Animation

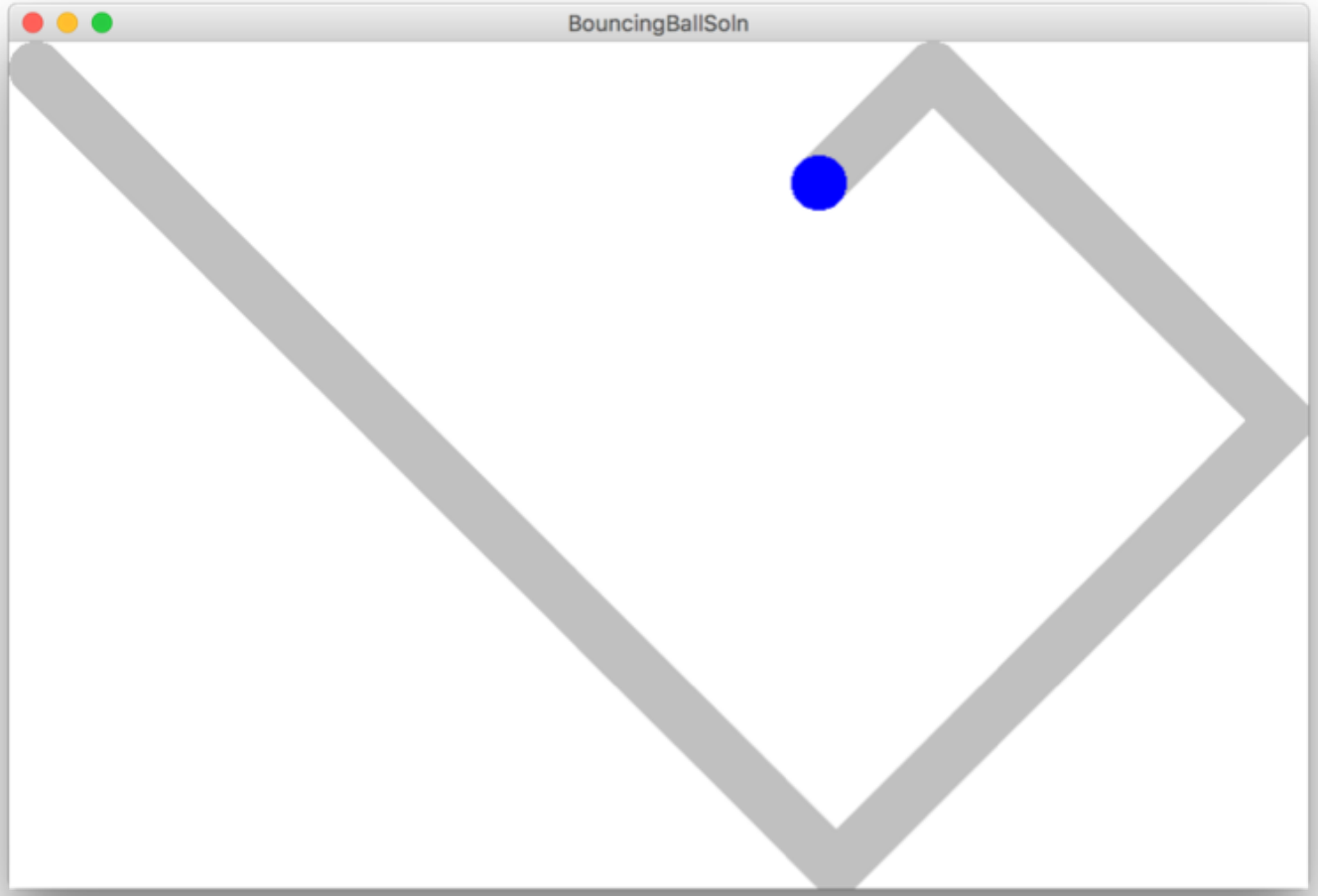
Chris Piech
CS106A, Stanford University

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

Learning Goals

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

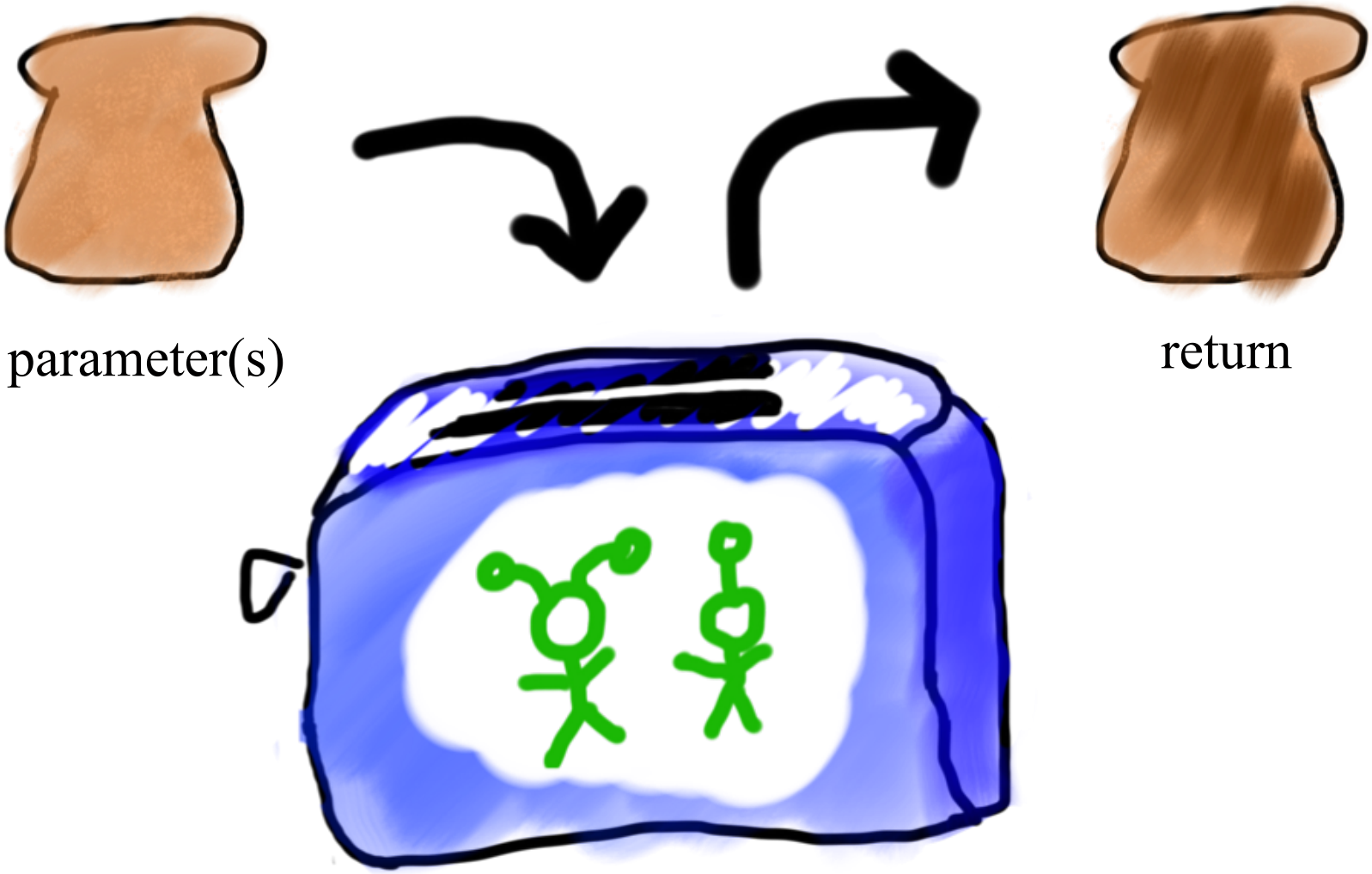
You will be able to write Bouncing Ball



Defining a Method

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

Methods are Like Toasters



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

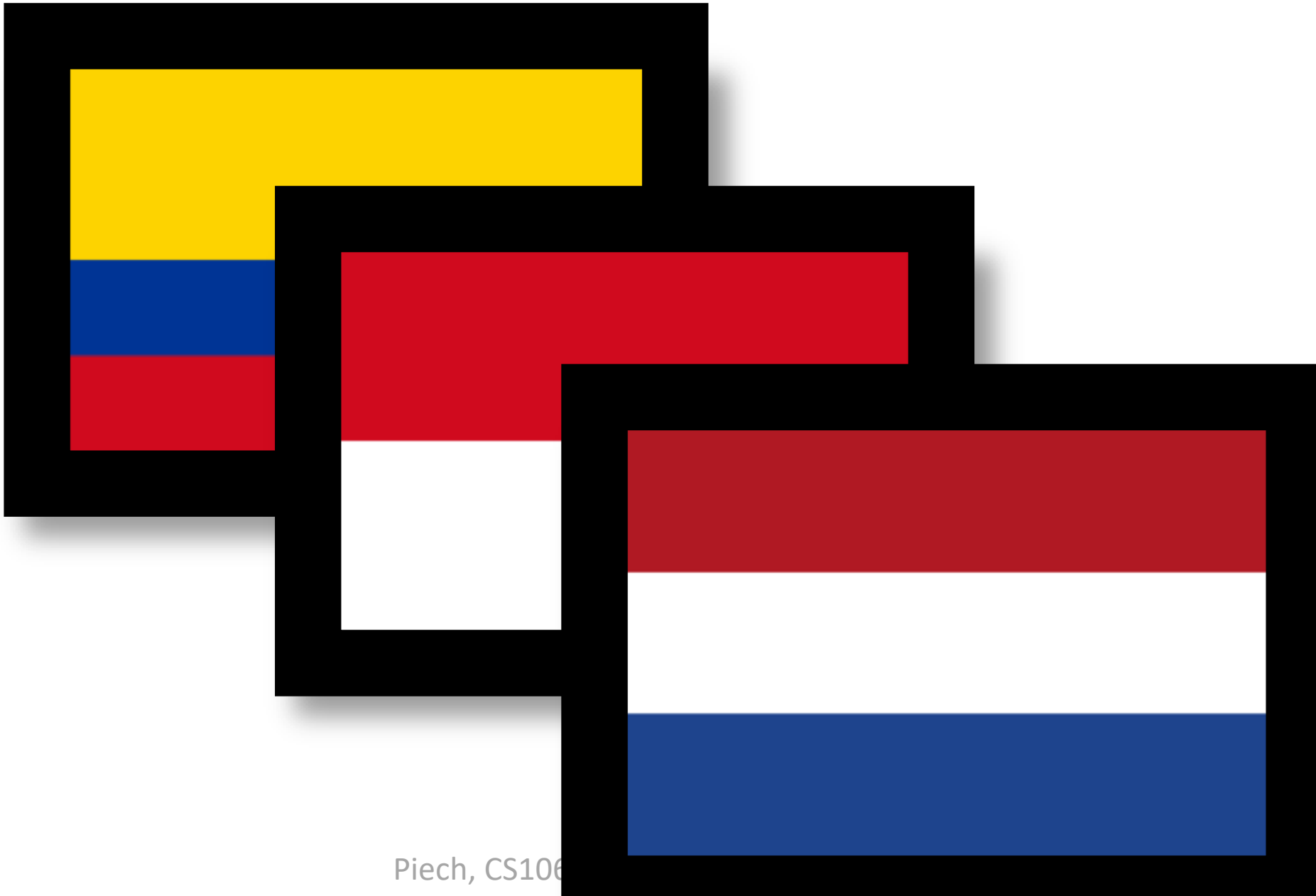
```
private double average(double a, double b) {
    double sum = a + b;
    return sum / 2;
}
```


When passing inputs to methods:

Copies are passed.
Via new variables.

The method definition names the new variables.

Drawing Flags



Flags

```
private void drawColombiasFlag() {
    // Call drawStripe three times
    // with different inputs each time
    drawStripe(Color.YELLOW, 0);
    drawStripe(Color.BLUE, 0.5);
    drawStripe(Color.RED, 0.75);
}
```

```
// Define drawStripe
// drawStripe needs two inputs (which come as variables)
// First a color, then a fraction down the screen
private void drawStripe(Color color, double fractionDown) {
    double y = getHeight() * fractionDown;
    GRect rect = new GRect(0, y, getWidth(), getHeight() - y);
    rect.setColor(color);
    rect.setFilled(true);
    add(rect);
}
```

method “definition”

Flags

```
private void drawColombiasFlag() {
    // Call drawStripe three times
    // with different inputs each time
    drawStripe(Color.YELLOW, 0);
    drawStripe(Color.BLUE, 0.5);
    drawStripe(Color.RED, 0.75);
}
```

```
// Define drawStripe
// drawStripe needs tw
// First a color, then a fraction down the screen
private void drawStripe(Color color, double fractionDown) {
    double y = getHeight() * fractionDown;
    GRect rect = new GRect(0, y, getWidth(), getHeight() - y);
    rect.setColor(color);
    rect.setFilled(true);
    add(rect);
}
```

Takes two inputs: (1) a color, (2) a fraction

Flags

```
private void drawColombiasFlag() {
    // Call drawStripe three times
    // with different inputs each
    drawStripe(Color.YELLOW, 0);
    drawStripe(Color.BLUE, 0.5);
    drawStripe(Color.RED, 0.75);
}


// Define drawStripe
// drawStripe needs two inputs (which come as variables)
// First a color, then a fraction down the screen
private void drawStripe(Color color, double fractionDown) {
    double y = getHeight() * fractionDown;
    GRect rect = new GRect(0, y, getWidth(), getHeight() - y);
    rect.setColor(color);
    rect.setFilled(true);
    add(rect);
}
```

Thus, every time the method is called, two inputs must be given!

