

PAT 204/504 (Fall 2026)

Creative Coding

Lecture 3: Keyboard Controls & Randomness

Instructor: Hao-Wen (Herman) Dong

Keyboard Controls

Built-in Keyboard Functions

Function	Called when
<code>keyPressed()</code>	a key is <i>pressed</i>
<code>keyReleased()</code>	a key is <i>released</i>
<code>keyTyped()</code>	a key is <i>clicked</i> (called repeatedly if held down)

| 🤔 Guess what this does?

```
void keyPressed() {  
    if (key == ' ') {  
        saveFrame("screenshot.png");  
    }  
}
```

| key

- `key` stores the most recent key used
 - Either pressed or *released*!

| 🤔 Guess what this does?

```
void keyPressed() {  
  if (key == CODED) {  
    if (keyCode == UP) {  
      noLoop();  
    }  
  }  
}
```

Disable looping
the draw() call

| AND Operator (&&)

```
void keyPressed() {  
  if (key == CODED) {  
    if (keyCode == UP) {  
      noLoop();  
    }  
  }  
}
```



```
void keyPressed() {  
  if (key == CODED && keyCode == UP) {  
    noLoop();  
  }  
}
```



| key & keyCode

- 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

| Controlling `draw()`

- `noLoop()` Disable looping the `draw()` call
- `loop()` Enable looping the `draw()` call
- `frameRate(fps)` Set the frame rate of looping `draw()` calls

keyCode

- How about this?

```
void keyPressed() {  
  if (key == CODED) {  
    if (keyCode == BACKSPACE) {  
      noLoop();  
    }  
  }  
}
```

BACKSPACE is *not* coded

keyCode

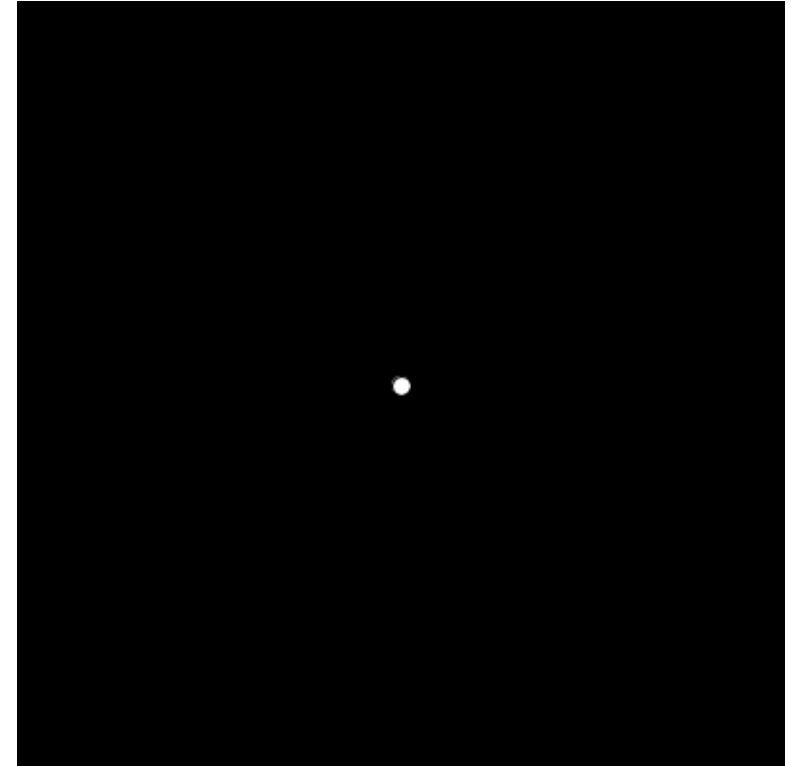
- Use simply **key** for BACKSPACE, TAB, ENTER, RETURN, ESC, and DELETE

```
void keyPressed() {  
    if (key == BACKSPACE) {  
        noLoop();  
    }  
}
```

🤔 Exercise: Use the Arrow Keys to Control a Ball

- Create a simple interface where you can move the ball around using the arrow keys
- **Hints:**
 - What **variables** do we need?
 - You'll need **keyCode** for the arrow keys

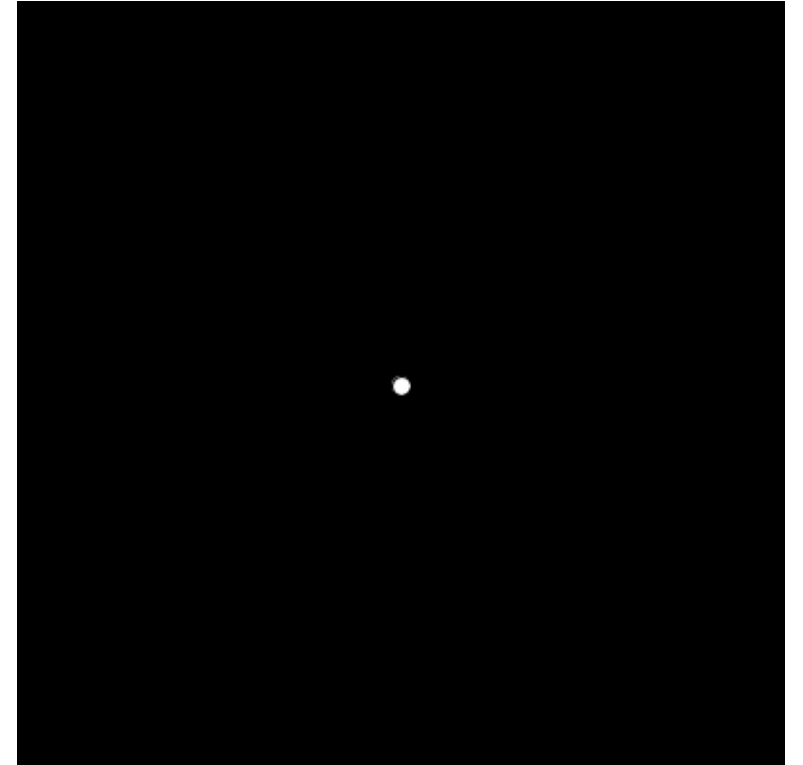
```
void keyPressed() {  
  if (key == CODED) {  
    if (keyCode == UP) {  
      doSomething();  
    }  
  }  
}
```



