PAT 204/504 (Fall 2024)

Creative Coding

Lecture 12: Midterm Assignment Showcase & Review

Instructor: Hao-Wen Dong

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

Midterm Assignment Showcase

- **Show us your work!**
- And then, please tell us
  - **What's the concept of your design?**
  - **How did you implement your desired features?**
  - **Is there something new you used?**
  - **What's the most challenging component?**
  - **What's something you'd like to add (or simplify) if given more time?**

Review – Basics

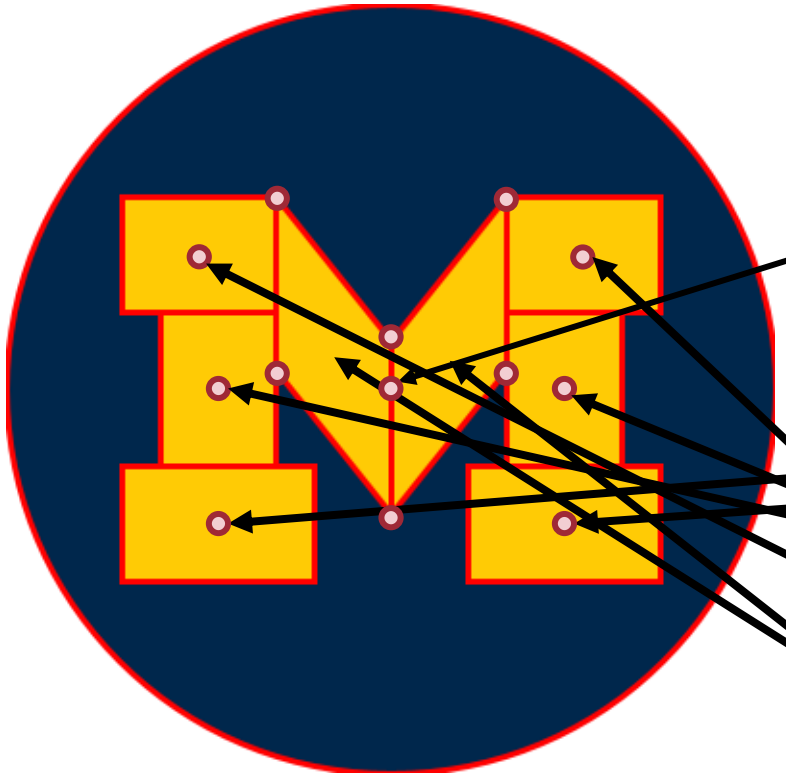
More Shapes

- 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)`

Can you recreate the Block M in Processing?

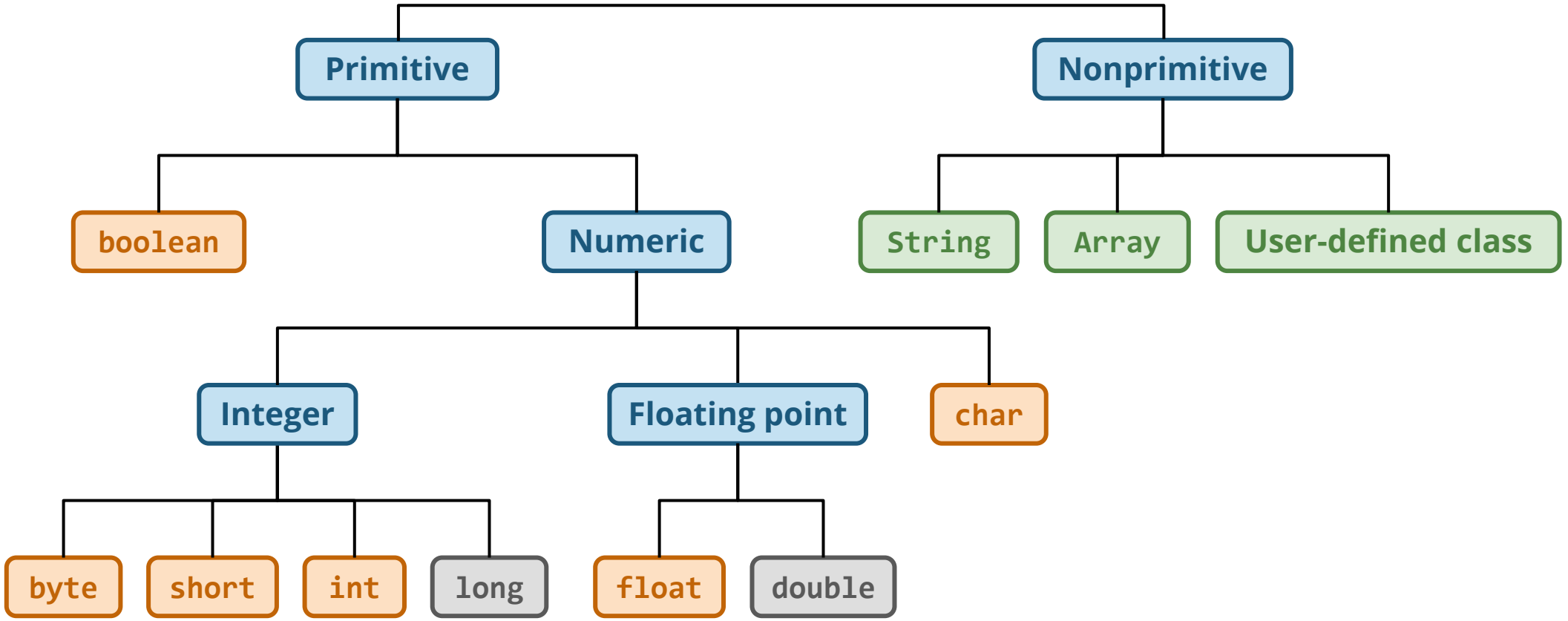
- Michigan **Blue**: **#00274C**
- Michigan **Maize**: **#FFCB05**

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);
}
```

Data Types


Primitive
Nonprimitive
boolean
Numeric
String
Array
User-defined class
Integer
Floating point
char
byte
short
int
long
float
double

Primitive Data Types

	Range	Default	Bytes
`boolean`	true, false	false	1
`byte`	-128 ~ 127	0	1
`int`	-2^{31} ~ $2^{31}-1$	0	4
`long`	-2^{63} ~ $2^{63}-1$	0	8
`float`	±1.4 E-45 ~ ±3.4 E38, ±∞, nan	0.0	4
`double`	±4.9 E-324 ~ ±1.8 E308, ±∞, nan	0.0	8
`color`	#00000000 ~ #FFFFFFFF	#00000000 (black)	4
`char`	0 to 65535 (letters, numbers, symbols, etc.)	'\u0000' (null character)	2

Many Ways to Represent a Color

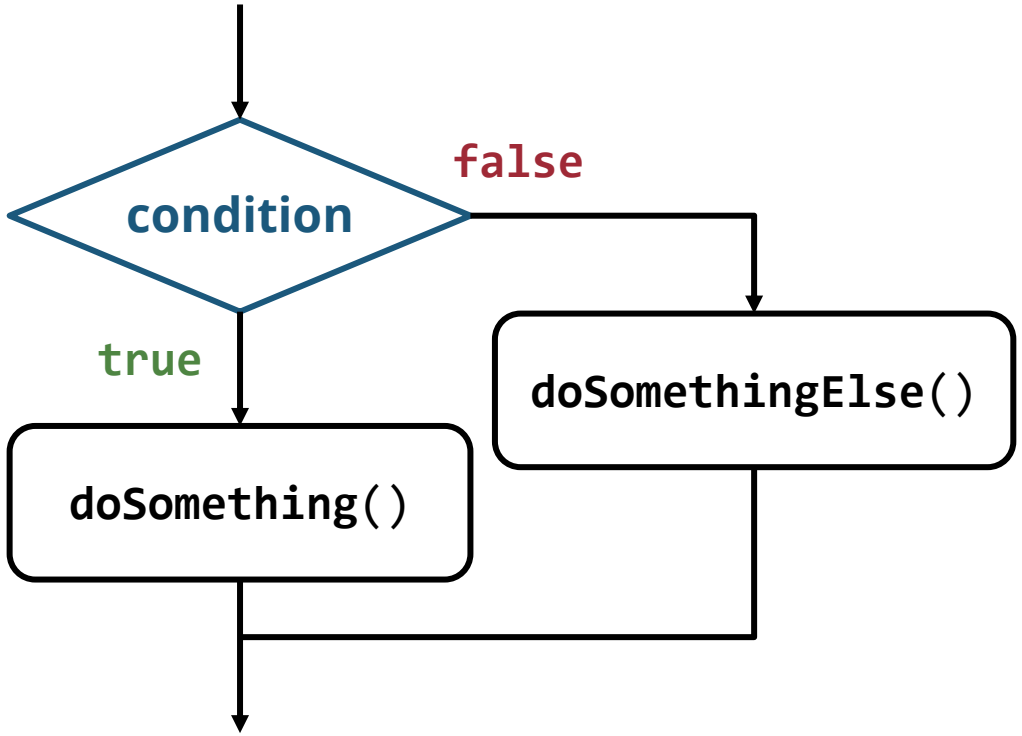
- `fill(grayscale)`
- `fill(R, G, B)`
- `fill(R, G, B, A)`
- `colorMode(HSB)`
  `fill(H, S, B)`
- `colorMode(HSB)`
  `fill(H, S, B, A)`
- `color c = (0, 39, 76)`
  `fill(c)`

Review – Conditionals

Conditionals – if-else Statement

- Control the program flow based on a **condition**