Flags

```
private void drawColombiasFlag() {
    // Call drawStripe three times
    // with different inputs each time
    drawStripe(Color.YELLOW, 0);
    drawStripe(Color.BLUE, 0.5);
    drawStripe(Color.RED, 0.75);
}
```

The method receives the first input in a box (aka as a variable), with the name color

```
private void drawStripe(Color color, double fractionDown) {
    double y = getHeight() * fractionDown;
    GRect rect = new GRect(0, y, getWidth(), getHeight() - y);
    rect.setColor(color);
    rect.setFilled(true);
    add(rect);
}
```

It can use the value in the color variable

Flags

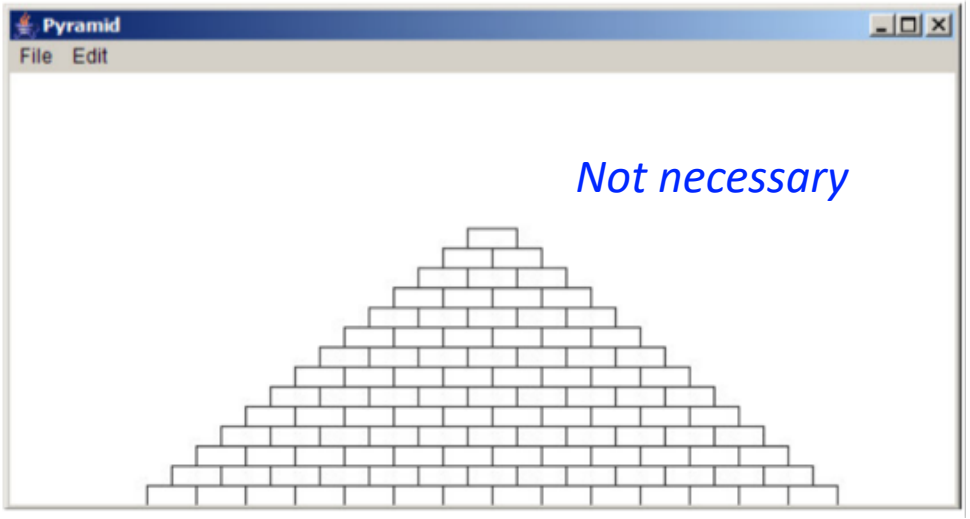
```
private void drawColombiasFlag() {
    // Call drawStripe three times
    // with different inputs each time
    drawStripe(Color.YELLOW, 0);
    drawStripe(Color.BLUE, 0.5);
    drawStripe(Color.RED, 0.75);
}
```

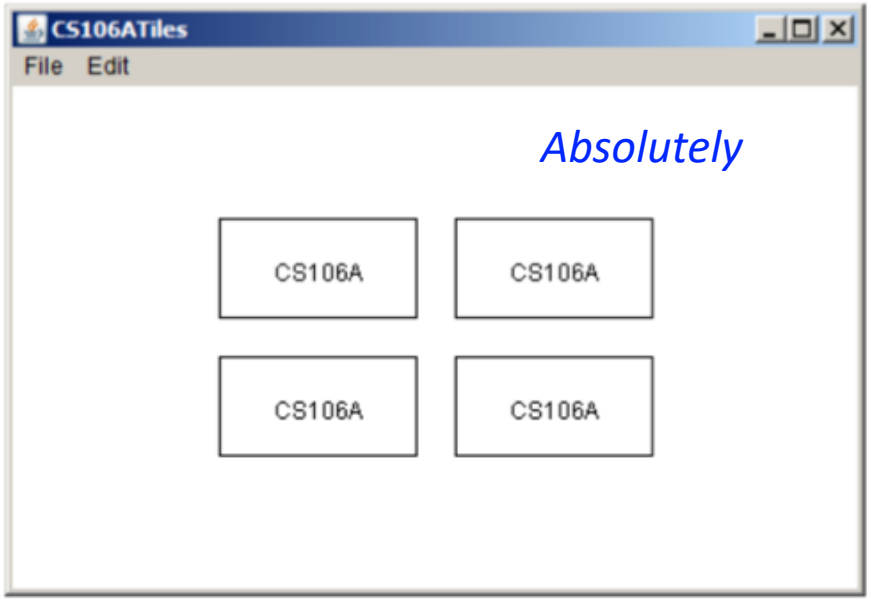
The method receives the second input in a box (aka as a variable), with the name fractionDown

```
// Define drawSt
// drawStripe ne
// First a color, then a fraction down the screen
private void drawStripe(Color color, double fractionDown) {
    double y = getHeight() * fractionDown;
    GRect rect = new GRe                    Height() - y);
    rect.setColor(color)
    rect.setFilled(true)
    add(rect);
}
```

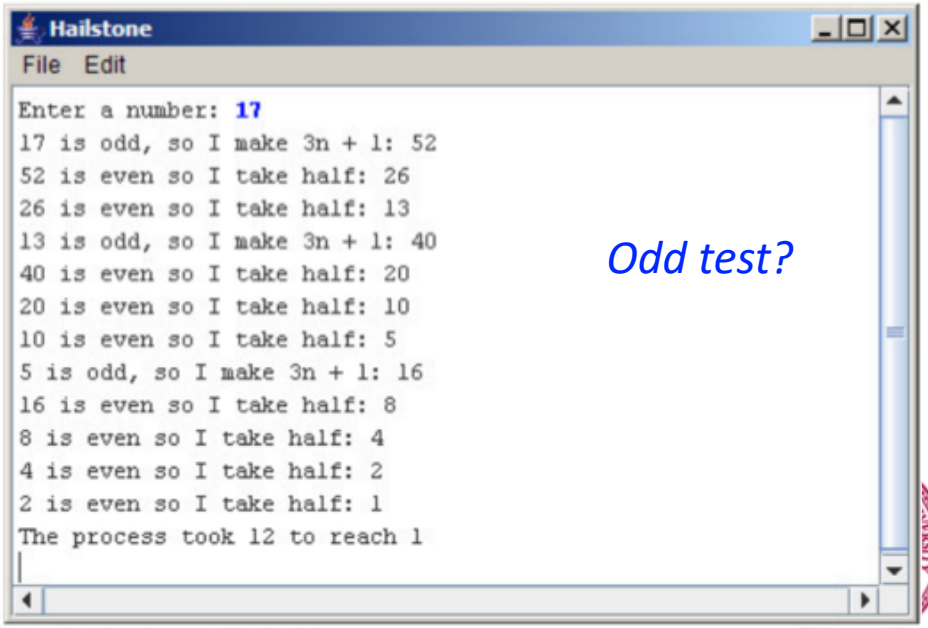
It can use the value in the fractionDown variable

Assn 2


Pyramid
File Edit
Not necessary


CS106ATiles
File Edit
Absolutely
CS106A
CS106A
CS106A
CS106A


Target
File Edit
Yes

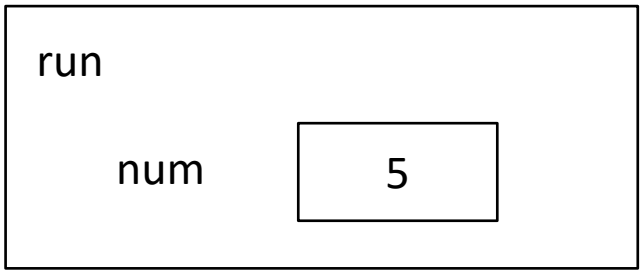

Hailstone
File Edit
Enter a number: 17
17 is odd, so I make 3n + 1: 52
52 is even so I take half: 26
26 is even so I take half: 13
13 is odd, so I make 3n + 1: 40
40 is even so I take half: 20
20 is even so I take half: 10
10 is even so I take half: 5
5 is odd, so I make 3n + 1: 16
16 is even so I take half: 8
8 is even so I take half: 4
4 is even so I take half: 2
2 is even so I take half: 1
The process took 12 to reach 1
Odd test?

As we left off..

Changed Name

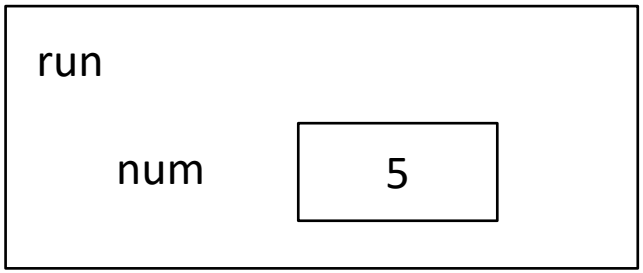
```
private void run() {
    int num = 5;
    cow(num);
}

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


Changed Name




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

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

Changed Name

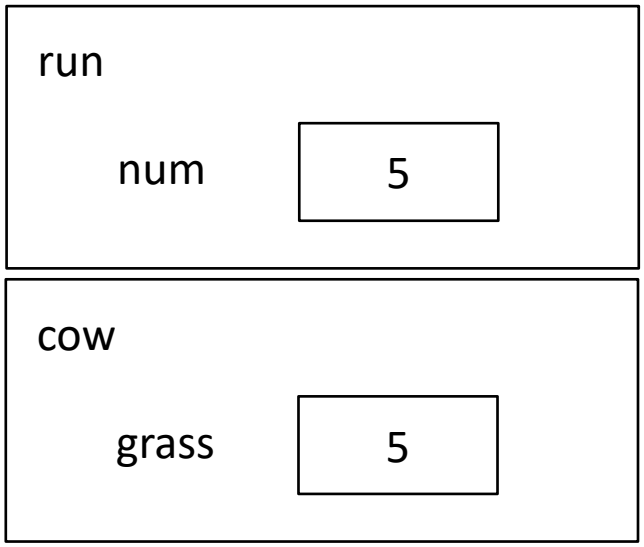



```
private void run() {
    int num = 5;
    cow(num);
}

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

Changed Name

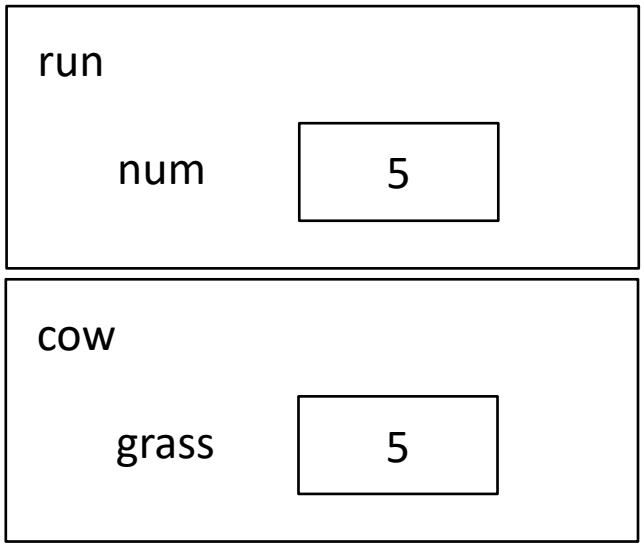
```
private void run() {
    int num = 5;
    cow(num);
}
```

