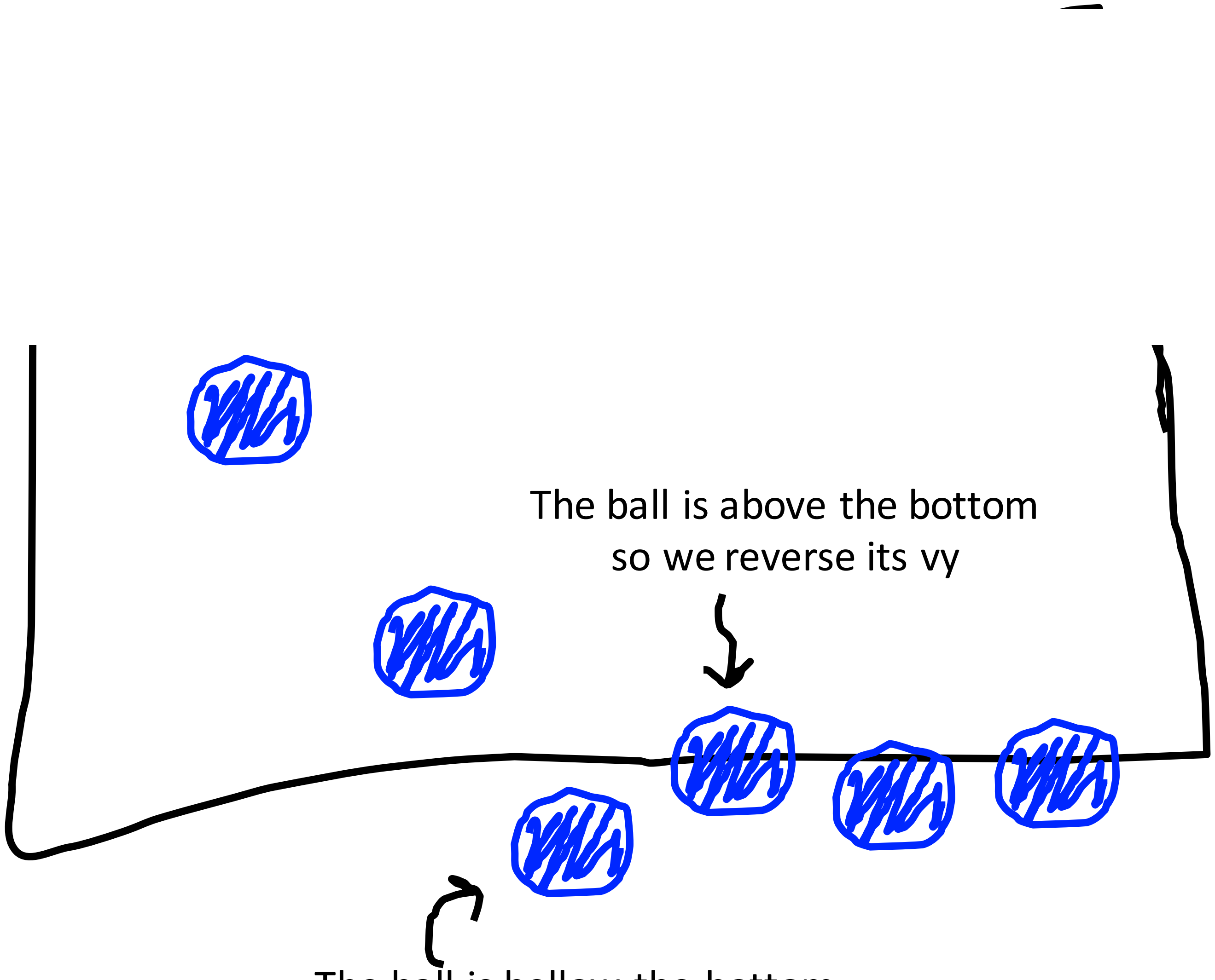


When reflecting horizontally: `vx = -vx`

Bouncing Ball



A Sticky Situation



The ball is above the bottom
so we reverse its vy

The ball is bellow the bottom
so we reverse its vy

Learning Goals

1. Feel more confident writing methods
2. Write animated programs