🤔 Exercise: Use the Arrow Keys to Control a Ball

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

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



More on Conditionals

Conditionals: if-else-if Statement

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

Conditionals: if-else-if Statement

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

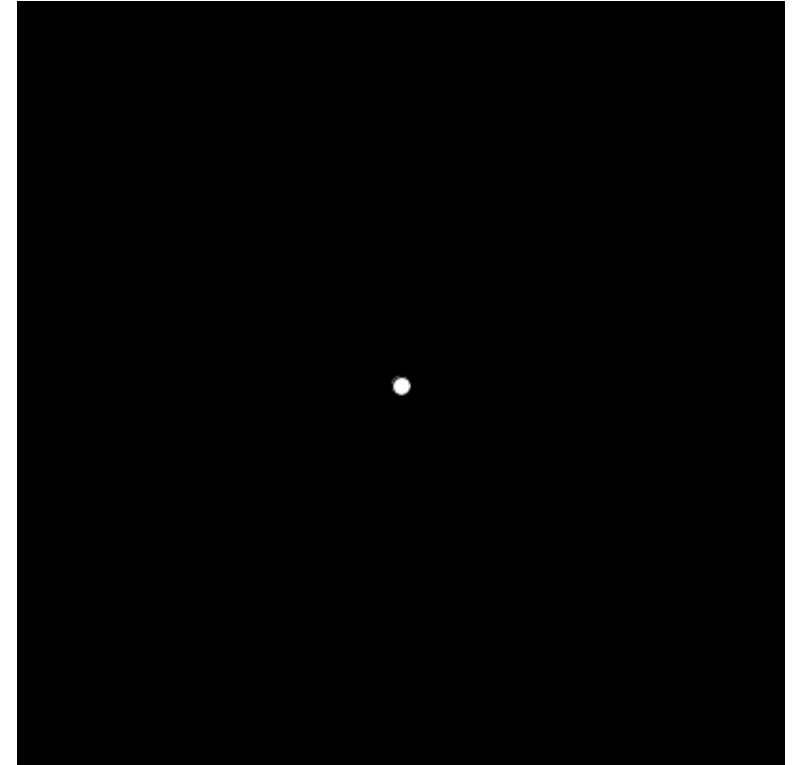


```
if (condition) {  
    doSomething();  
} else {  
    if (condition2) {  
        doSomethingElse();  
    } else {  
        doYetSomethingElse();  
    }  
}
```

🤔 Exercise: Use the Arrow Keys to Control a Ball

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

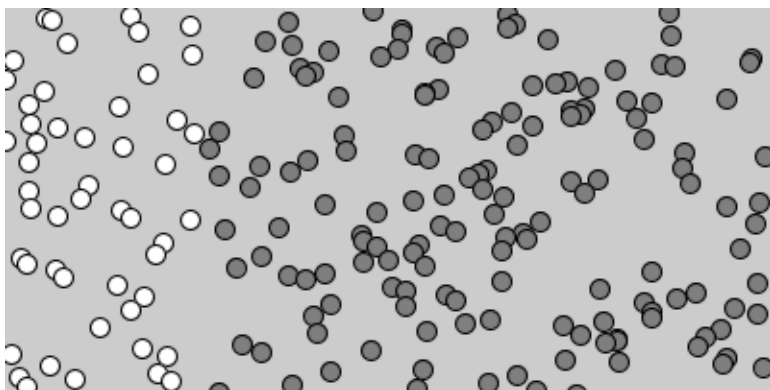
void keyPressed() {
  if (key == CODED) {
    if (keyCode == LEFT) {
      x = x - step;
    } else if (keyCode == RIGHT) {
      x = x + step;
    } else if (keyCode == UP) {
      y = y - step;
    } else if (keyCode == DOWN) {
      y = y + step;
    }
  }
}
```



else if vs. if

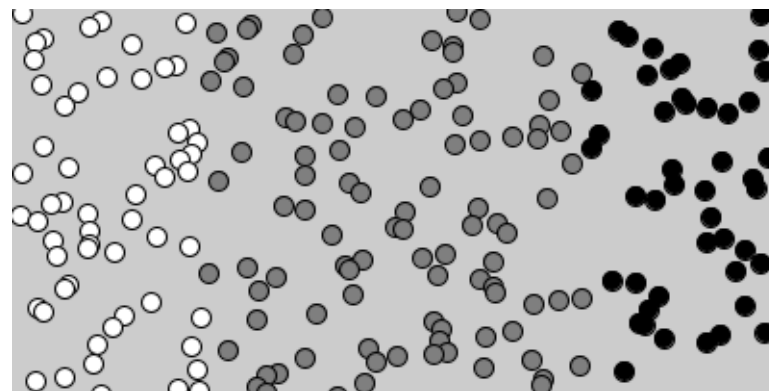
```
if (mouseX > 300) {  
  fill(0);  
}  
if (mouseX < 100) {  
  fill(255);  
} else {  
  fill(127);  
}
```

mouseX ≥ 100