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



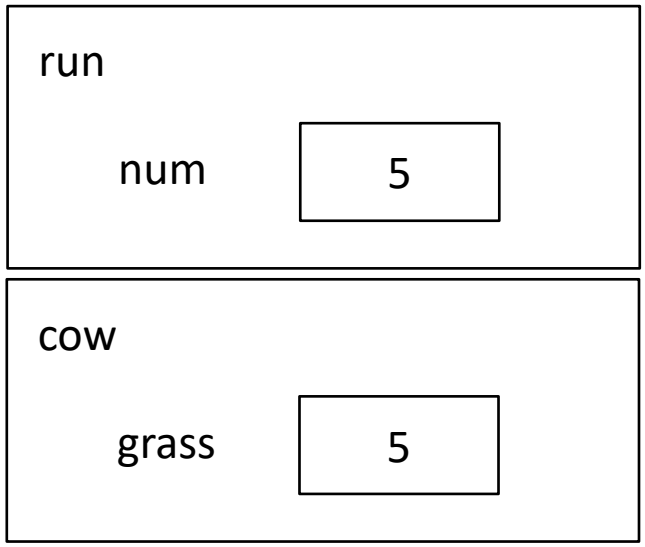
Changed Name

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

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



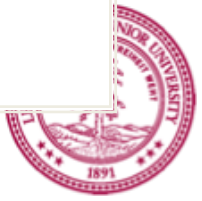
Changed Name

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

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



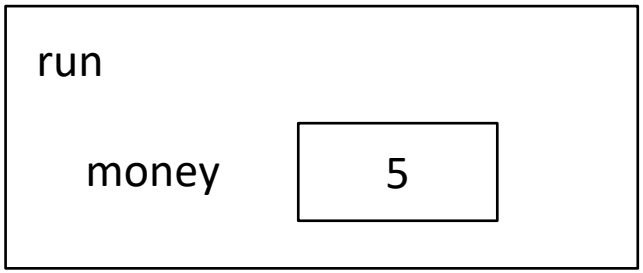
Same Variable Name

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

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



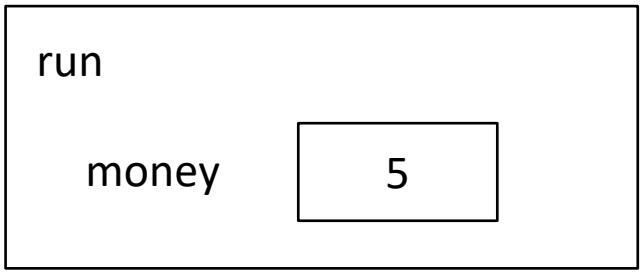
Same Variable Name

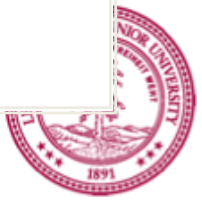
```
private void run() {
    int money = 5;
    retireEarly();
    println(money);
}

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



Same Variable Name

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

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

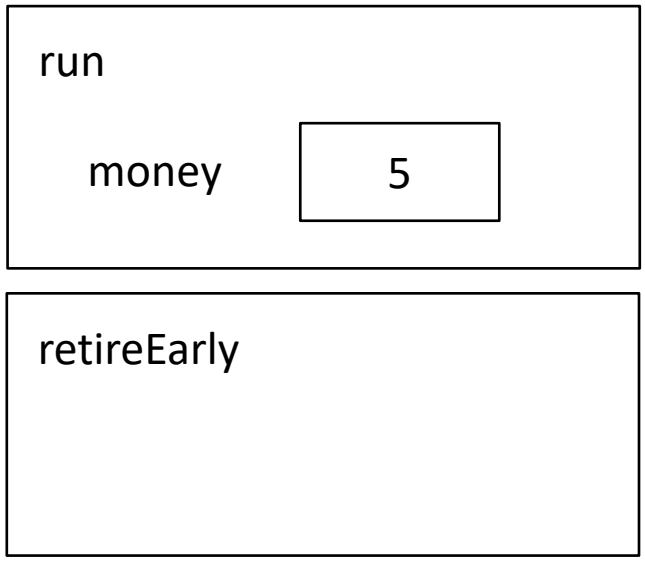




Same Variable Name

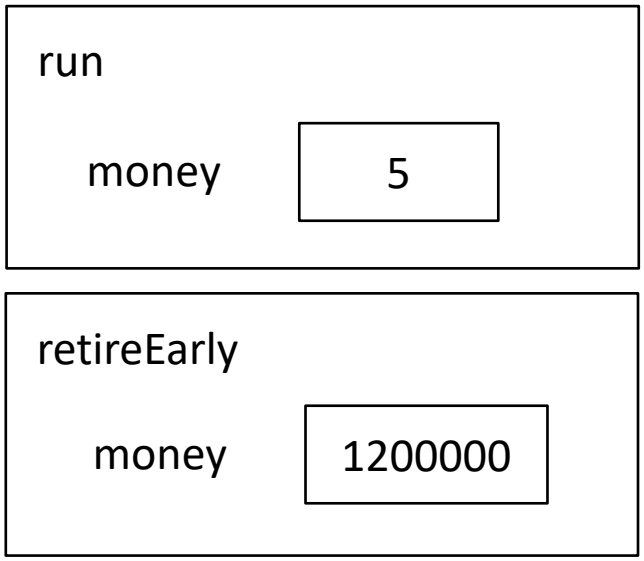
```
private void run() {
    int money = 5;
    retireEarly();
    println(money);
}
```

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



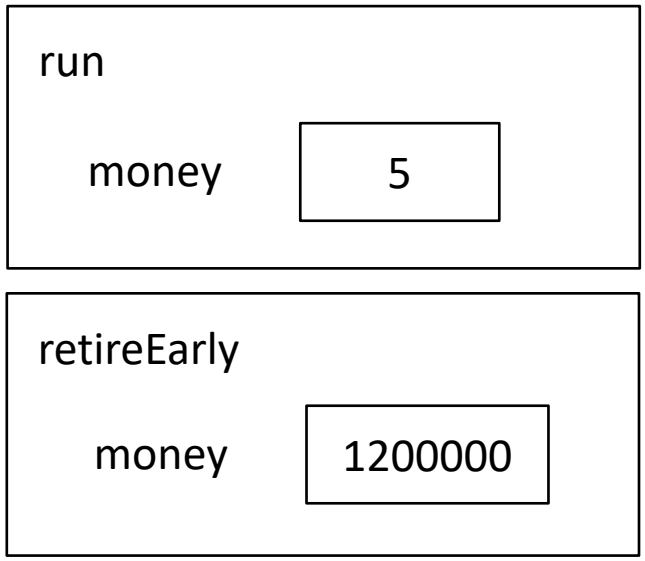
Same Variable Name

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

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



Same Variable Name

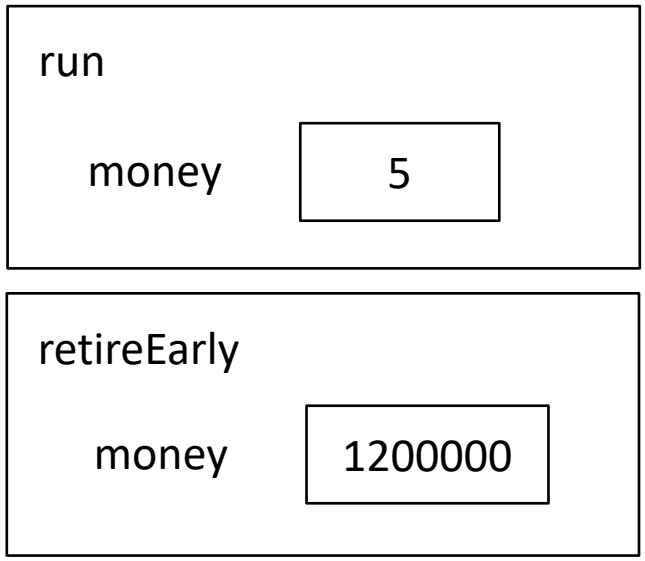
```
private void run() {
    int money = 5;
    retireEarly();
    println(money);
}

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




```
1200000
```

Same Variable Name

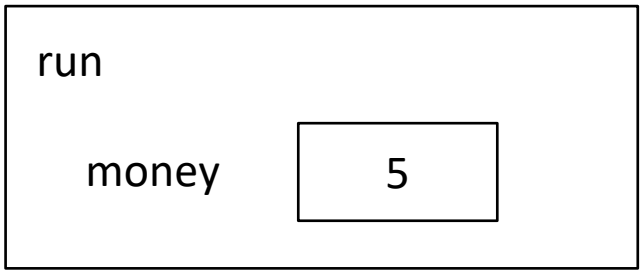
```
private void run() {
    int money = 5;
    retireEarly();
    println(money);
}

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




```
1200000
```

Same Variable Name

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

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




```
1200000
5
```

No Methods in Methods

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

Illegal modifier for parameter goodbye, only final is permitted

No Methods in Methods

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

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


Remember Booleans?

Boolean Variable

```
boolean karelIsAwesome = true;

boolean myBool = 1 < 2;
```


Is Divisible By

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


Boolean Return

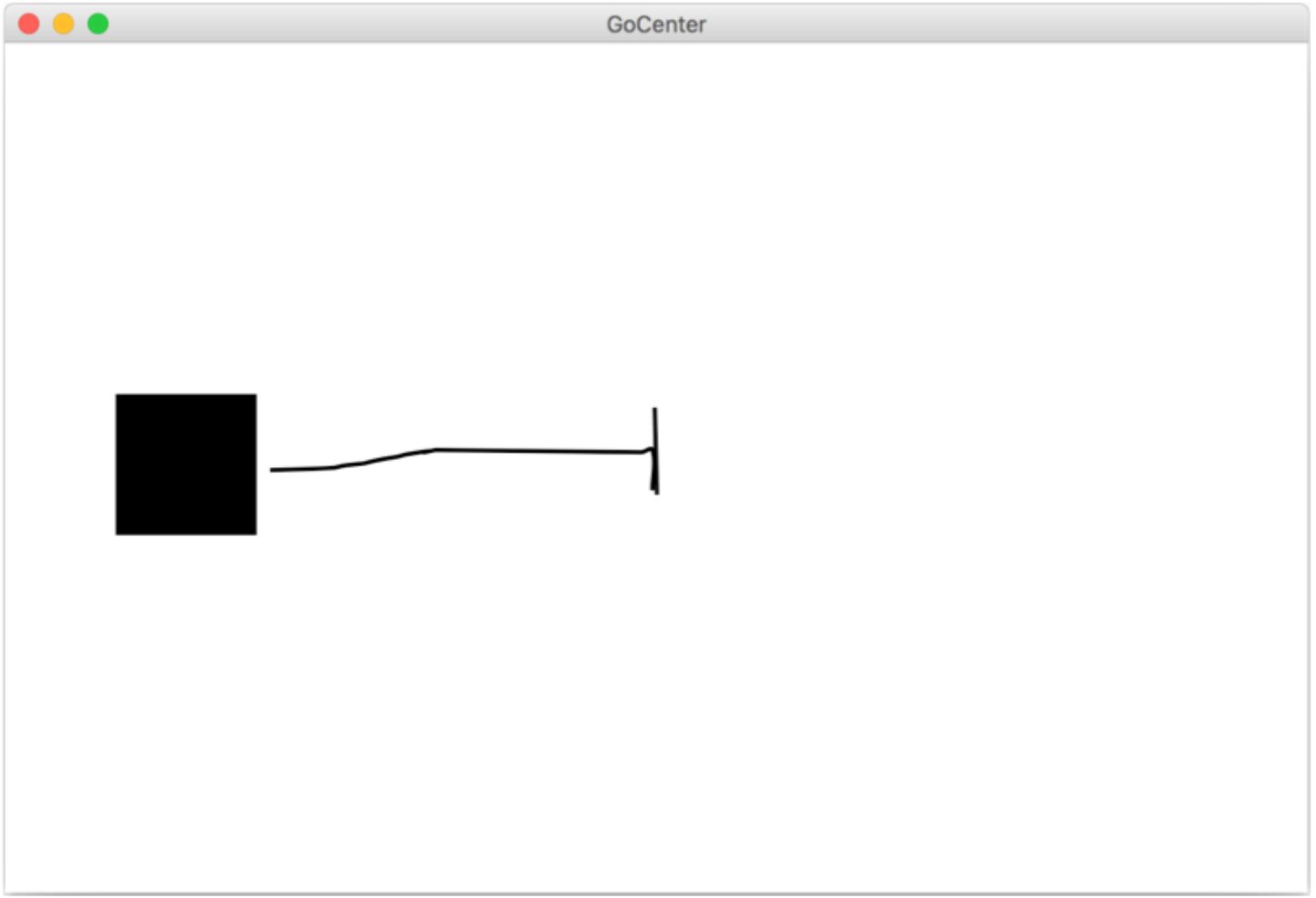
```
private void run() {
   for(int i = 1; i <= 100; i++) {
      if(isDivisibleBy(i, 7)) {
         println(i);
      }
   }
}