```
if (condition) {
  doSomething();
} else {
  doSomethingElse();
}
```



Conditionals – if-else-if Statement

```
if (condition) {
  doSomething();
} else if (condition2) {
  doSomethingElse();
} else {
  doYetSomethingElse();
}
```

```
if (condition) {
  doSomething();
} else {
  if (condition2) {
    doSomethingElse();
  } else {
    doYetSomethingElse();
  }
}
```

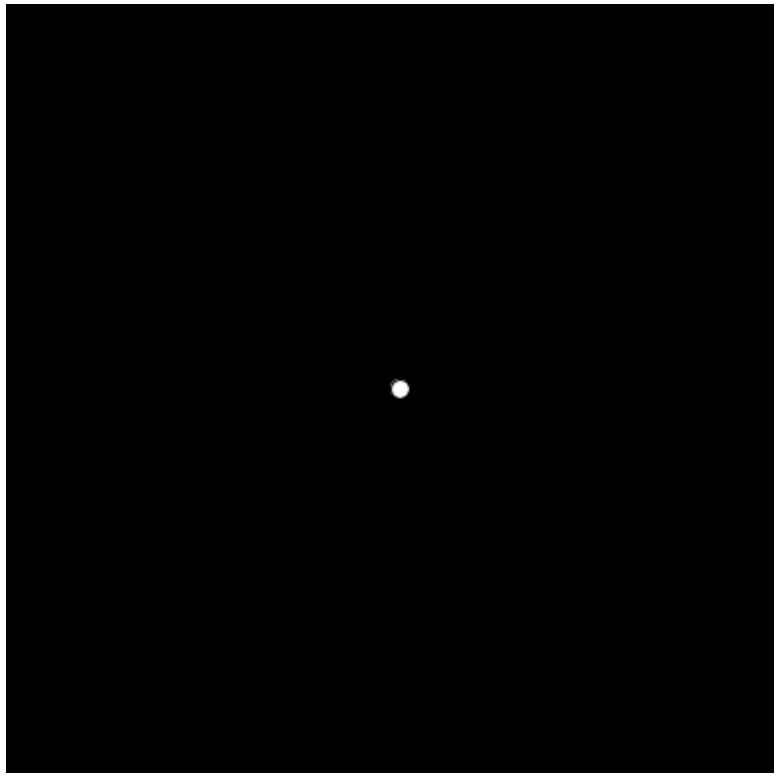
`switch` Statement

```
if (keyCode == LEFT) {
  x = x - step;
} else if (keyCode == RIGHT) {
  x = x + step;
} else if (keyCode == UP) {
  y = y - step;
} else if (keyCode == DOWN) {
  y = y + step;
}
```

```
switch(keyCode) {
  case LEFT:
    x = x - step;
    break;

  case RIGHT:
    x = x + step;
    break;

  case UP:
    y = y - step;
    break;

  case DOWN:
    y = y + step;
    break;
}
```

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);
}
```



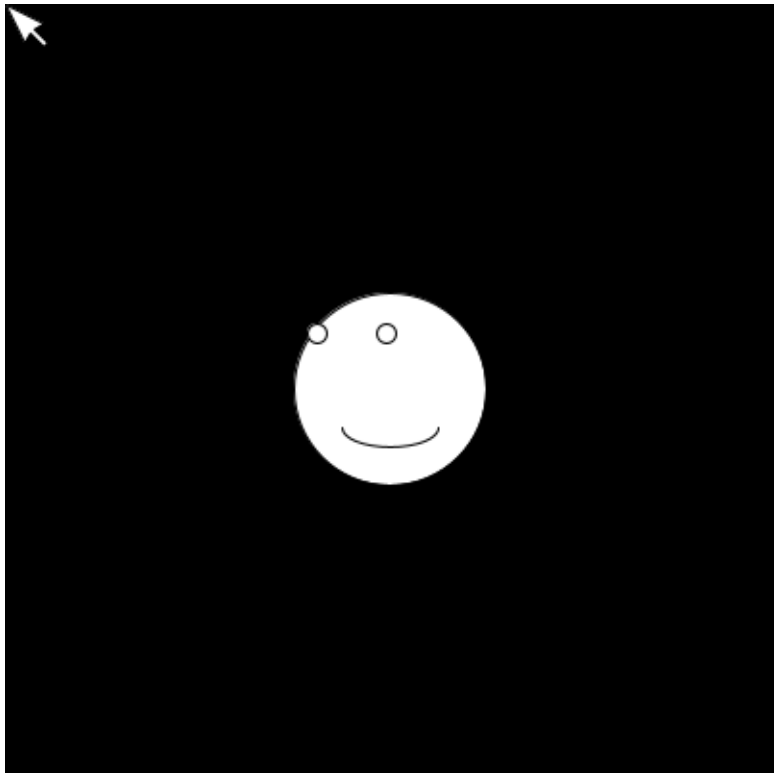
Homework 1: Bouncing Hello World

- 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**

Review – Controls

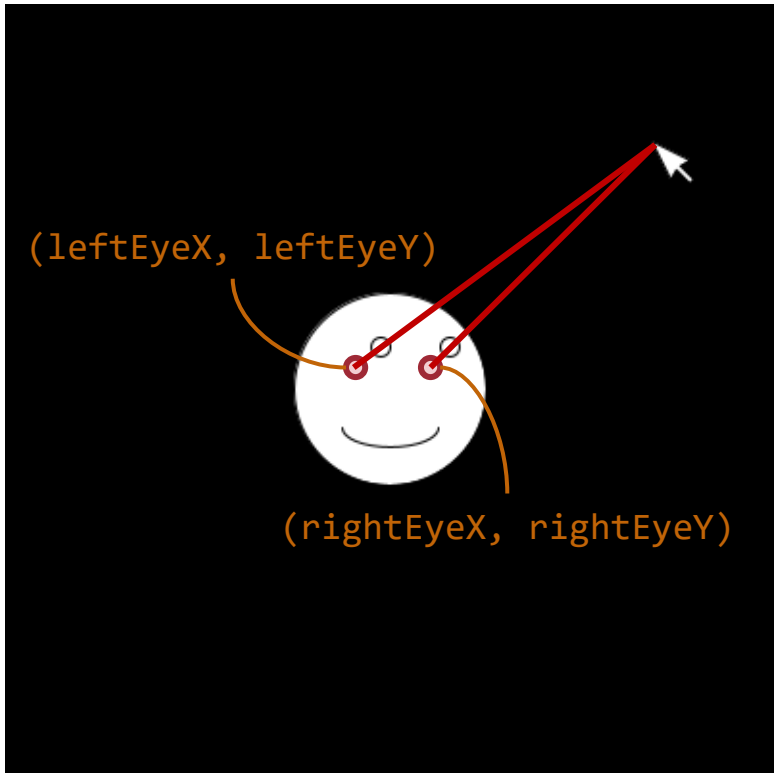
Exercise: ~~Creepy~~ Eyes

- 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)`



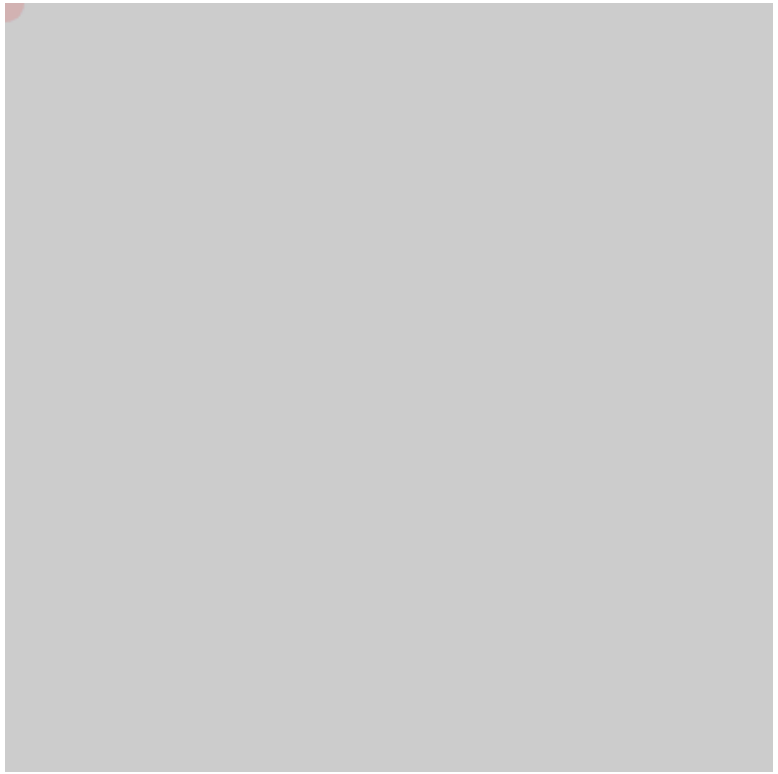
Exercise: ~~Creepy~~ Eyes

```
// 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
);
```