```
if (mouseX > 300) {  
  fill(0);  
} else if (mouseX < 100) {  
  fill(255);  
} else {  
  fill(127);  
}
```

100 ≤ mouseX ≤ 300



| switch Statement

```
switch(expression) {  
  case value1:  
    doSomething();  
    break;  
  
  case value2:  
    doSomethingElse();  
    break;  
  
  case value3:  
    doYetSomethingElse();  
    break;  
}
```

🤔 Exercise: Rewrite it using `switch`

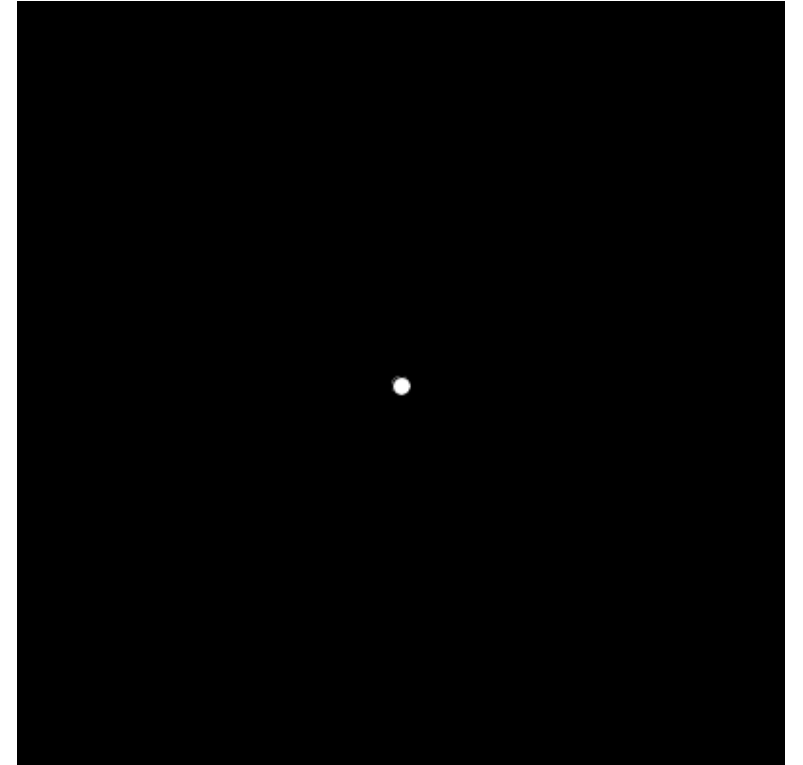
```
float x = 200;
float y = 200;
float step = 10;

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;
    }
  }
}
```

```
switch(expression) {
  case value1:
    doSomething();
    break;

  case value2:
    doSomethingElse();
    break;

  case value3:
    doYetSomethingElse();
    break;
}
```



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;  
}
```

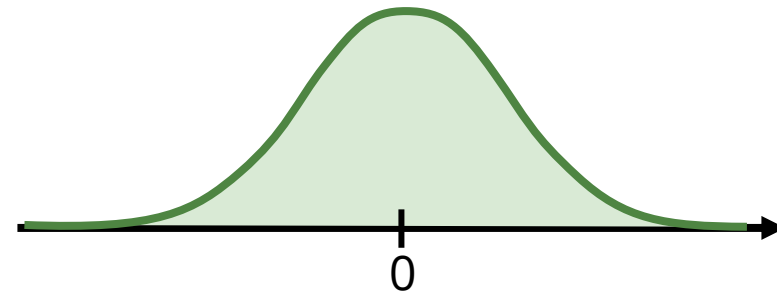
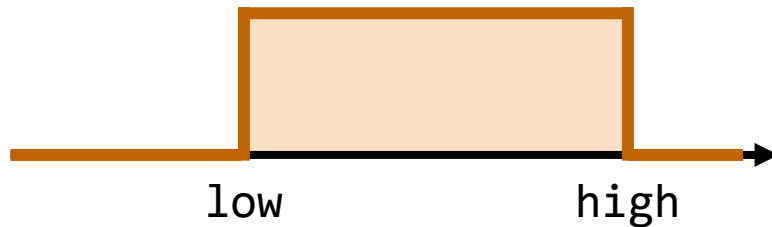
| Combining Multiple Conditions in switch

```
switch(expression) {  
    case value1:  
    case value2:  
    case value3:  
        doSomethingElse();  
        break;  
  
    case value4:  
        doYetSomethingElse();  
        break;  
}
```

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 $N[0, 1]$



Generating Random Numbers with random()