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


Methods are broccoli

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

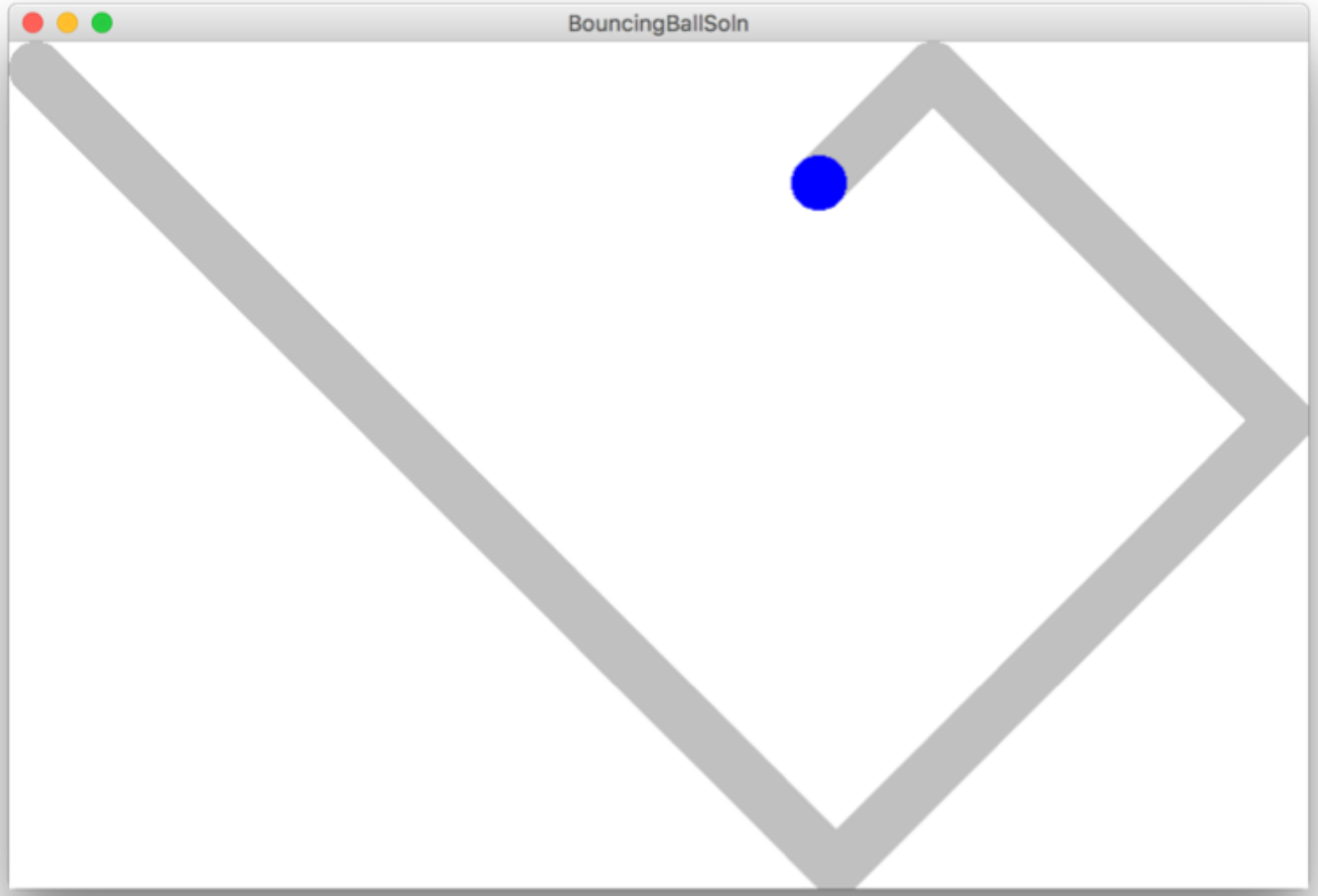
```
private void run() {
    // setup

    while(true) {
        // update world

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

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

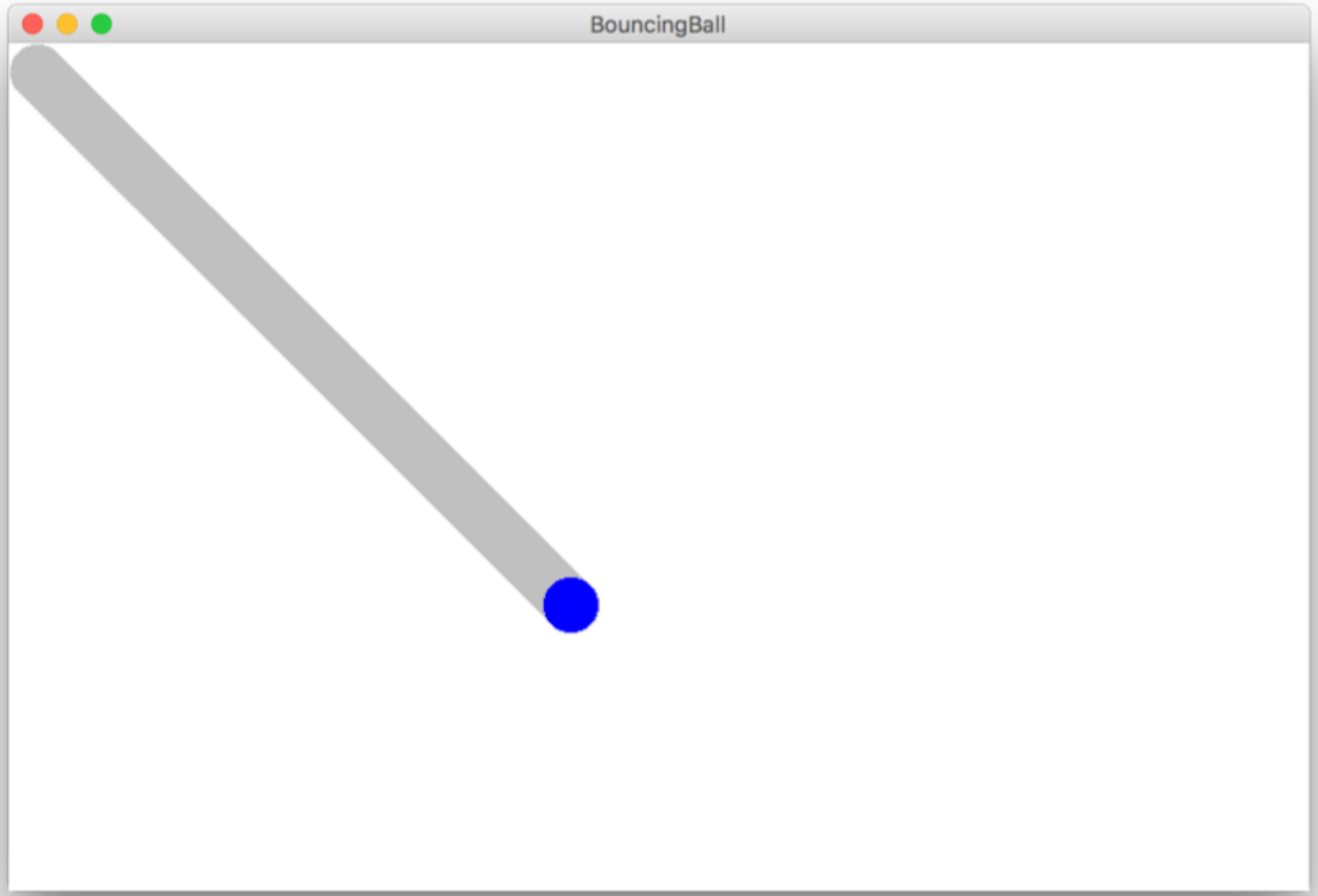