Exercise: Rainbow Paint

- What you'll need:
  - `mouseX` & `mouseY`
  - `colorMode(HSB)` for HSB color mode
  - alpha value for transparency

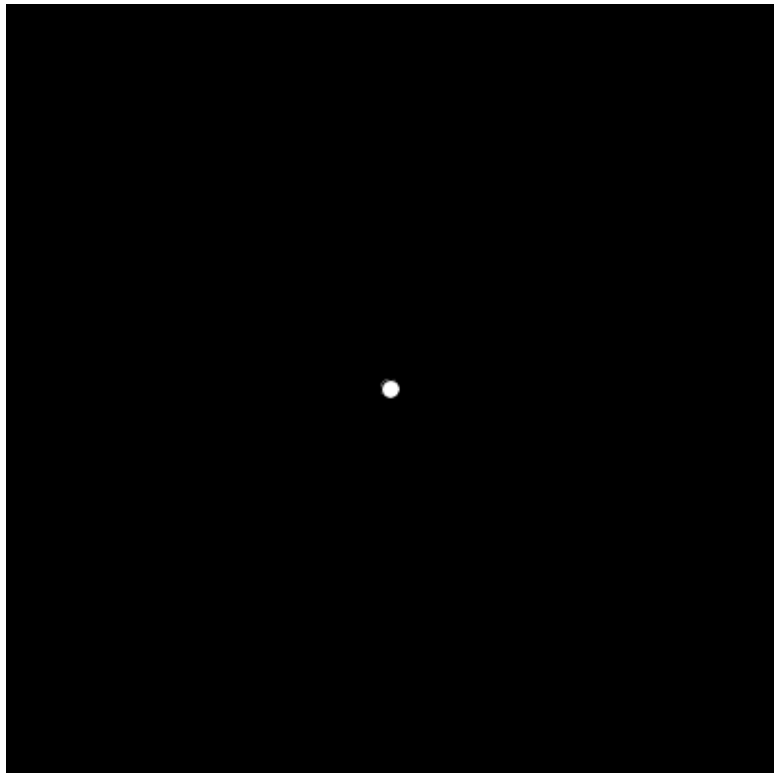


key & keyCode

- key stores the most recent key used (either pressed or released)
- For special (non-ASCII) keys, keyCode is used
  - **UP**, **DOWN**, **LEFT**, **RIGHT**
  - **ALT**, **CONTROL**, **SHIFT**
  - You don't need keyCode for BACKSPACE, TAB, ENTER, RETURN, ESC, and DELETE

Exercise: Use the Arrow Keys to Control a Ball

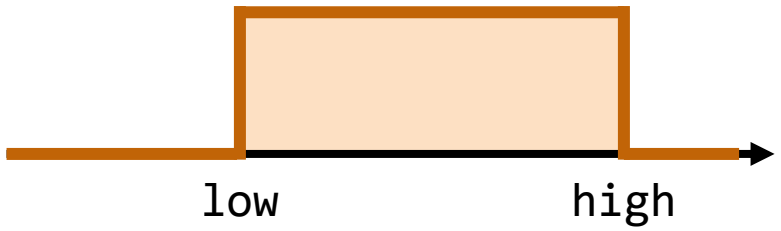
```
float x = 200;
float y = 200;
float step = 10;

void keyPressed() {
  if (key == CODED) {
    if (keyCode == LEFT) {
      x = x - step;
    } else if (keyCode == RIGHT) {
      x = x + step;
    } else if (keyCode == UP) {
      y = y - step;
    } else if (keyCode == DOWN) {
      y = y + step;
    }
  }
}
```