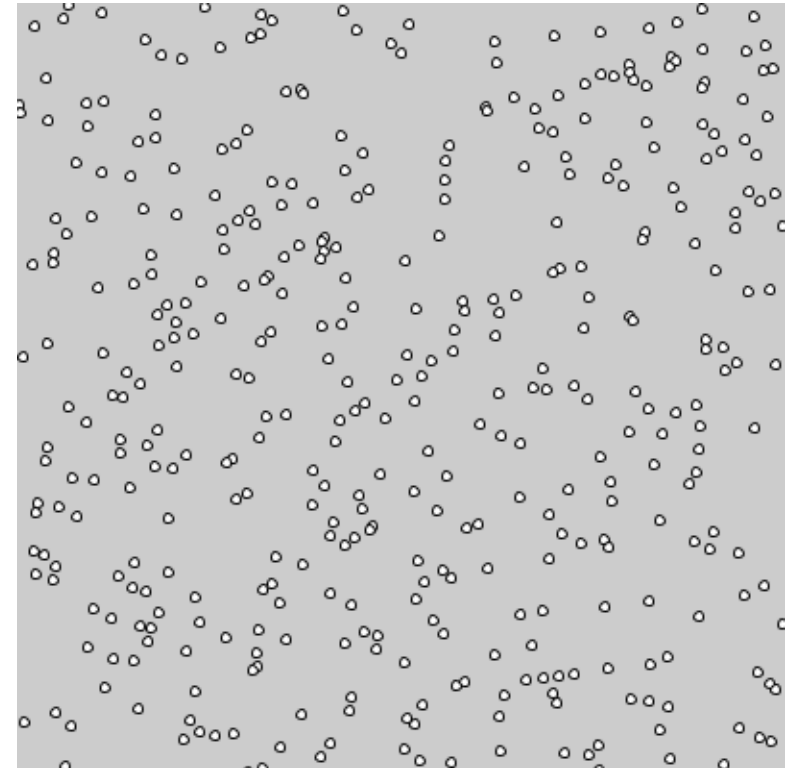
```
float x, y;

void setup() {
  // Create a 400x400 canvas
  size(400, 400);
}

void draw() {
  // Generate a random number
  x = random(400);

  // Gradually increase the y-position
  y = y + 1;

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



Generating Random Numbers with random()

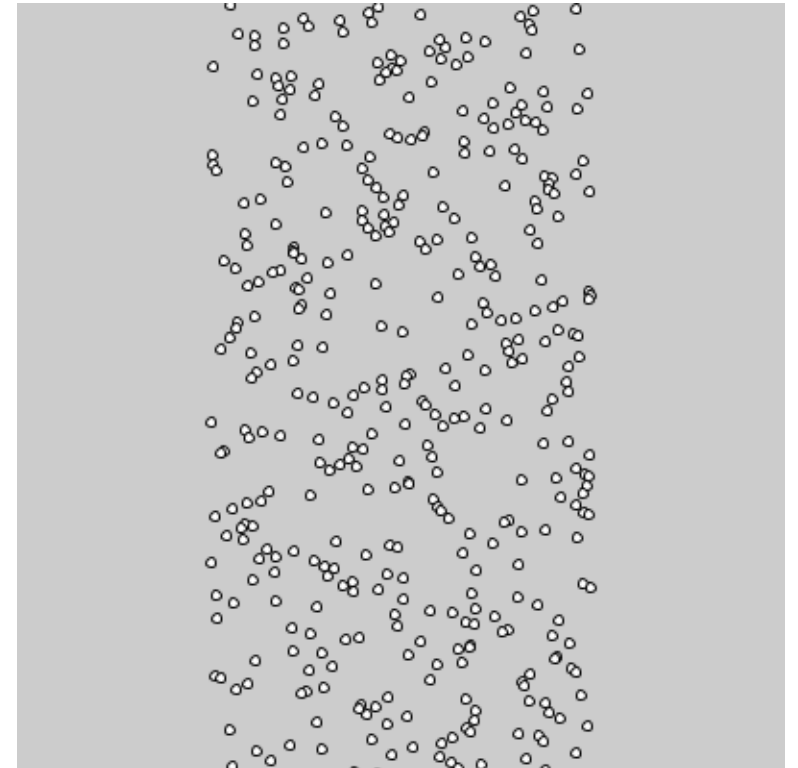
```
float x, y;

void setup() {
  // Create a 400x400 canvas
  size(400, 400);
}

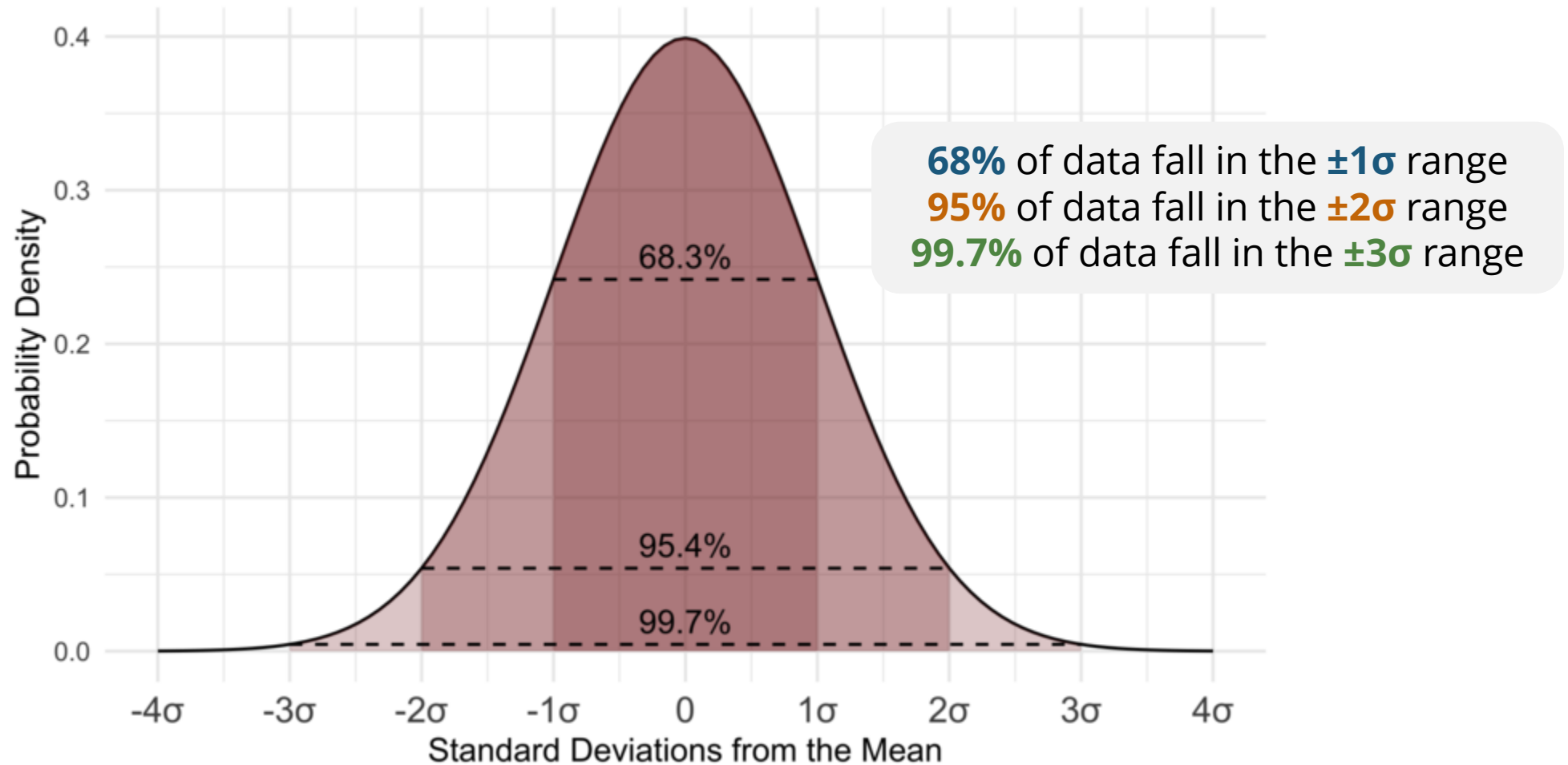
void draw() {
  // Generate a random number
  x = random(100, 300);