Move To Center

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


Bouncing Ball

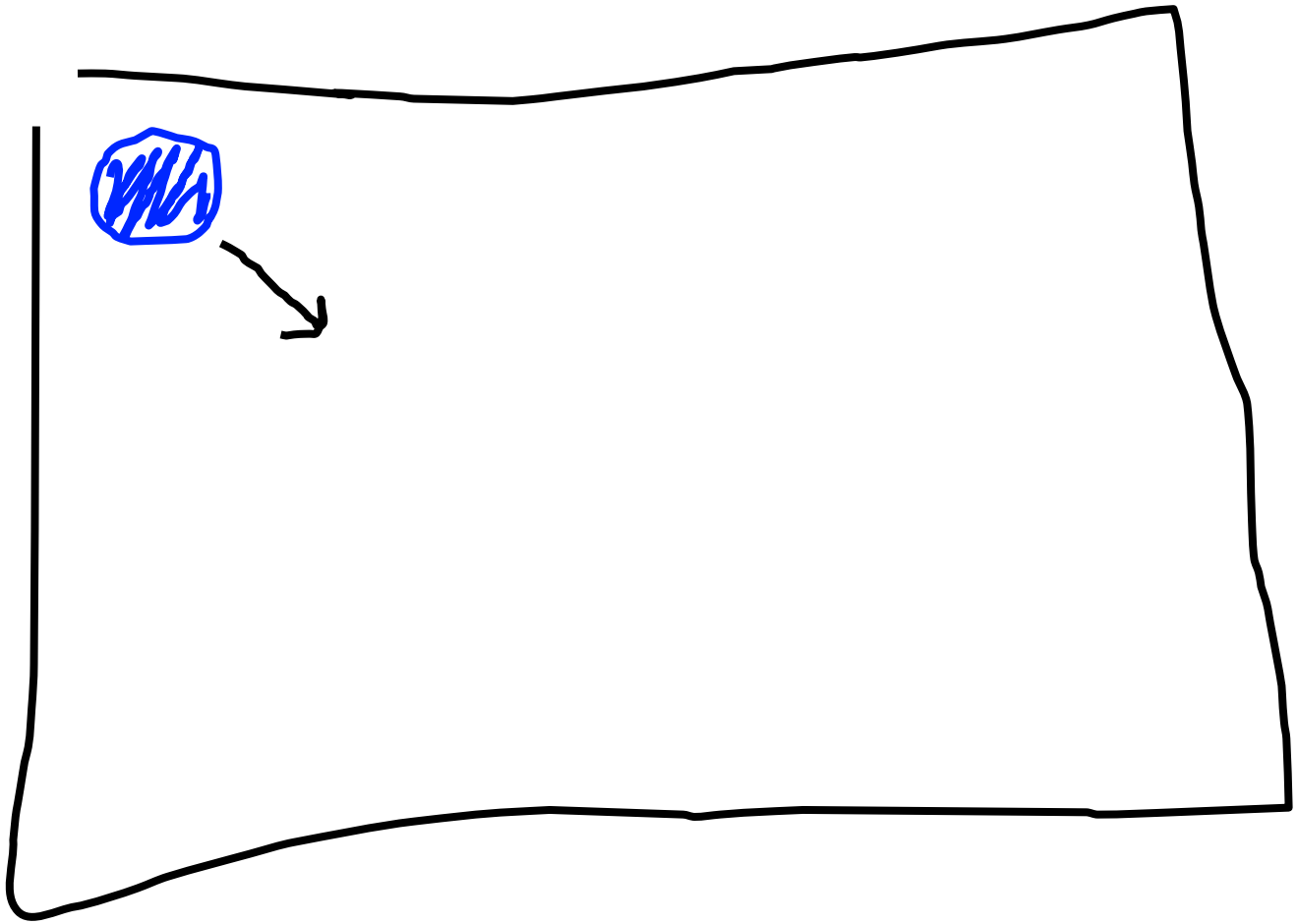


Milestone #1



Bouncing Ball

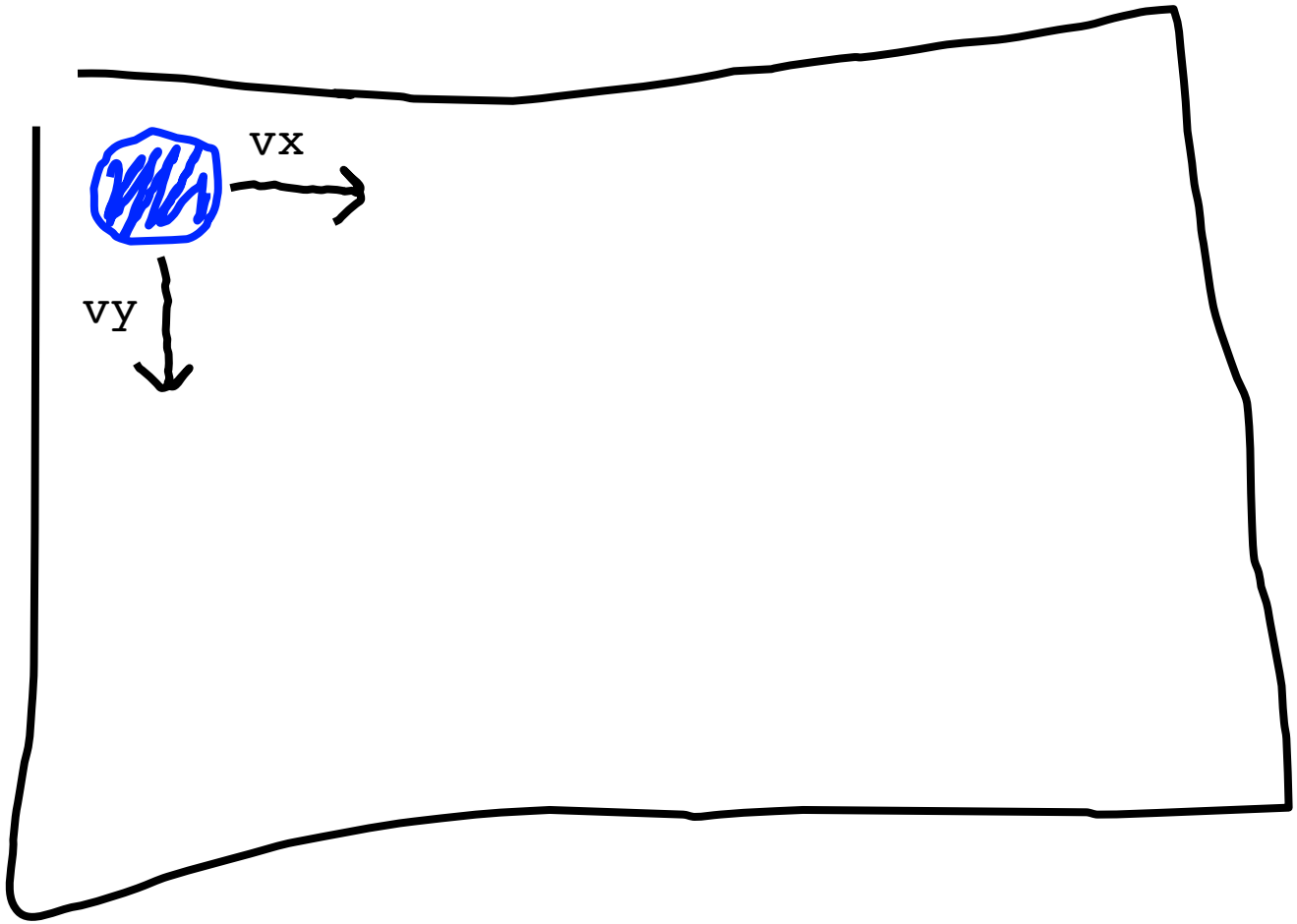
First heartbeat



Velocity: how much the ball position changes each heartbeat

Bouncing Ball

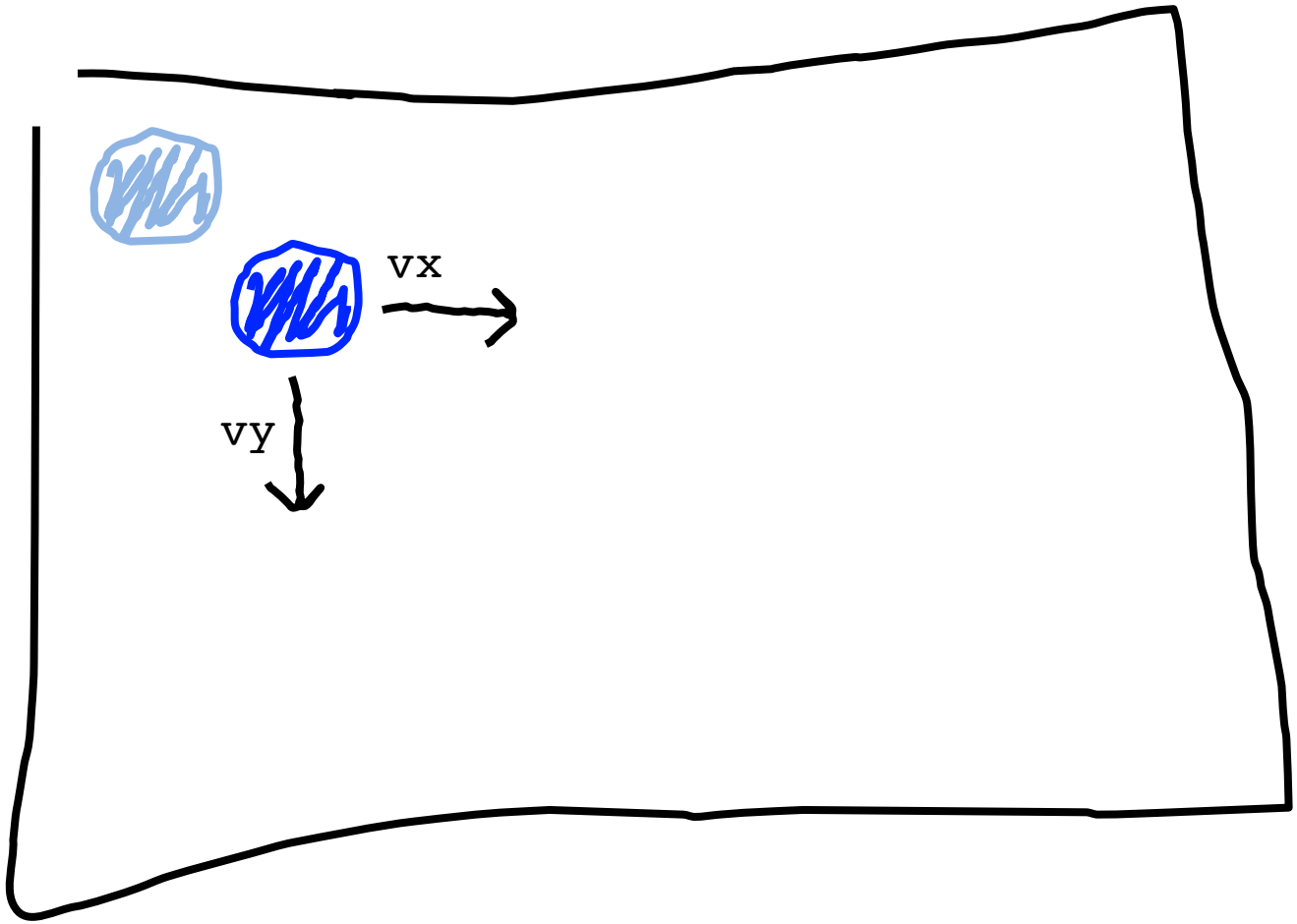
First heartbeat



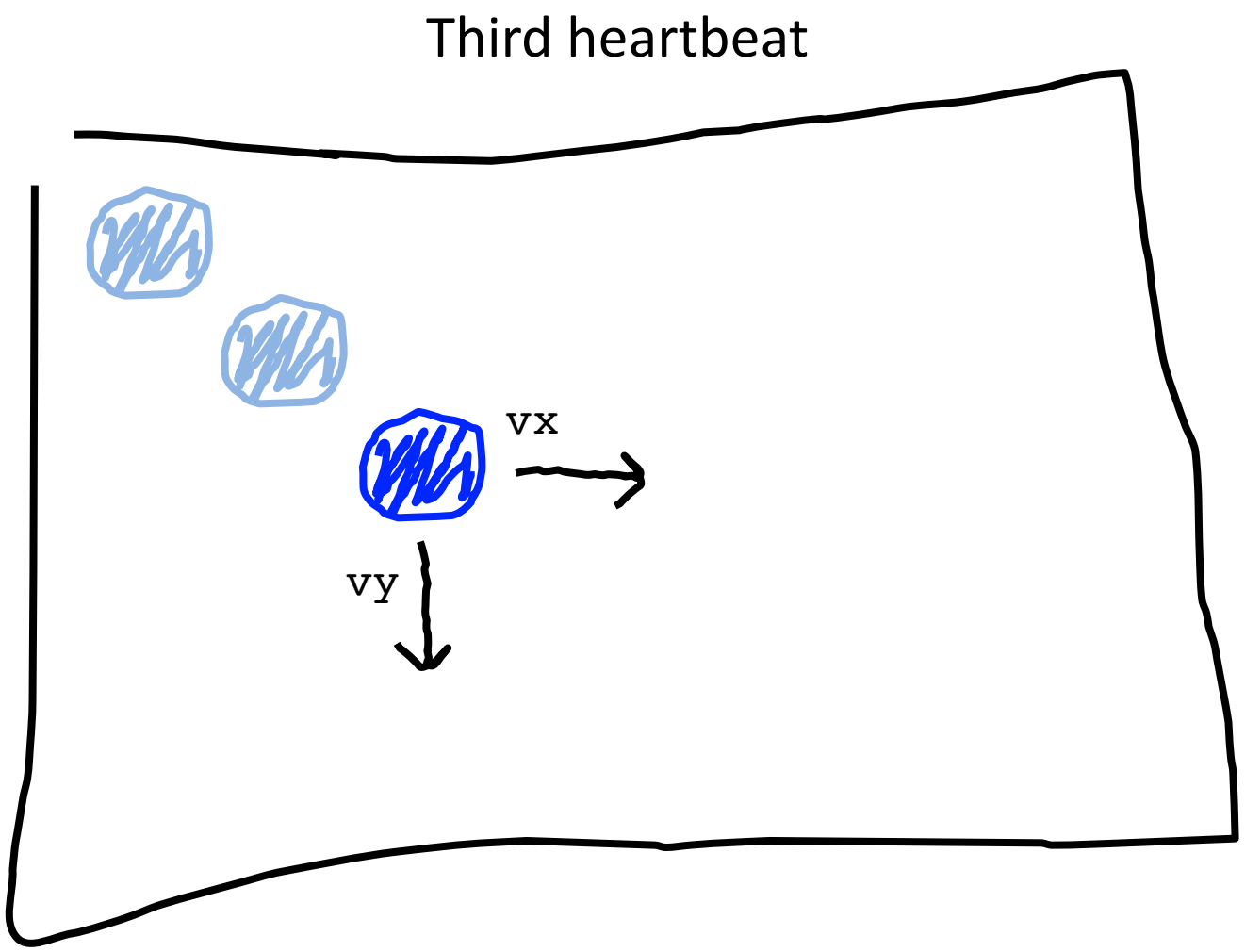