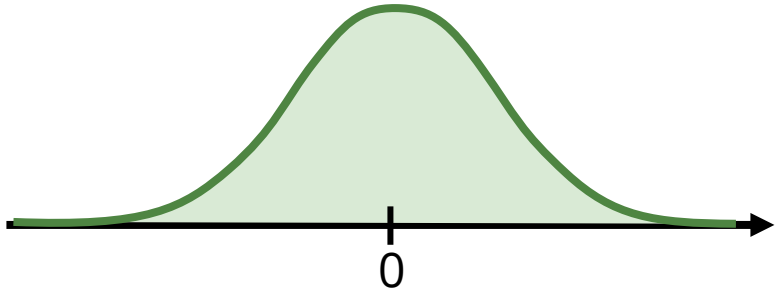


Review – Randomness

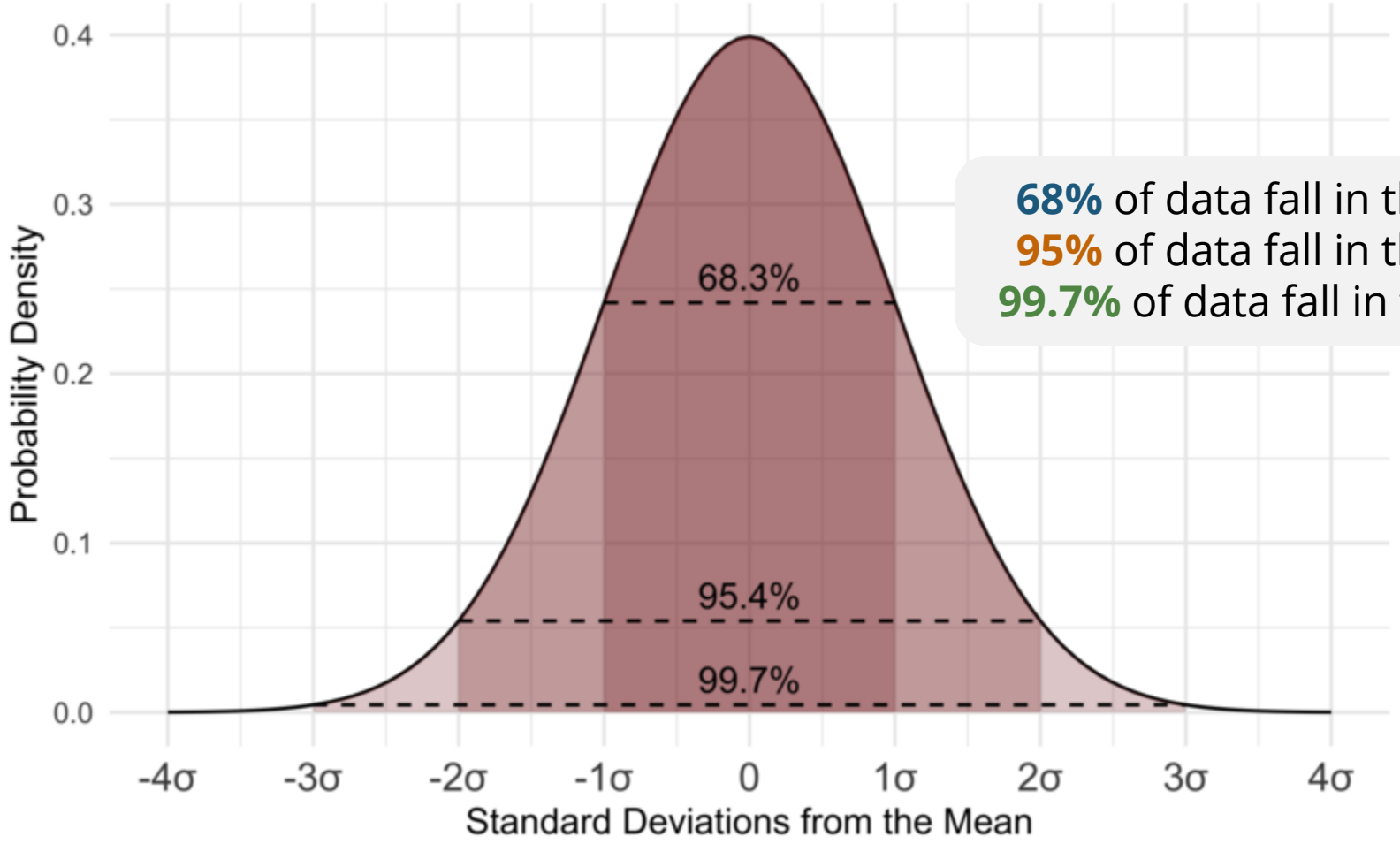
Randomness

- `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 $N[0, 1]$





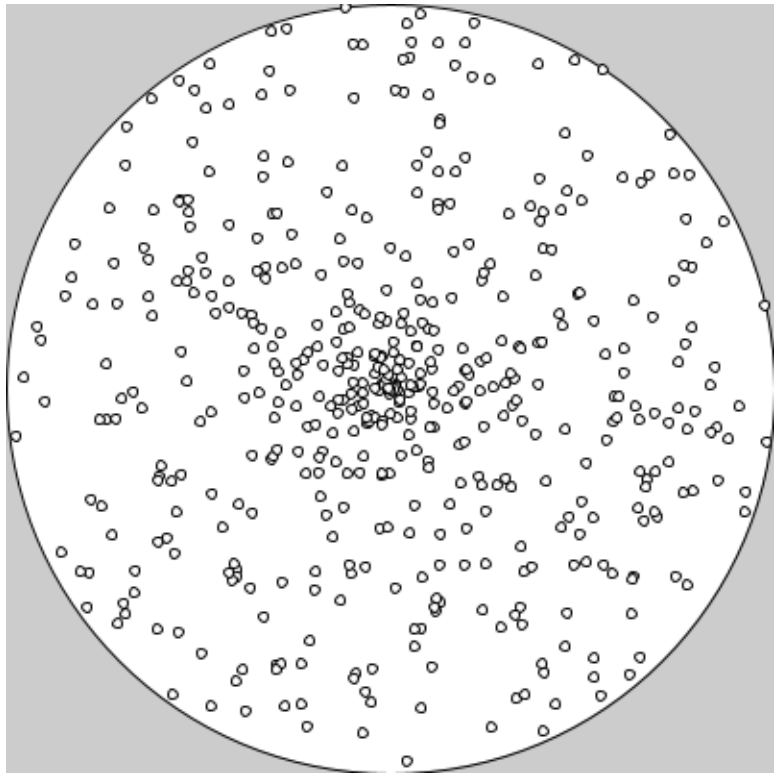
Gaussian Distribution & the 68–95–99.7 Rule



68% of data fall in the **±1σ** range
95% of data fall in the **±2σ** range
99.7% of data fall in the **±3σ** range

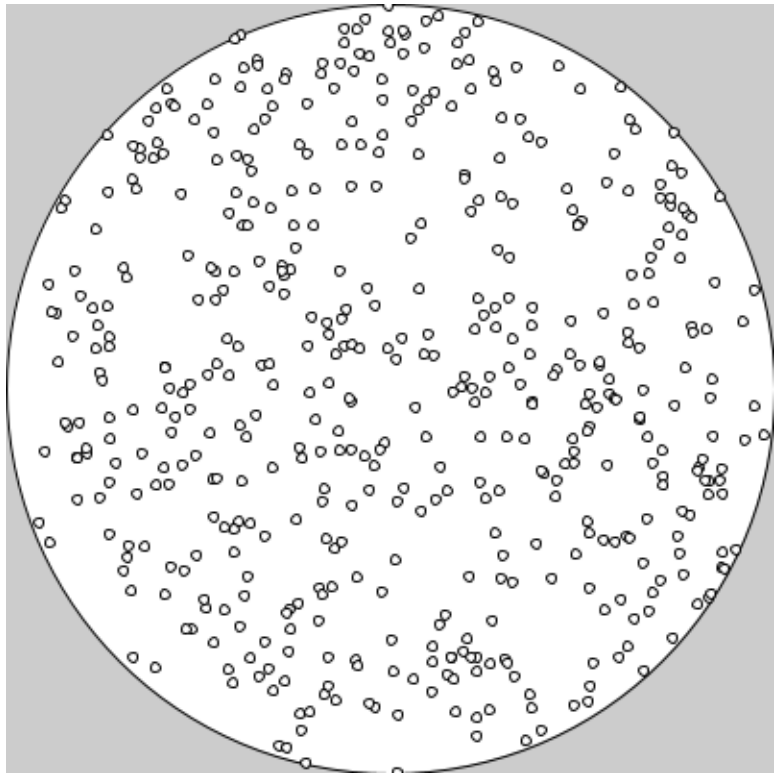
Exercise: Random Points in a Circle (Polar Coordinate Sampling)

```
// Generate a random radius and angle
r = random(200);
theta = random(0, TWO_PI);

// Calculate the x- and y-positions
x = 200 + r * cos(theta);
y = 200 + r * sin(theta);

// Draw a circle
circle(x, y, 5);
```



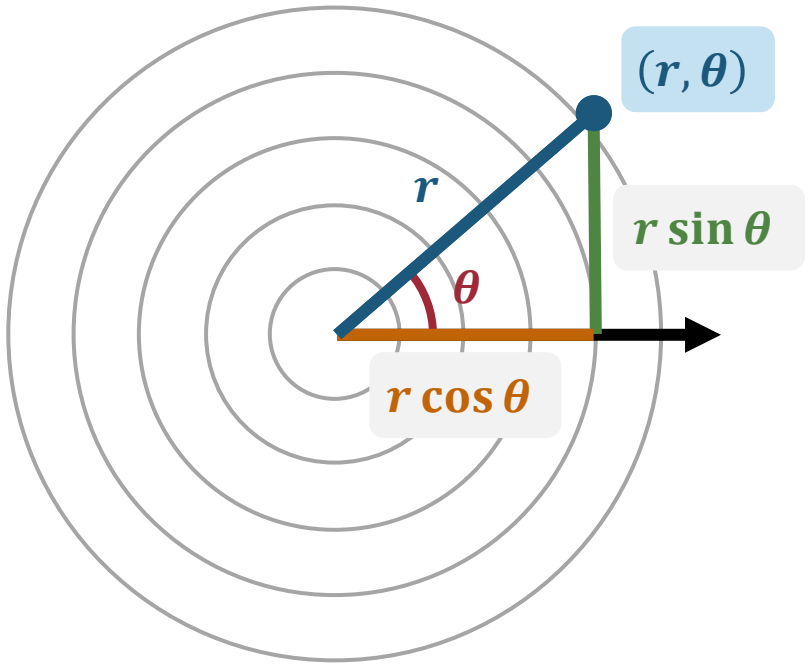
Advanced Method: Inversion Transform Sampling

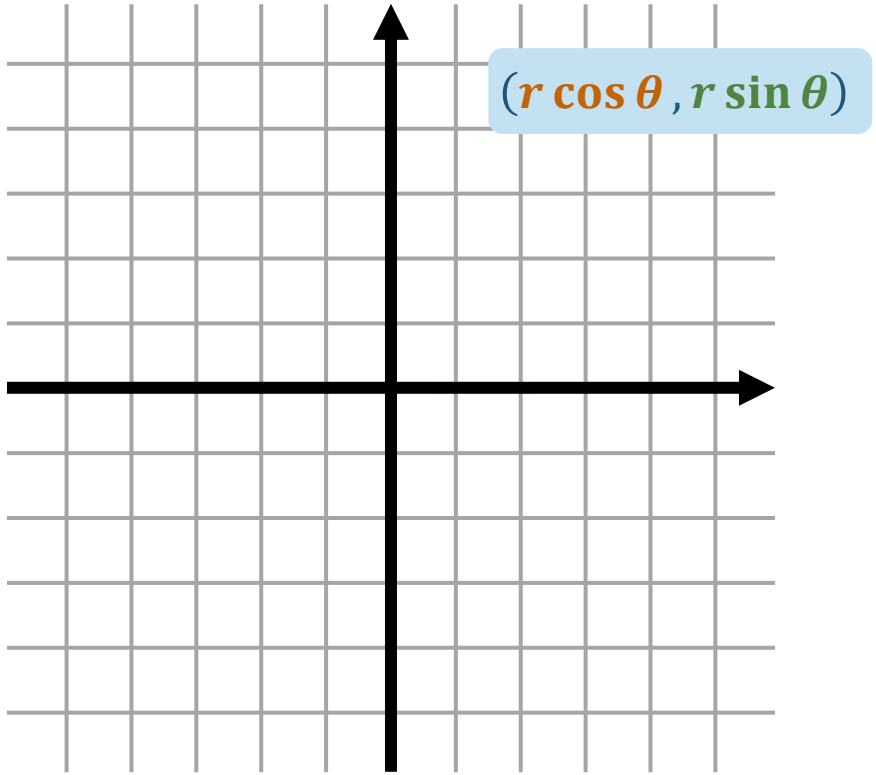
```
// Generate a random radius and angle
r = 200 * sqrt(random(1));
theta = random(0, TWO_PI);

// Calculate the x- and y-positions
x = 200 + r * cos(theta);
y = 200 + r * sin(theta);

// Draw a circle
circle(x, y, 5);
```



Conversion: Polar → Cartesian

Or, you can use `PVector.fromAngle()` and then `setMag()`!