  // Gradually increase the y-position
  y = y + 1;

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



Gaussian Distribution & the 68–95–99.7 Rule



(Source: D Wells)

Random Numbers with `randomGaussian()`

```
float x, y;

void setup() {
  // Create a 400x400 canvas
  size(400, 400);
}

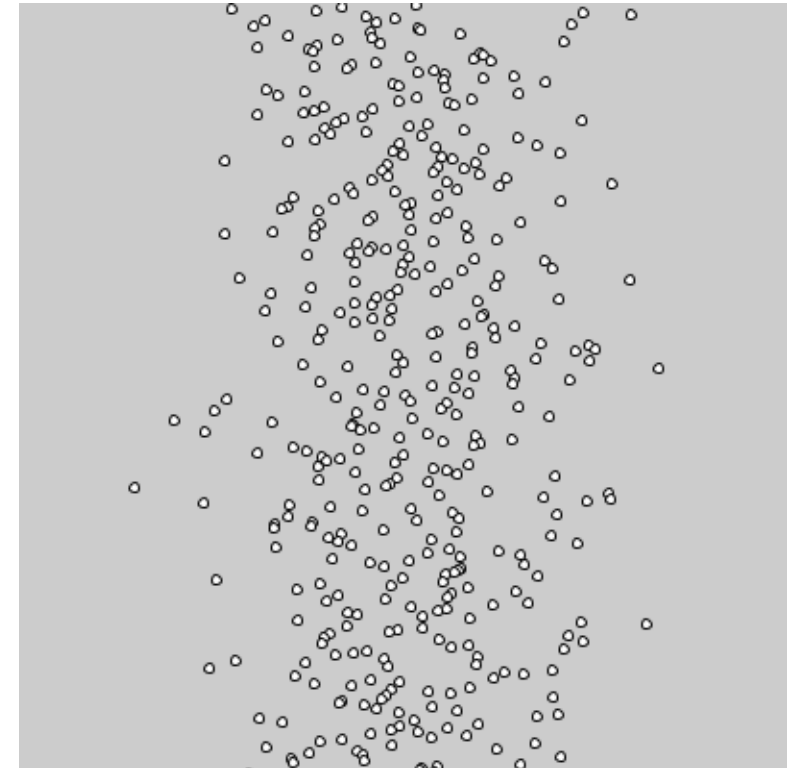
void draw() {
  // Generate a random number
  x = randomGaussian() * 50 + 200;

  // Gradually increase the y-position
  y = y + 1;

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

$\sigma = 50$

Centered at 200

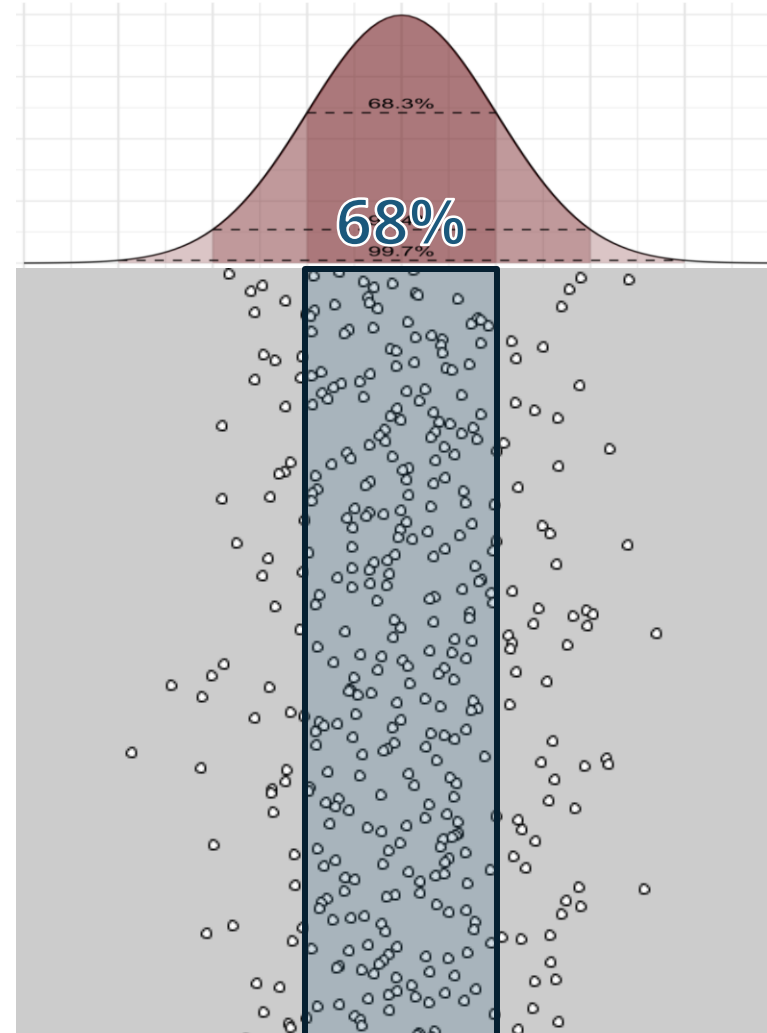


| The 68–95–99.7 Rule