The `GOval` **`move`** method takes in
a change in x and a change in y

Bouncing Ball

Second heartbeat

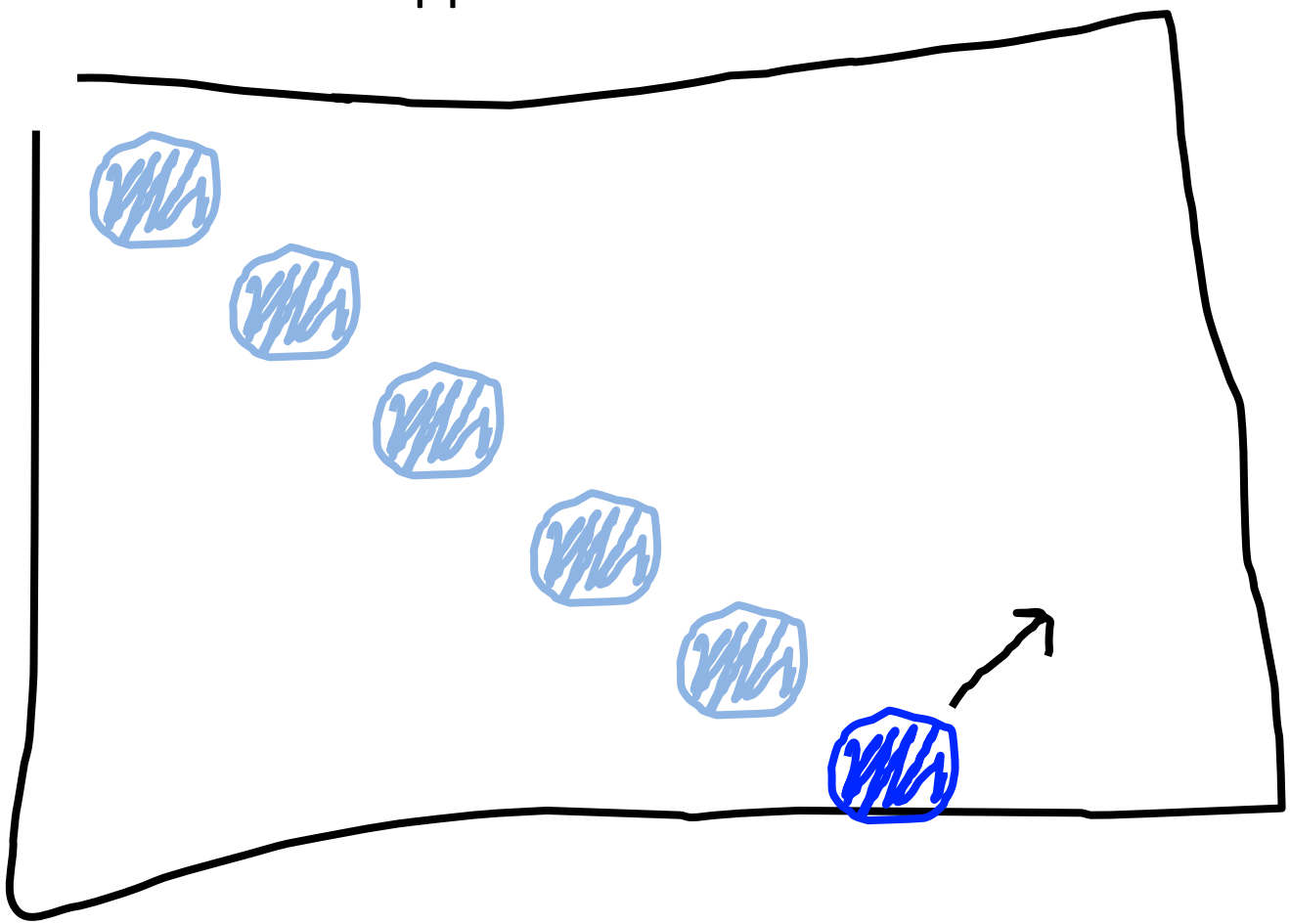


Bouncing Ball



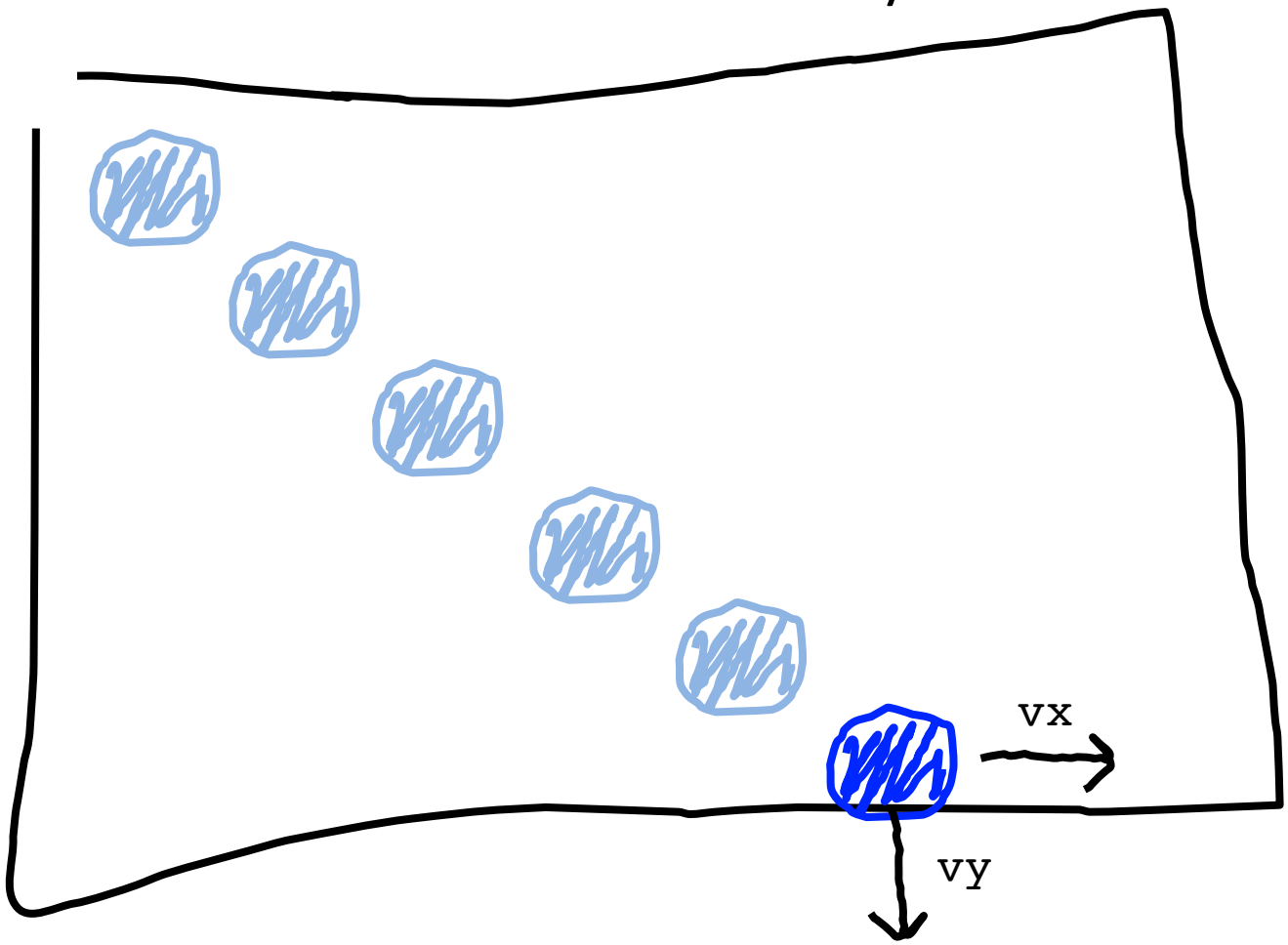
Bouncing Ball

What happens when we hit a wall?



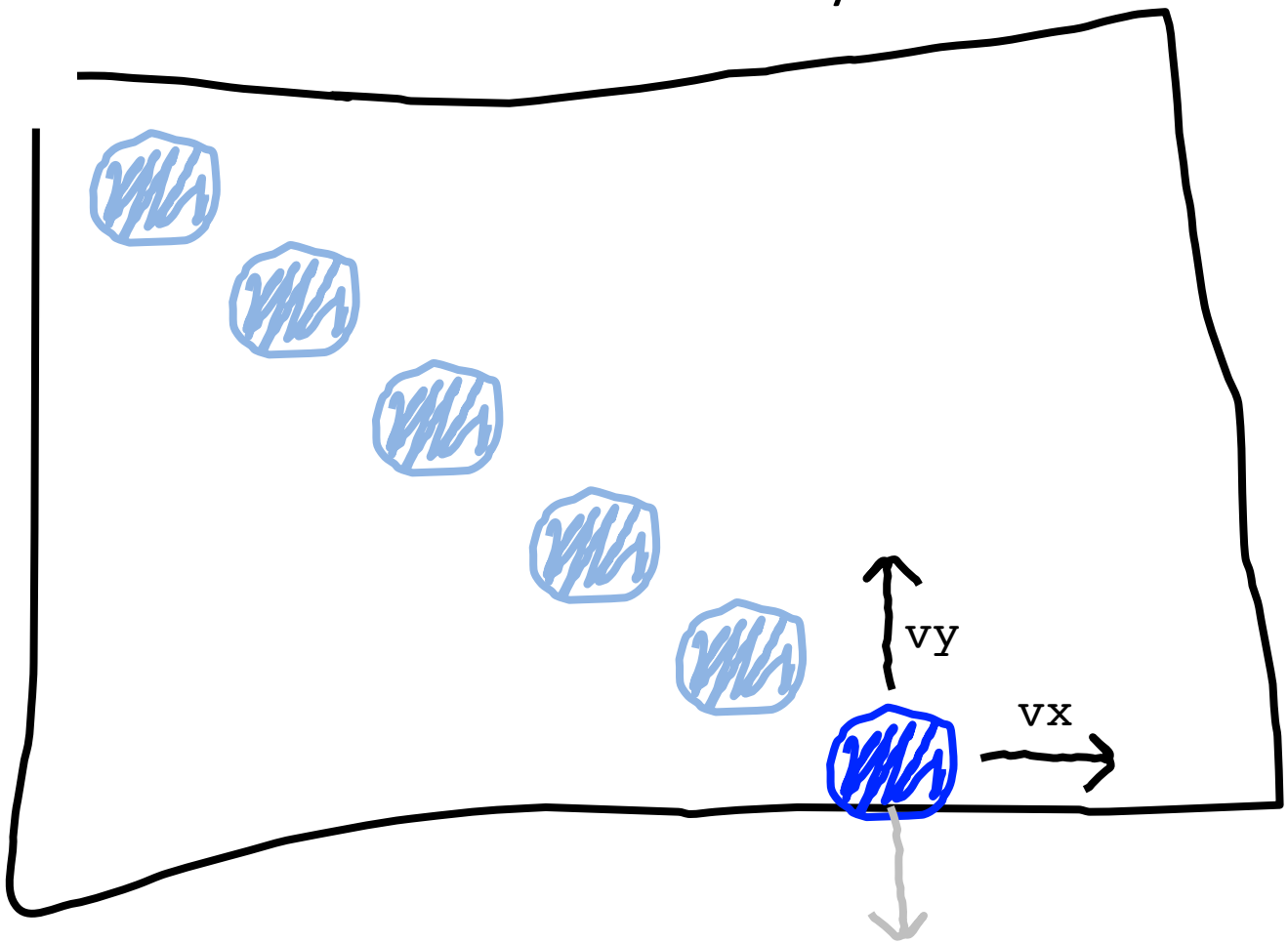
Bouncing Ball

We have this velocity



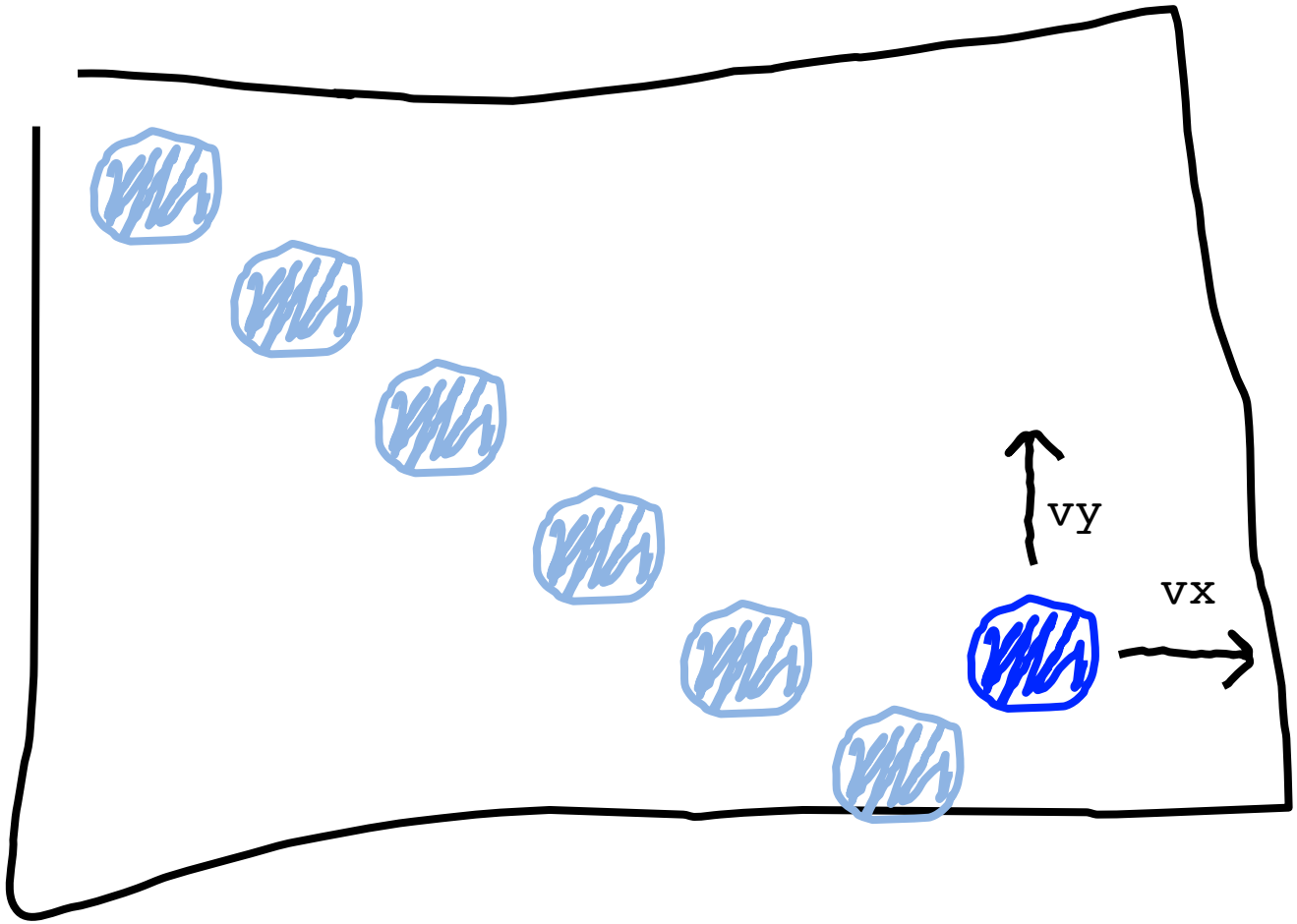