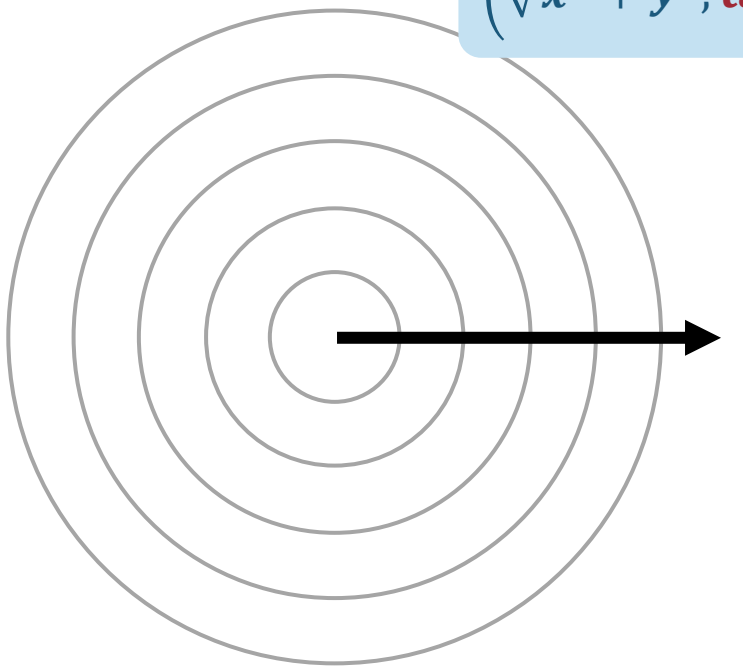


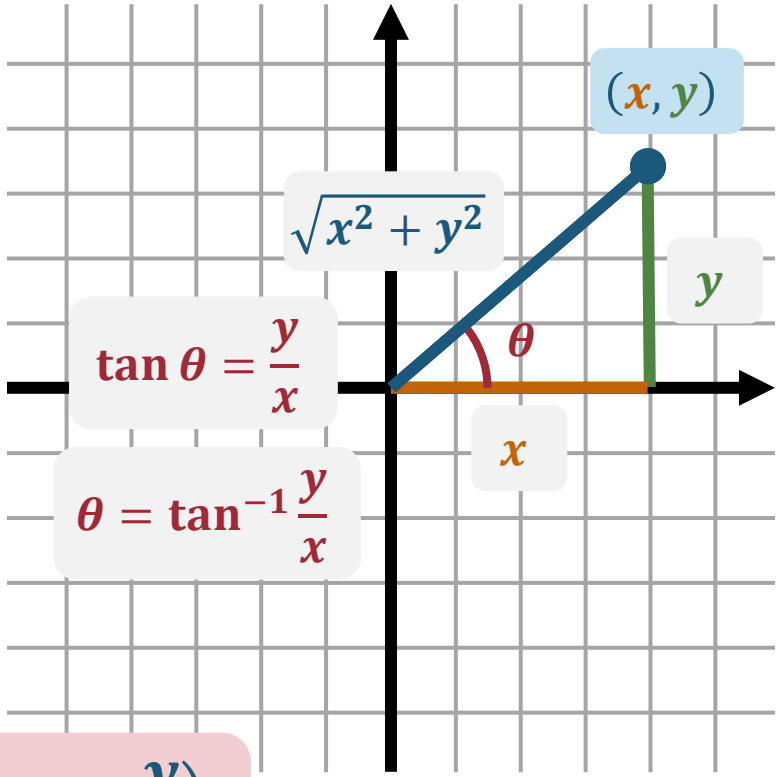
$$(r, \theta) \rightarrow (r \cos \theta, r \sin \theta)$$

Conversion: Cartesian → Polar

Or, you can use a `PVector` and call `mag()` and `heading()`!

$$\left(\sqrt{x^2 + y^2}, \tan^{-1}\frac{y}{x}\right)$$





$$(x, y) \rightarrow \left(\sqrt{x^2 + y^2}, \tan^{-1}\frac{y}{x}\right)$$

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

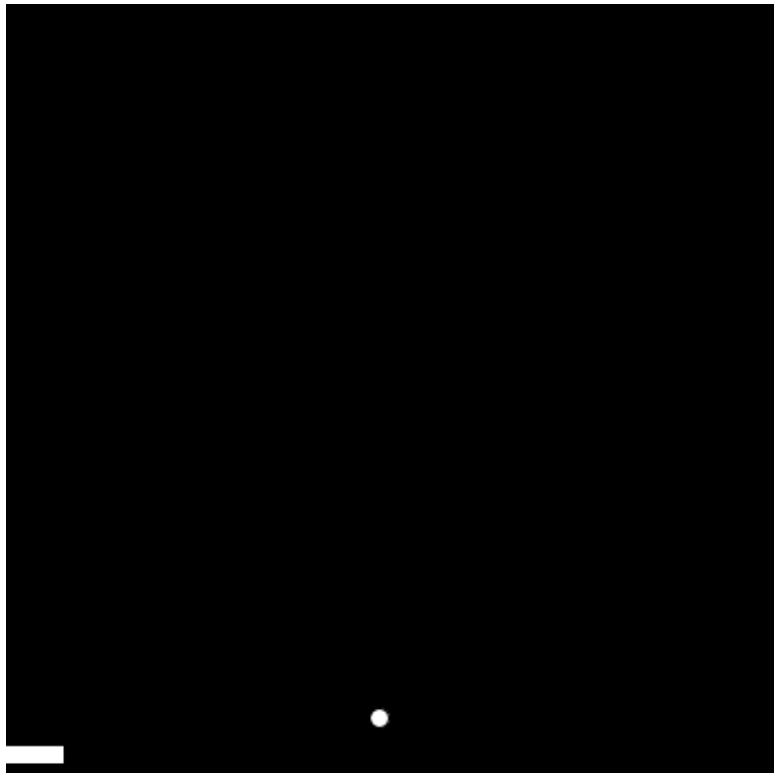
```
PVector v = new PVector(1, 0);
v.rotate(PI / 4);
println(v);
```

Static method

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

Homework 2: Paddle Ball Game

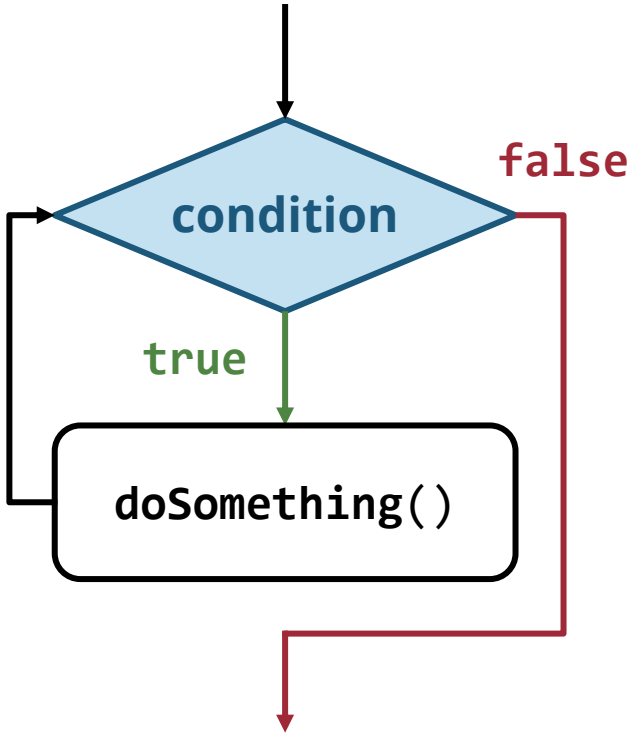
- Instructions will be released on Gradescope
- Features
  - Use the mouse to control the paddle bar
  - Show “GAME OVER!” when the paddle bar does not catch the ball
  - Click the mouse to restart the game
    - You’ll implement an `init()` function that will be called when the game starts or restarts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