```
x = randomGaussian() * 50 + 200;
```

Standard
deviation

Mean

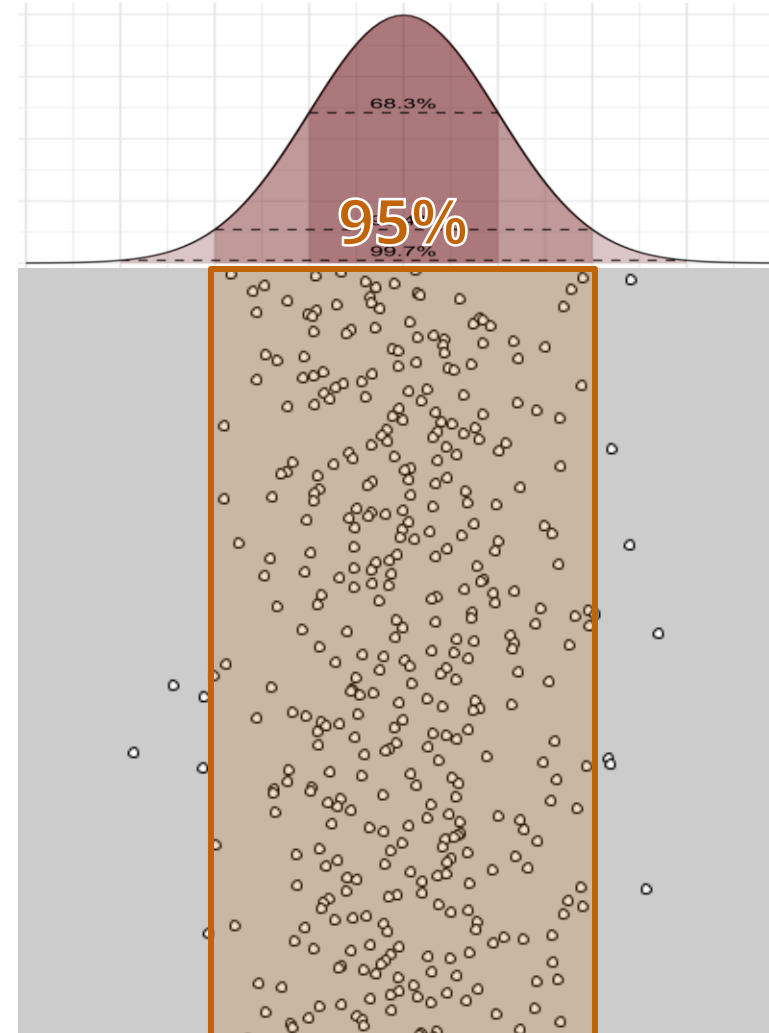


| The 68–95–99.7 Rule

```
x = randomGaussian() * 50 + 200;
```

Standard
deviation

Mean

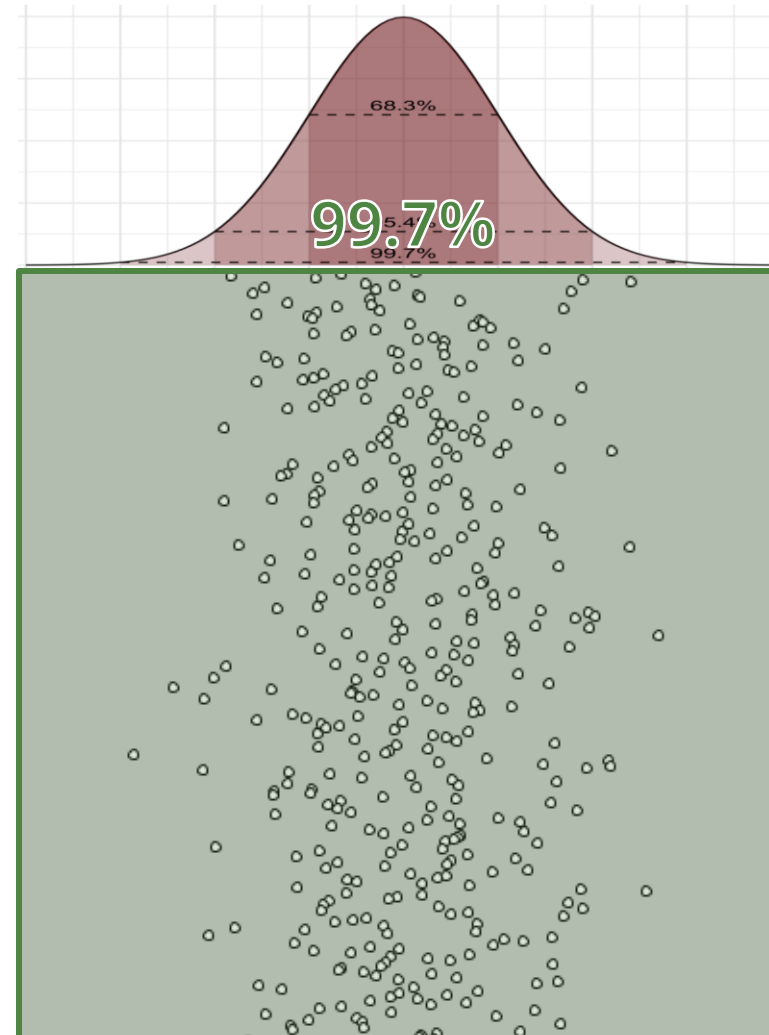


| The 68–95–99.7 Rule