Bouncing Ball

Our new velocity

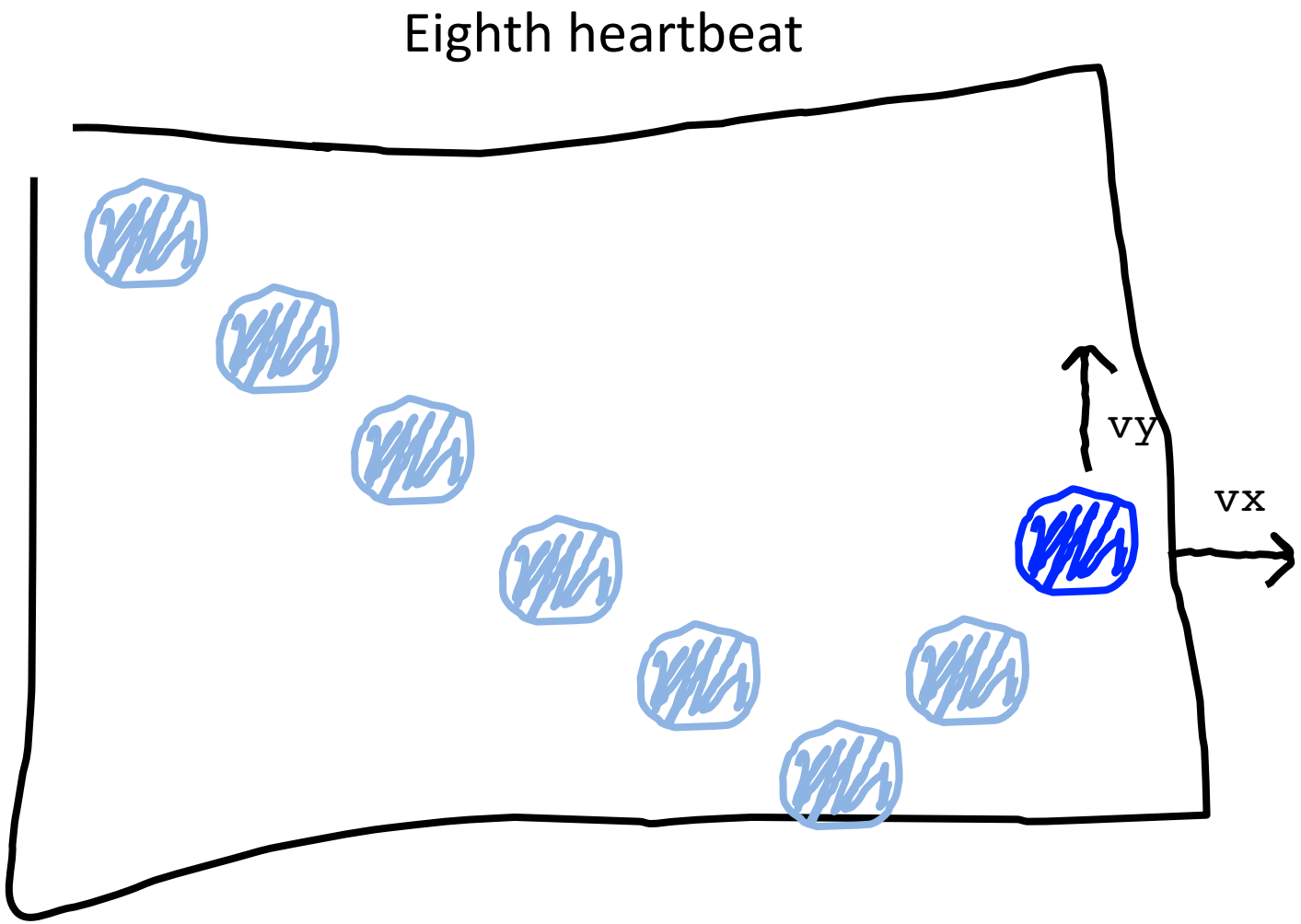


Bouncing Ball

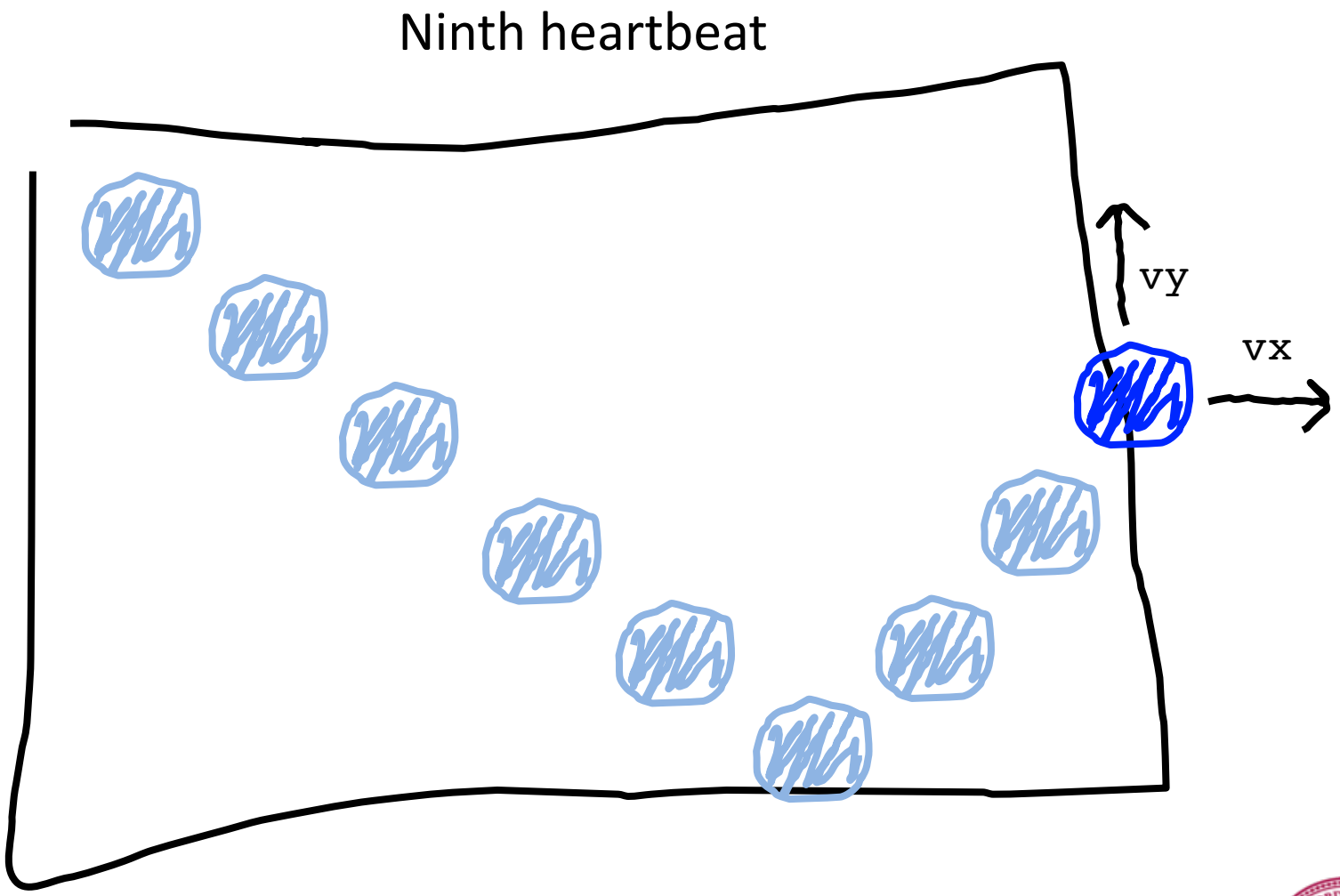
Seventh heartbeat



Bouncing Ball

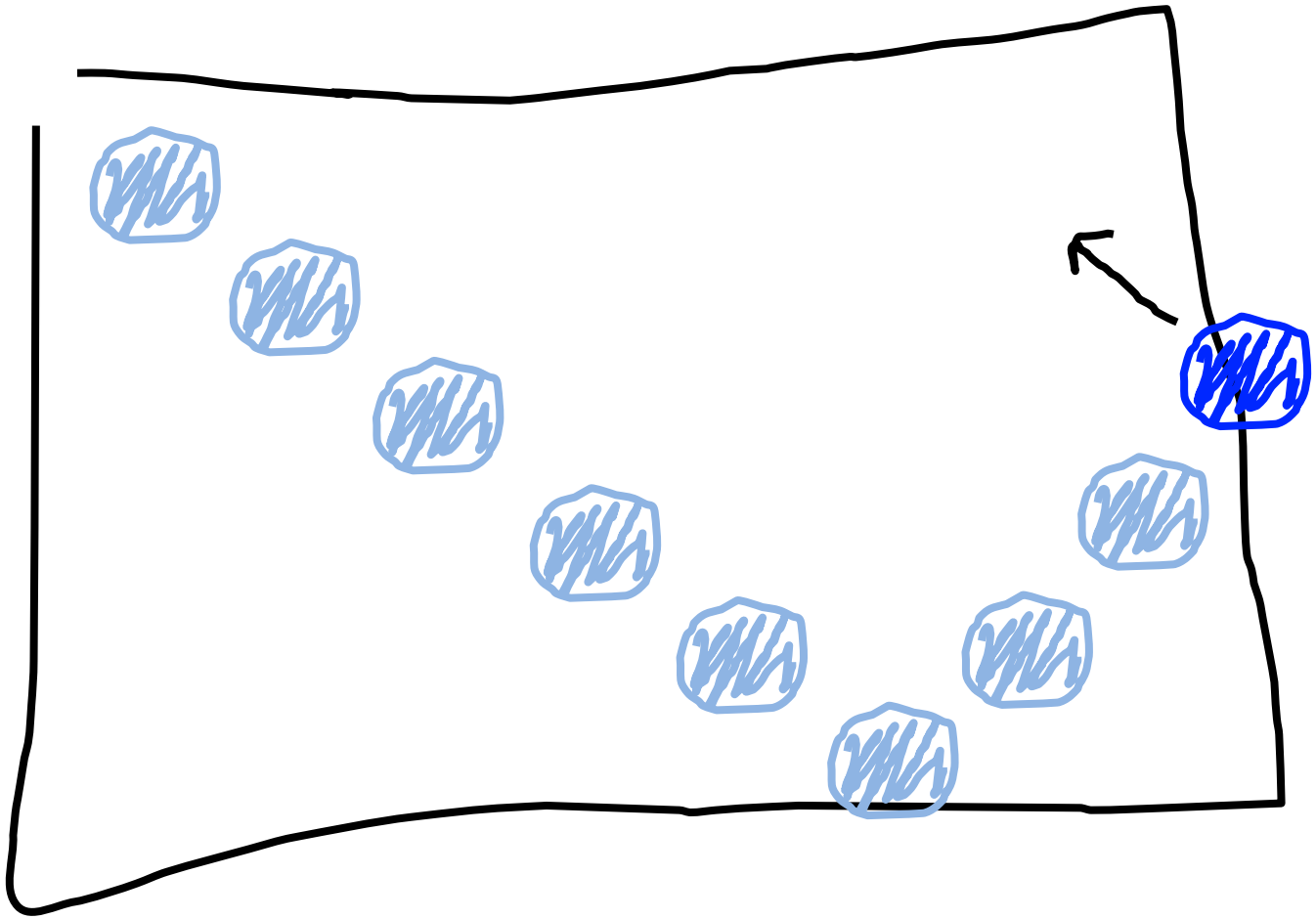


Bouncing Ball



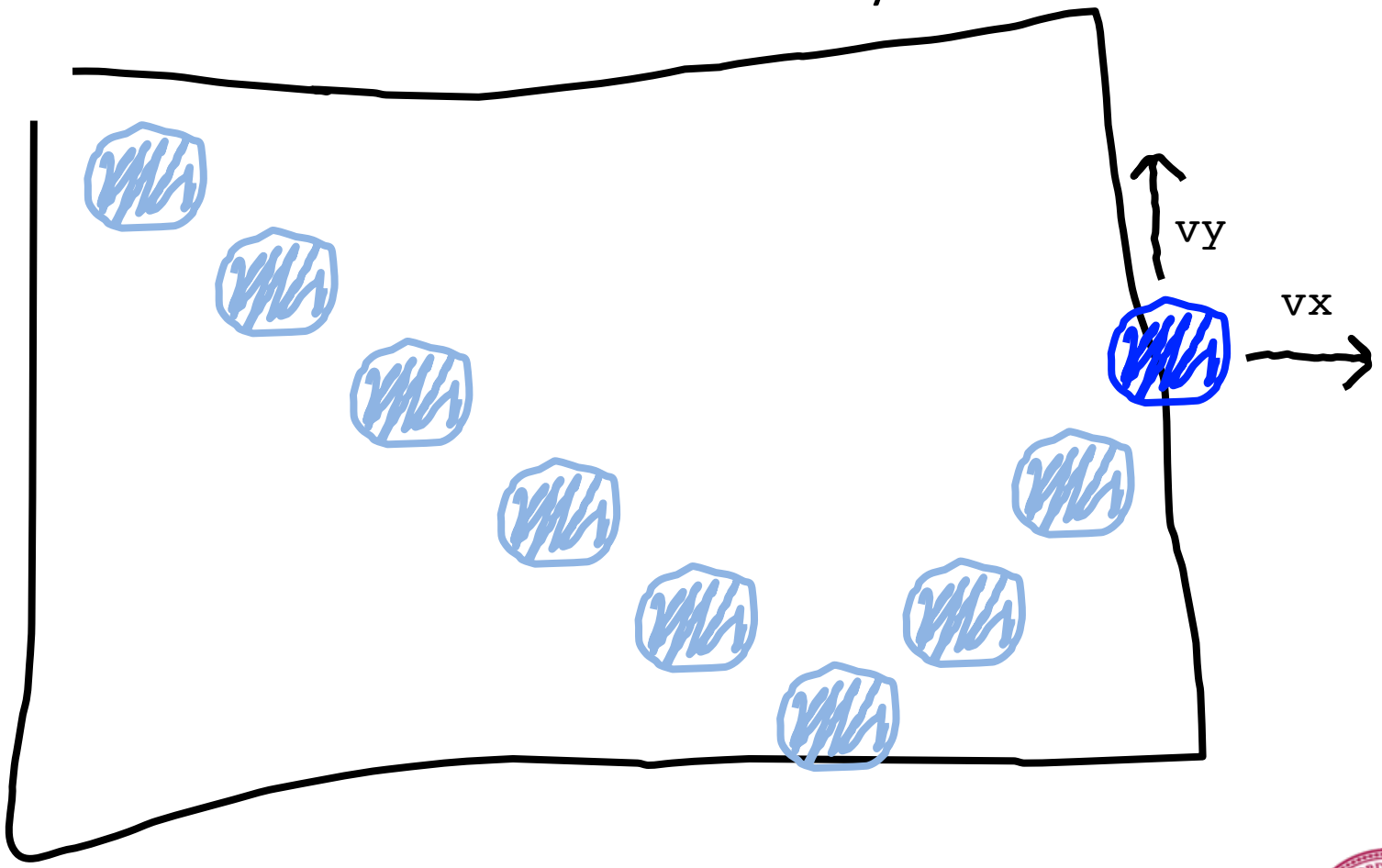
Bouncing Ball

We want this!



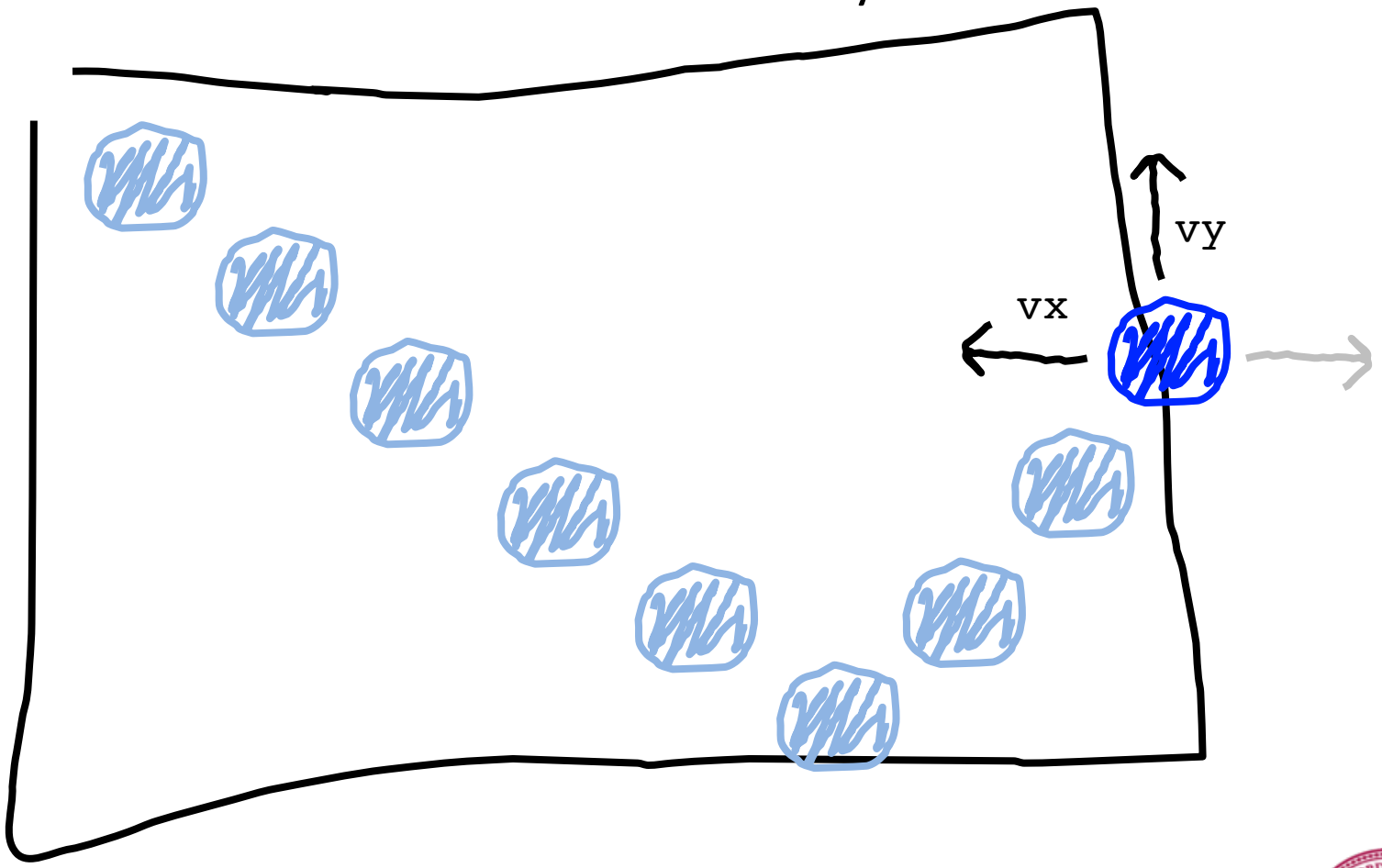
Bouncing Ball

This was our old velocity