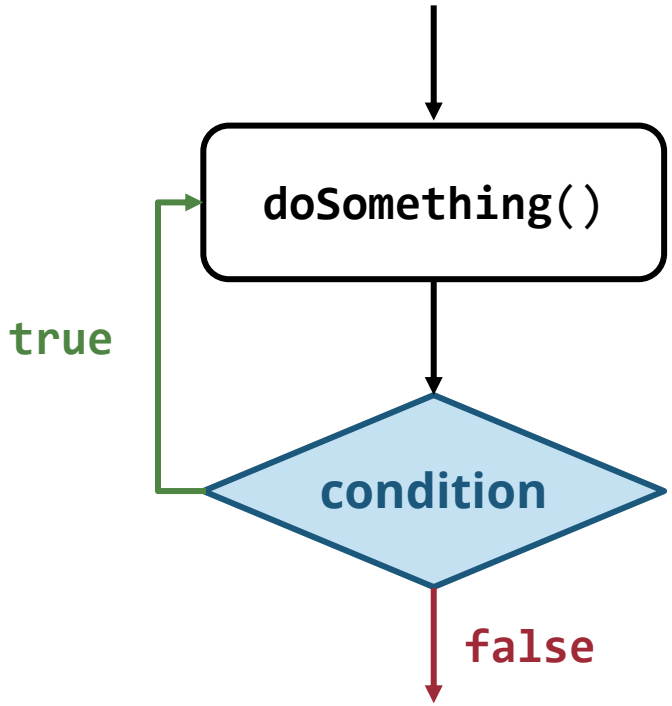
Review – Loops

while vs do-while Loops

```
while (condition) {
  doSomething();
}
```




```
do {
  doSomething();
}
while (condition);
```



break vs return vs exit()

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

void draw() {
  while (condition) {
    doSomething();

    if (condition2) {
      break;
    }
  }
  doSomethingElse();
}
```

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

void draw() {
  while (condition) {
    doSomething();

    if (condition2) {
      return;
    }
  }
  doSomethingElse();
}
```

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

void draw() {
  while (condition) {
    doSomething();

    if (condition2) {
      exit();
    }
  }
  doSomethingElse();
}
```

for Loop

```
// Initialize x
float x = 0;

// Draw the circles
while (x <= 200) {
  circle(x, 200, 10);
  x = x + 20;
}
```

Initialization

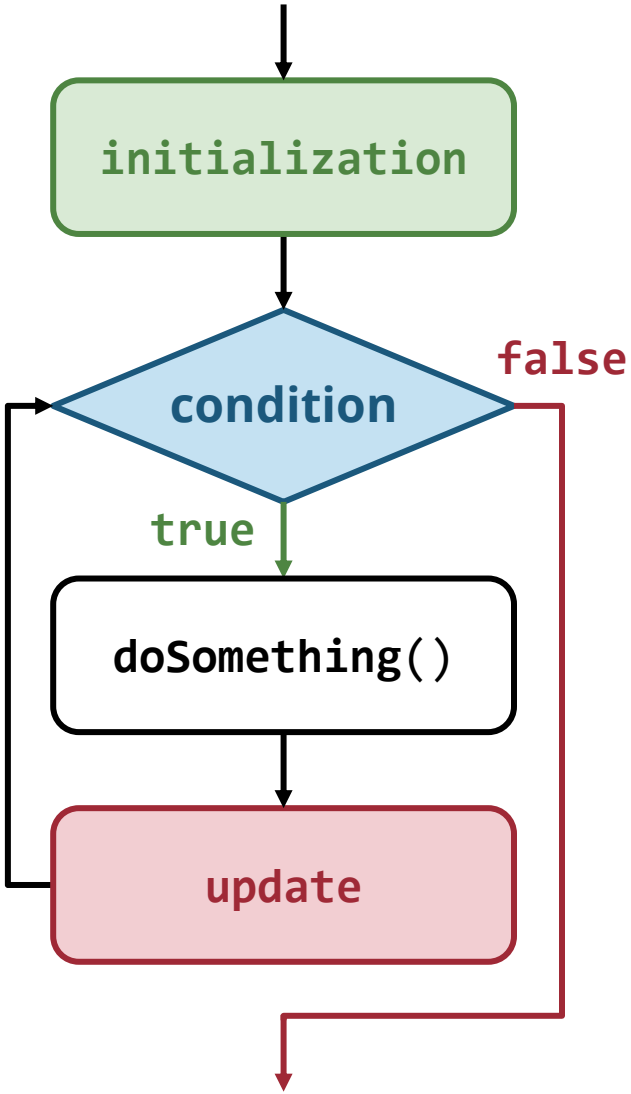
Condition

Update

```
for (float x = 0; x <= 200; x = x + 20) {
  circle(x, 200, 10);
}
```

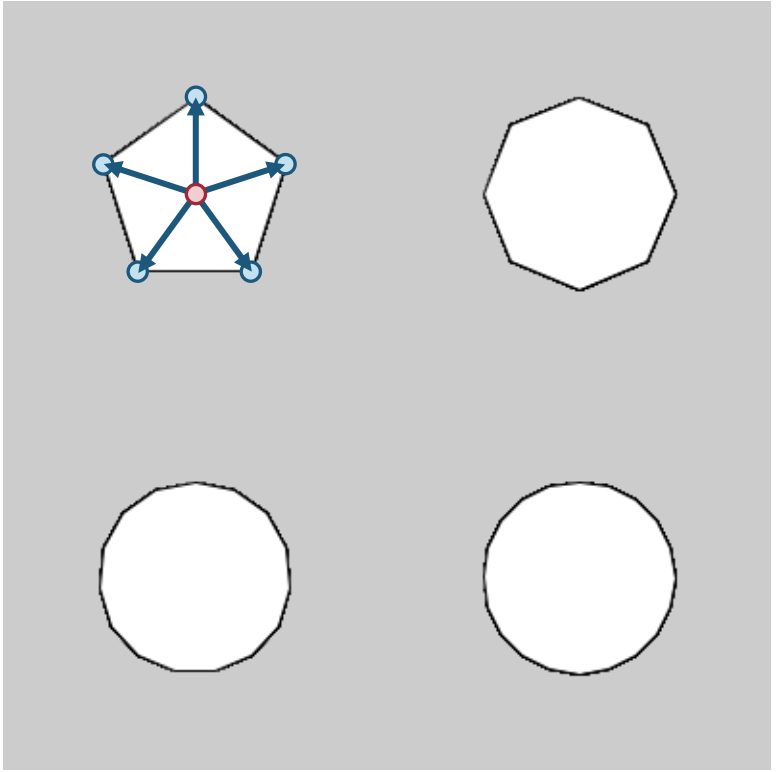
for Loop

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



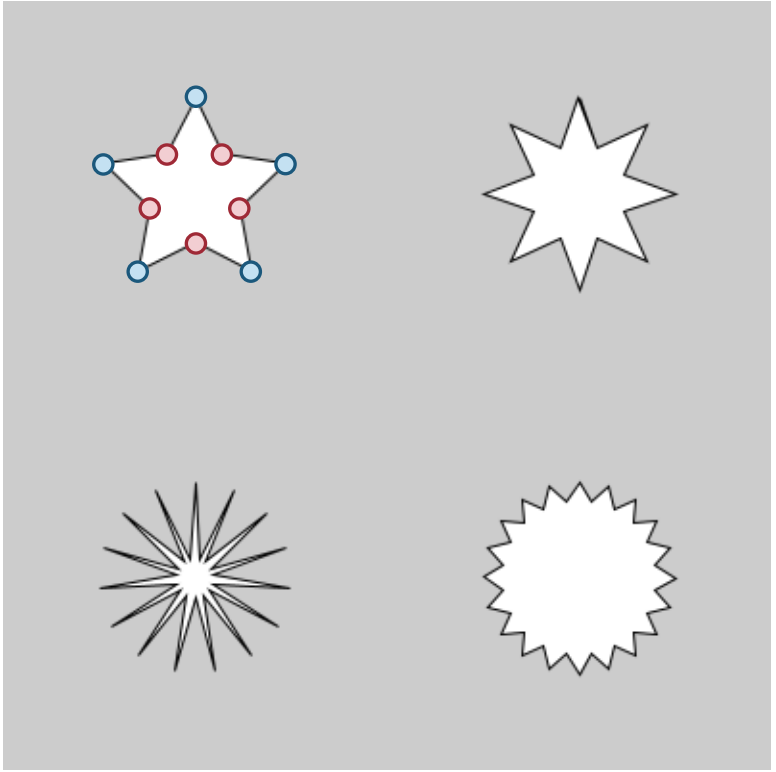
Exercise: Regular Polygons