```
x = randomGaussian() * 50 + 200;
```

Standard
deviation

Mean



Random Numbers with `randomGaussian()`

```
float x, y;

void setup() {
  // Create a 400x400 canvas
  size(400, 400);
}

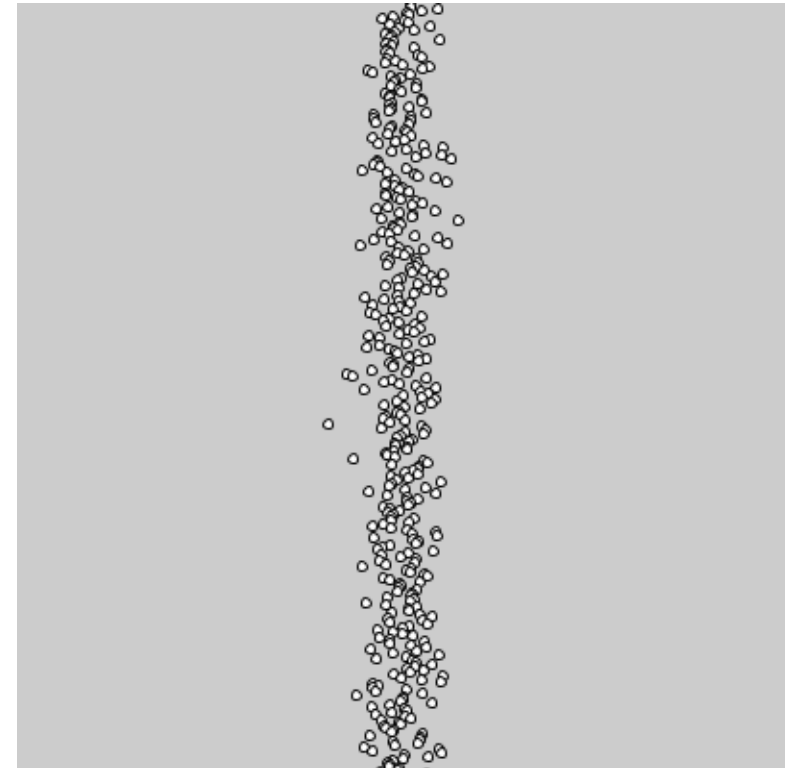
void draw() {
  // Generate a random number
  x = randomGaussian() * 10 + 200;

  // Gradually increase the y-position
  y = y + 1;