Bouncing Ball

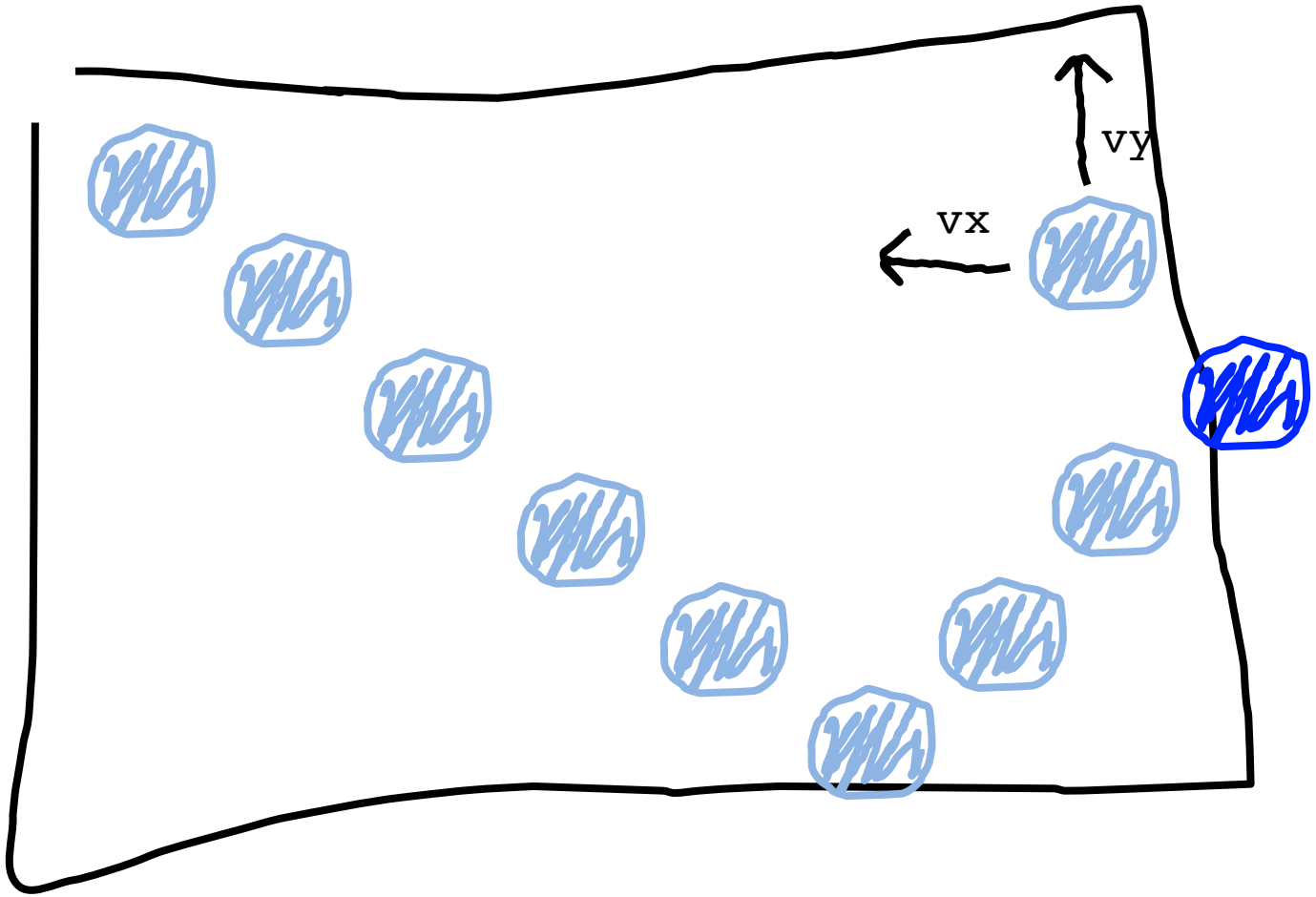
This is our new velocity



When reflecting horizontally: `vx = -vx`

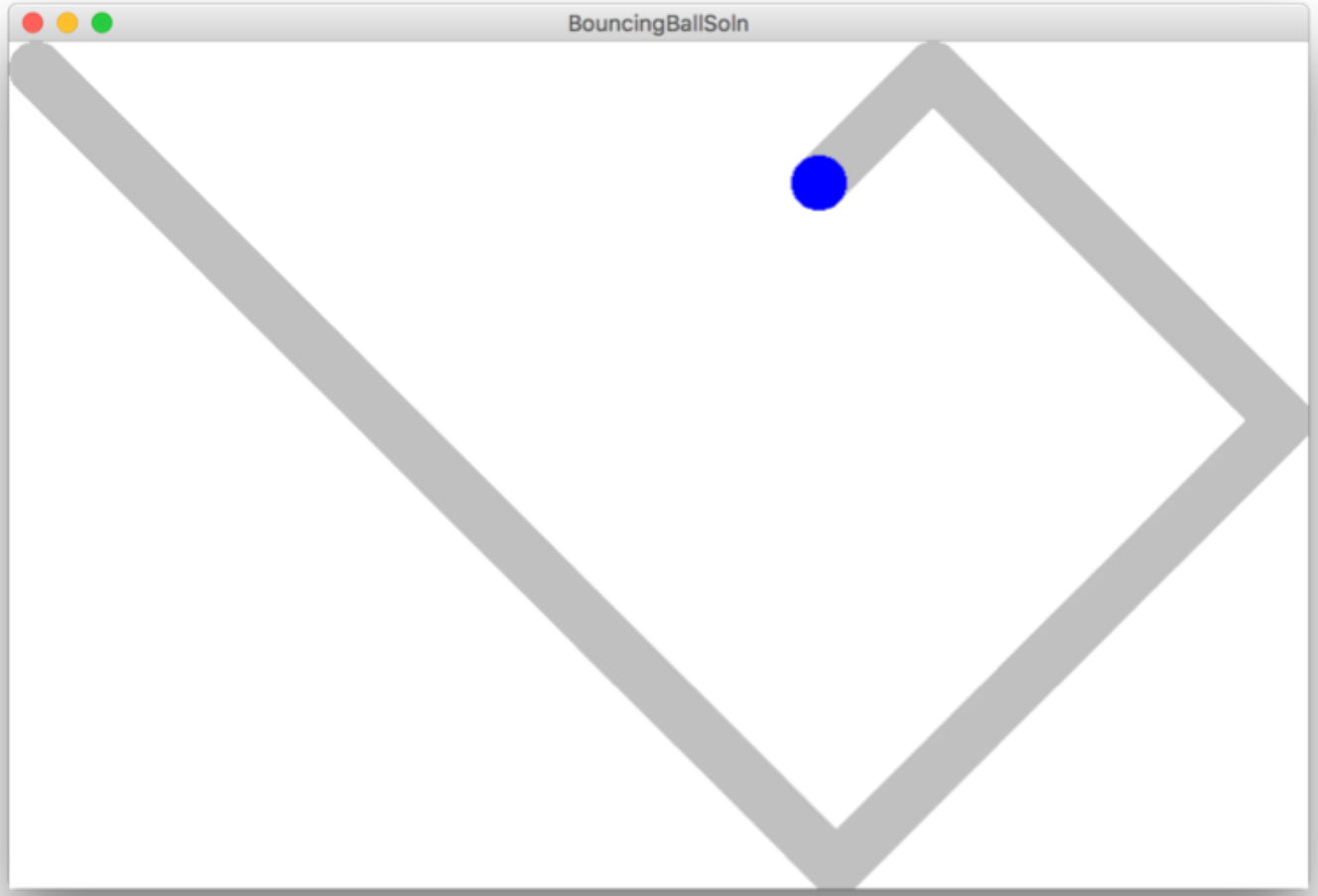
Bouncing Ball

Tenth heartbeat

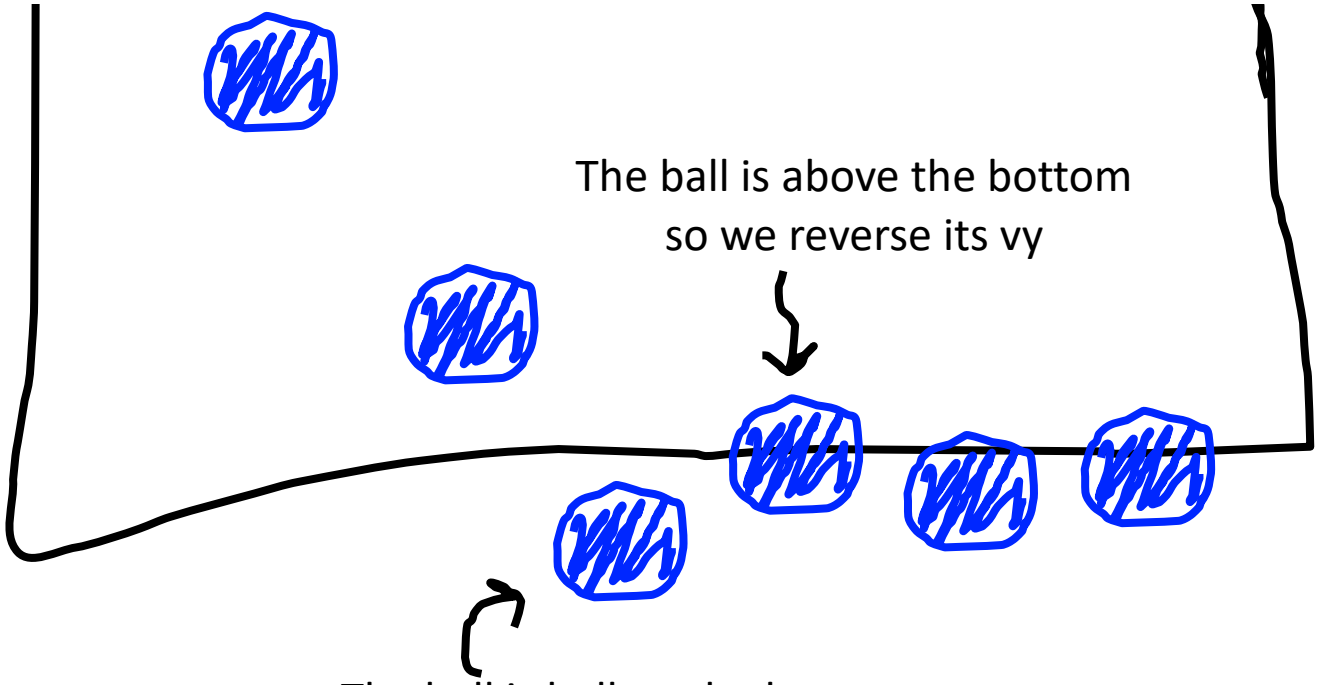


When reflecting horizontally: `vx = -vx`

Bouncing Ball



A Sticky Situation



Learning Goals

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