```
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);
}
```



Example: Stars

```
void star(float x, float y, float r1, float r2, int n) {
  float vertexX, vertexY;
  float angle = TWO_PI / n ;
  beginShape();
  for (float a = 0; a < TWO_PI; a += angle) {
    vertexX = x + radius1 * cos(a - HALF_PI);
    vertexY = y + radius1 * sin(a - HALF_PI);
    vertex(vertexX, vertexY);
    vertexX = x + radius2 * cos(a + angle / 2 - HALF_PI);
    vertexY = y + radius2 * sin(a + angle / 2 - HALF_PI);
    vertex(vertexX, vertexY);
  }
  endShape(CLOSE);
}
```



For vs For-each Loop

For-each loop

```
float[] pos = {100, 200, 300};

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

void draw() {
  for (float x: pos) {
    circle(x, 200, 50);
  }
}
```

For loop

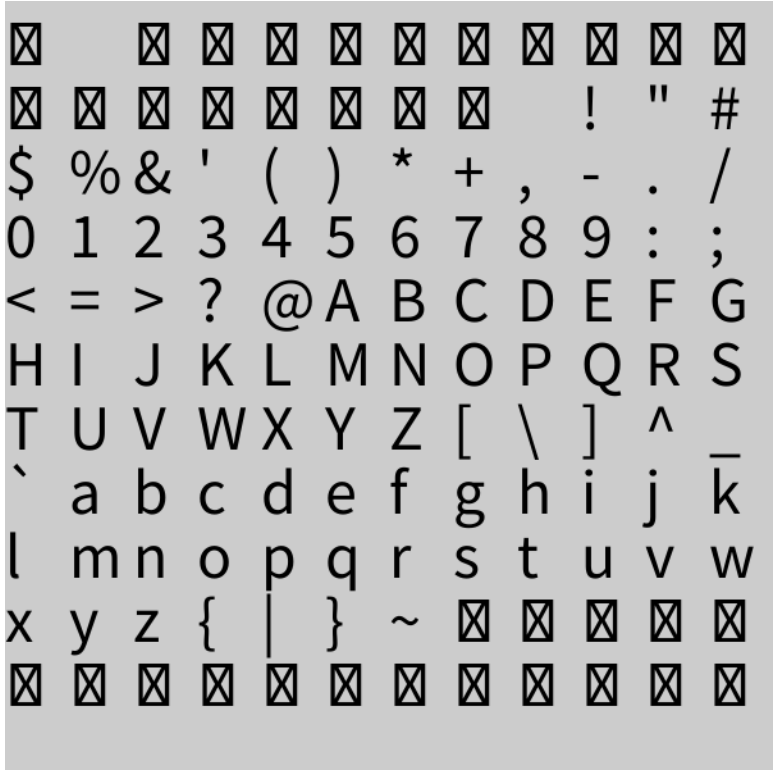
```
float[] pos = {100, 200, 300};

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

void draw() {
  for (int i = 0; i < pos.length; i++) {
    circle(pos[i], 200, 50);
  }
}
```

Exercise: Character Wall

- Print a matrix of characters using `for` loops

- **Approach 1**
  - Use a **nested `for` loop**
  - Loop over the **x index** and **y index**

- **Approach 2**
  - Use a **single `for` loop**
  - Loop over the **character code**



Two Ways of Looping

- **Approach 1**
  - Use a **nested `for` loop**
  - Loop over the **x index** and **y index**

```
for (int i = 0; i < 12; i++) {
    for (int j = 0; j < 12; j++) {
      char code = char(i + 12 * j);
      text(code, i * 50, j * 50);
    }
  }
```

- **Approach 2**
  - Use a **single `for` loop**
  - Loop over the **character code**

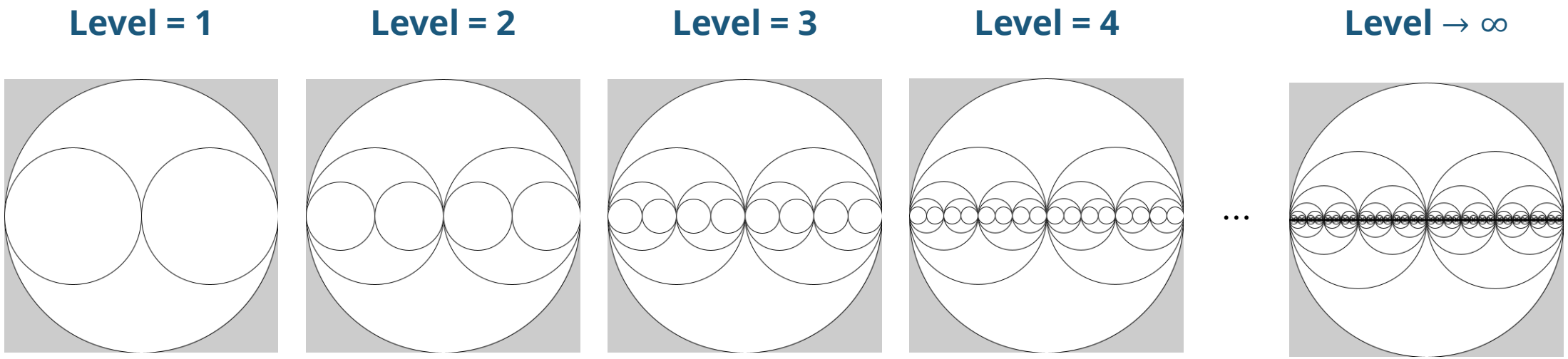
```
for (char code = 0; code < 144; code++){
    idx = int(code);
    i = idx % 12;
    j = idx / 12;
    text(code, i * 50, j * 50);
  }
```

Common way to turn a 1D sequence into a 2D matrix

Review – Recursion

Recursion

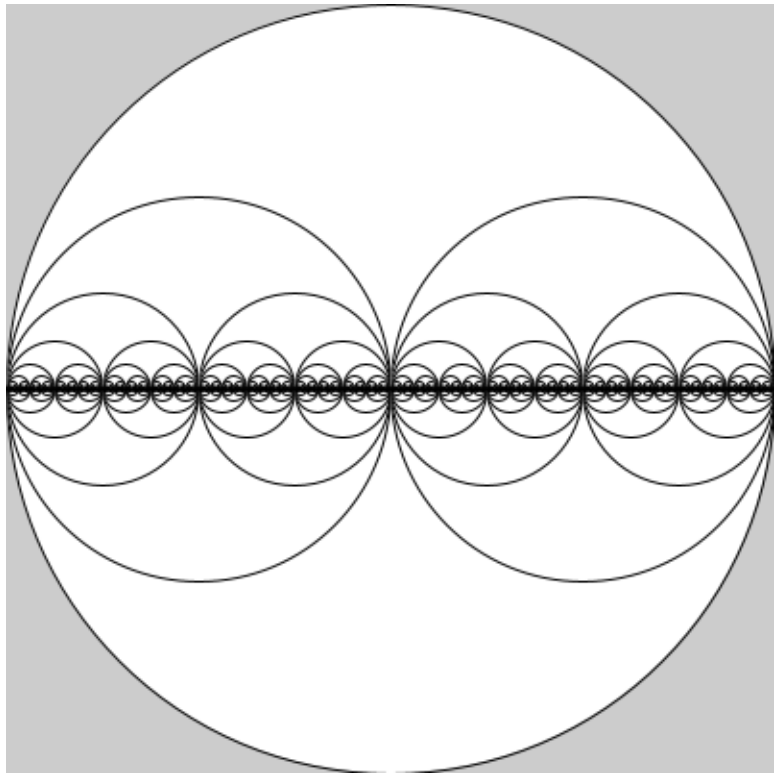
- Recursively calling a function



Example: Recursive Circles

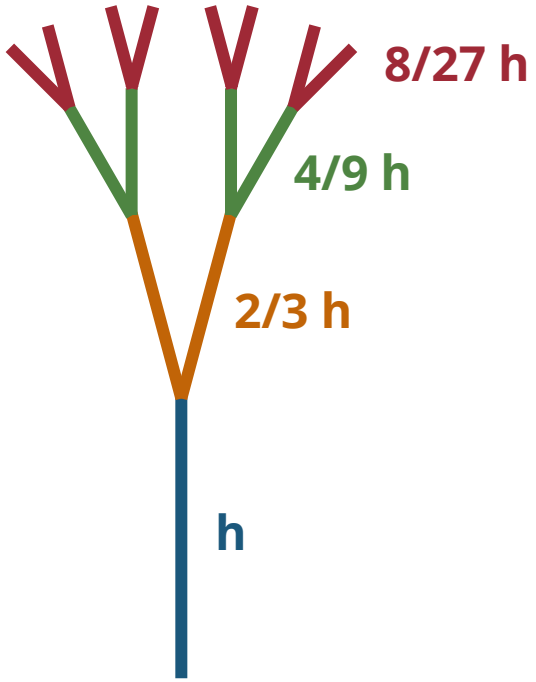
```
void drawCircles(float x, float y, float w) {
  if (w < 1) return;
  circle(x - w / 4, y, w / 2);
  drawCircles(x - w / 4, y, w / 2);

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

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

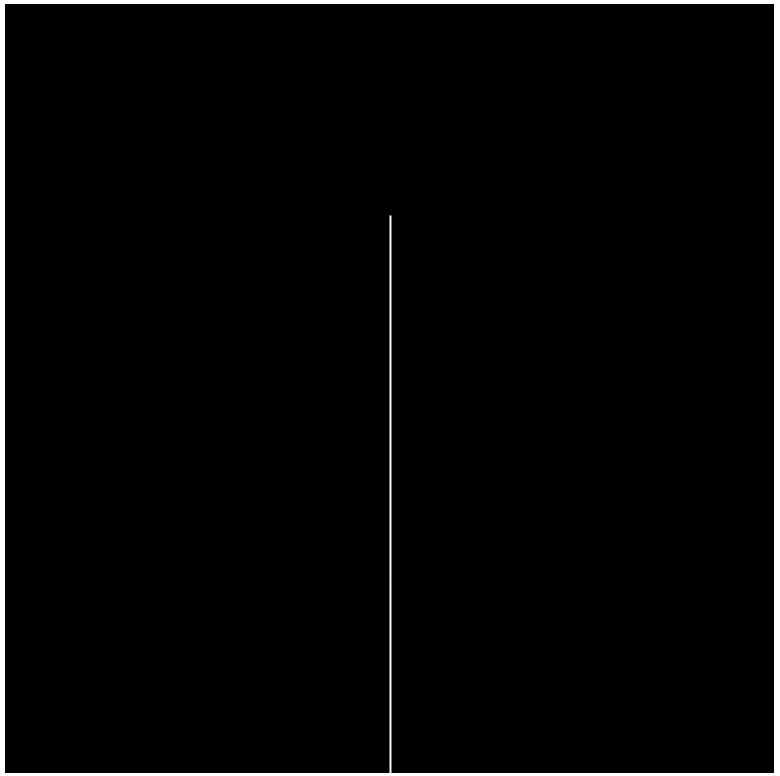
Stop condition



Example: Recursive Tree

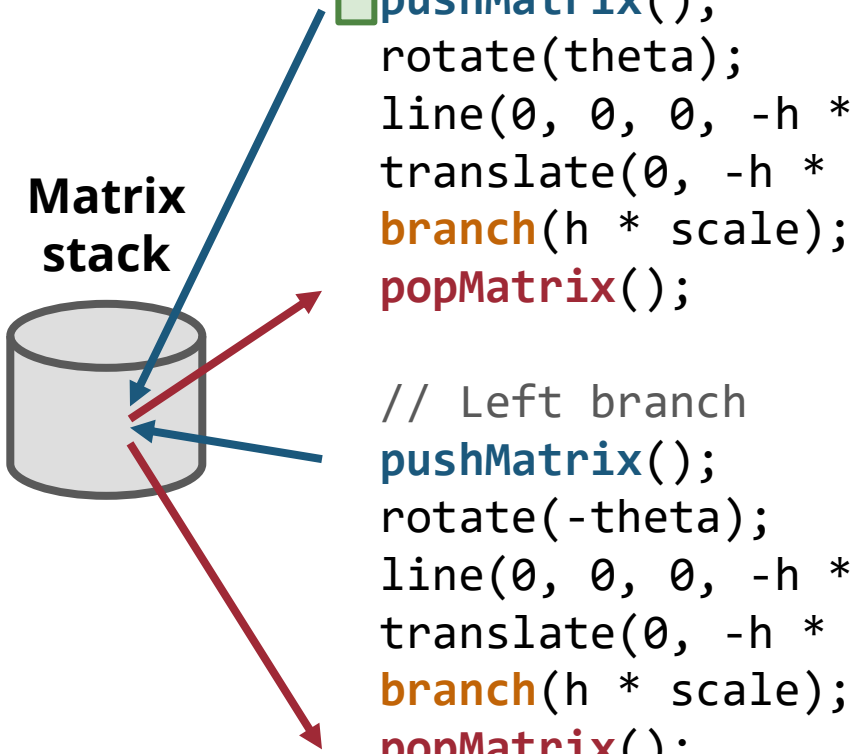
- Symmetric branches of 2/3 length of its root
  - One branch is rotated counterclockwise for a fixed angle
  - The other branch is rotated clockwise for a fixed angle

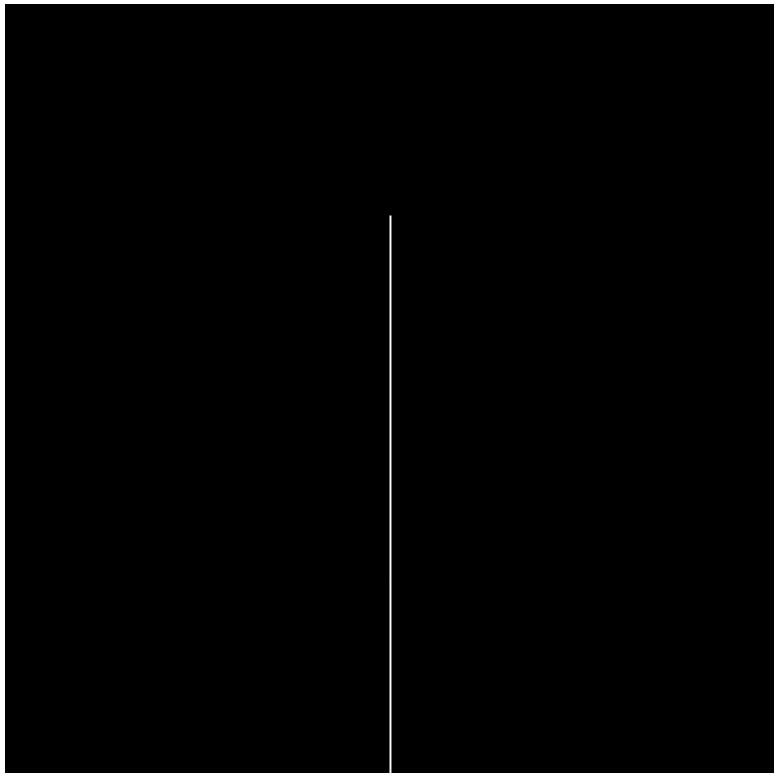




Example: Recursive Tree

```
void branch(float h) {
  if (h < 2) break;

  // Right branch
  pushMatrix();
  rotate(theta);
  line(0, 0, 0, -h * scale);
  translate(0, -h * scale);
  branch(h * scale);
  popMatrix();

  // Left branch
  pushMatrix();
  rotate(-theta);
  line(0, 0, 0, -h * scale);
  translate(0, -h * scale);
  branch(h * scale);
  popMatrix();
}
```





Review – Arrays and Lists

Array vs List

Array

```
float[] pos = new float[3];
```
Declaration

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

  pos[0] = 100;
  pos[1] = 200;
  pos[2] = 300;
}

void draw() {
  for (int i = 0; i < pos.length; i++) {
    circle(pos[i], 200, 50);
  }
}
```
Initialization

Length of the array

List

```
FloatList pos = new IntList();
```
Declaration

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

  pos.append(100);
  pos.append(200);
  pos.append(300);
}

void draw() {
  for (int i = 0; i < pos.size(); i++) {
    circle(pos.get(i), 200, 50);
  }
}
```
Initialization

Length of the list

Sorting an Array vs Sorting a List

```
int[] arr = {3, 2, 1};
sort(arr);
println(arr);
```

```
[0] 3
[1] 2
[2] 1
```

sort(arr) returns a new sorted array

```
int[] arr = {3, 2, 1};
arr = sort(arr);
println(arr);
```

```
[0] 1
[1] 2
[2] 3
```

```
IntList li = new IntList();
li.append(3);
li.append(2);
li.append(1);
li.sort();
println(li);
```

```
IntList size=3 [ 1, 2, 3 ]
```

List.sort() returns the original list, sorted

Array vs List

	Array	List
Size	Fixed	Dynamic
Item data type	Same	Can be different
Access speed	Faster	Slower
Memory requirement	Low	High
Multi-dimensional	Possible	Not supported