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

$\sigma = 10$

Centered at 200

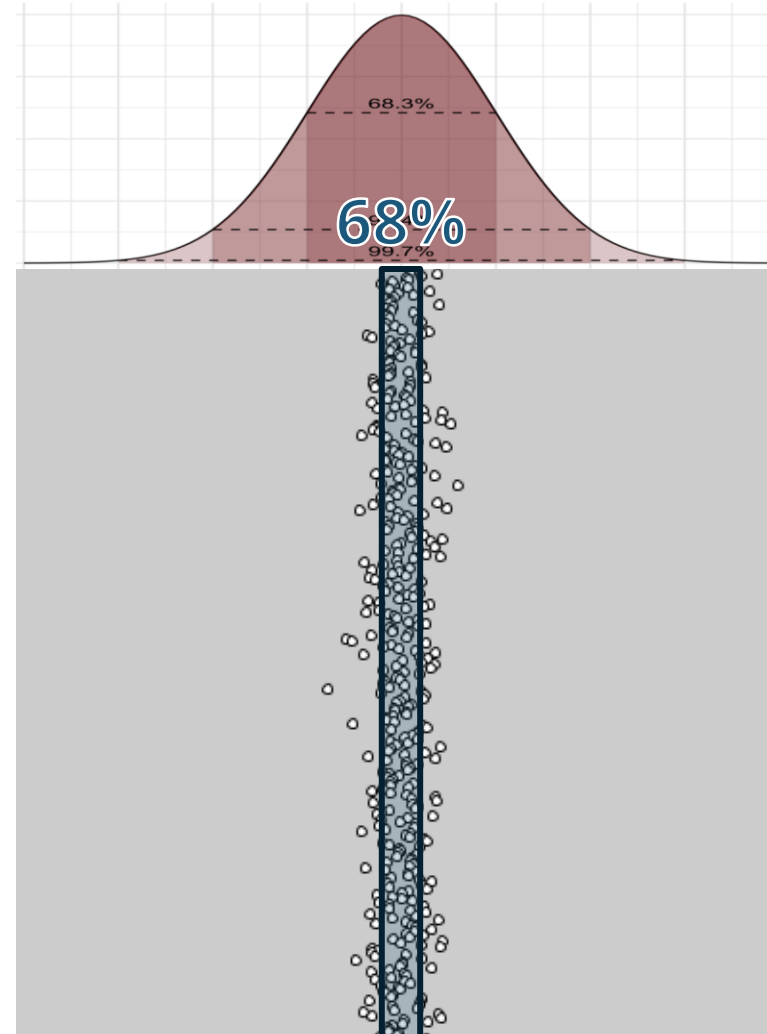


| The 68–95–99.7 Rule

```
x = randomGaussian() * 10 + 200;
```

Standard
deviation

Mean

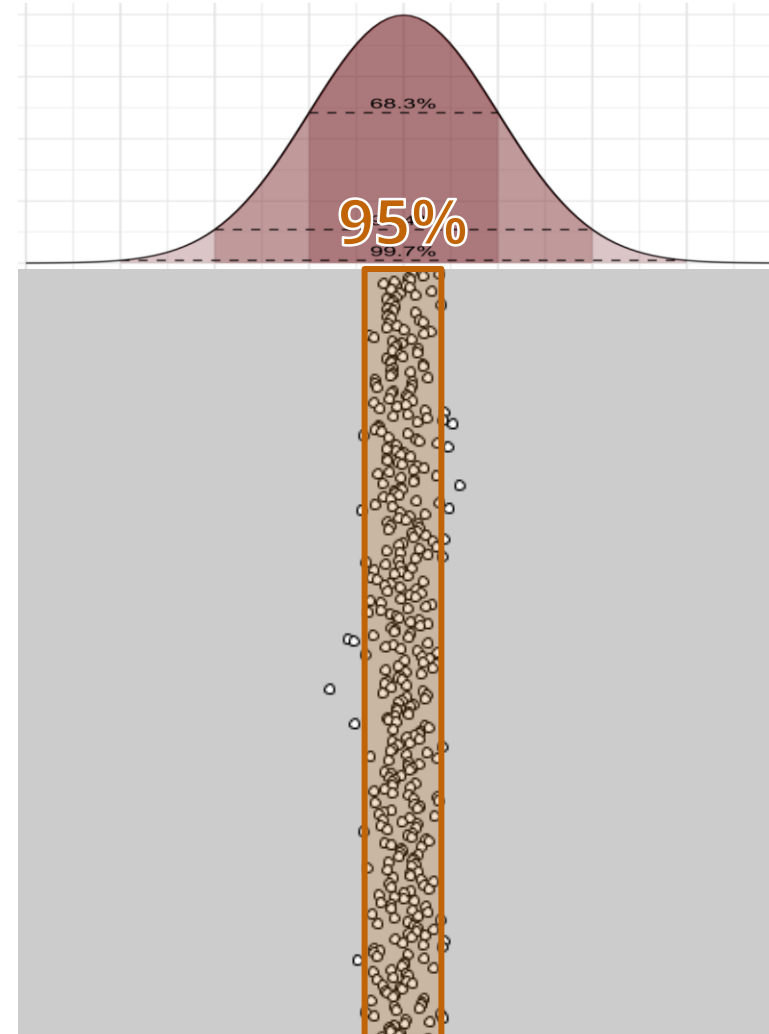


| The 68–95–99.7 Rule

```
x = randomGaussian() * 10 + 200;
```

Standard
deviation

Mean

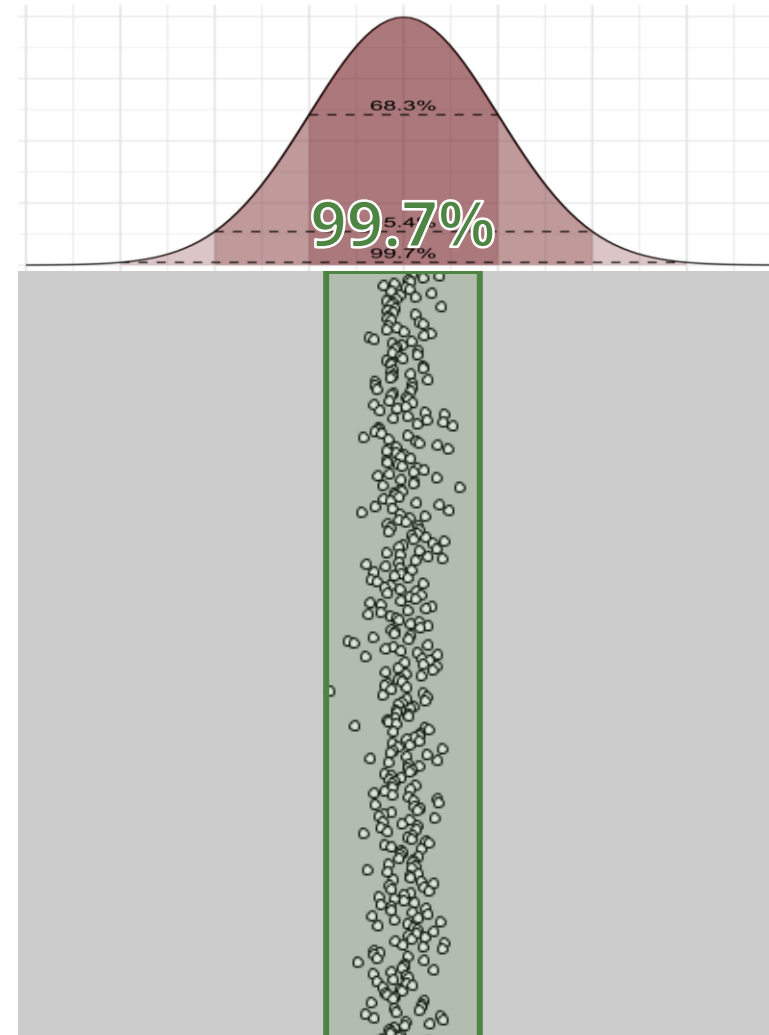


| The 68–95–99.7 Rule

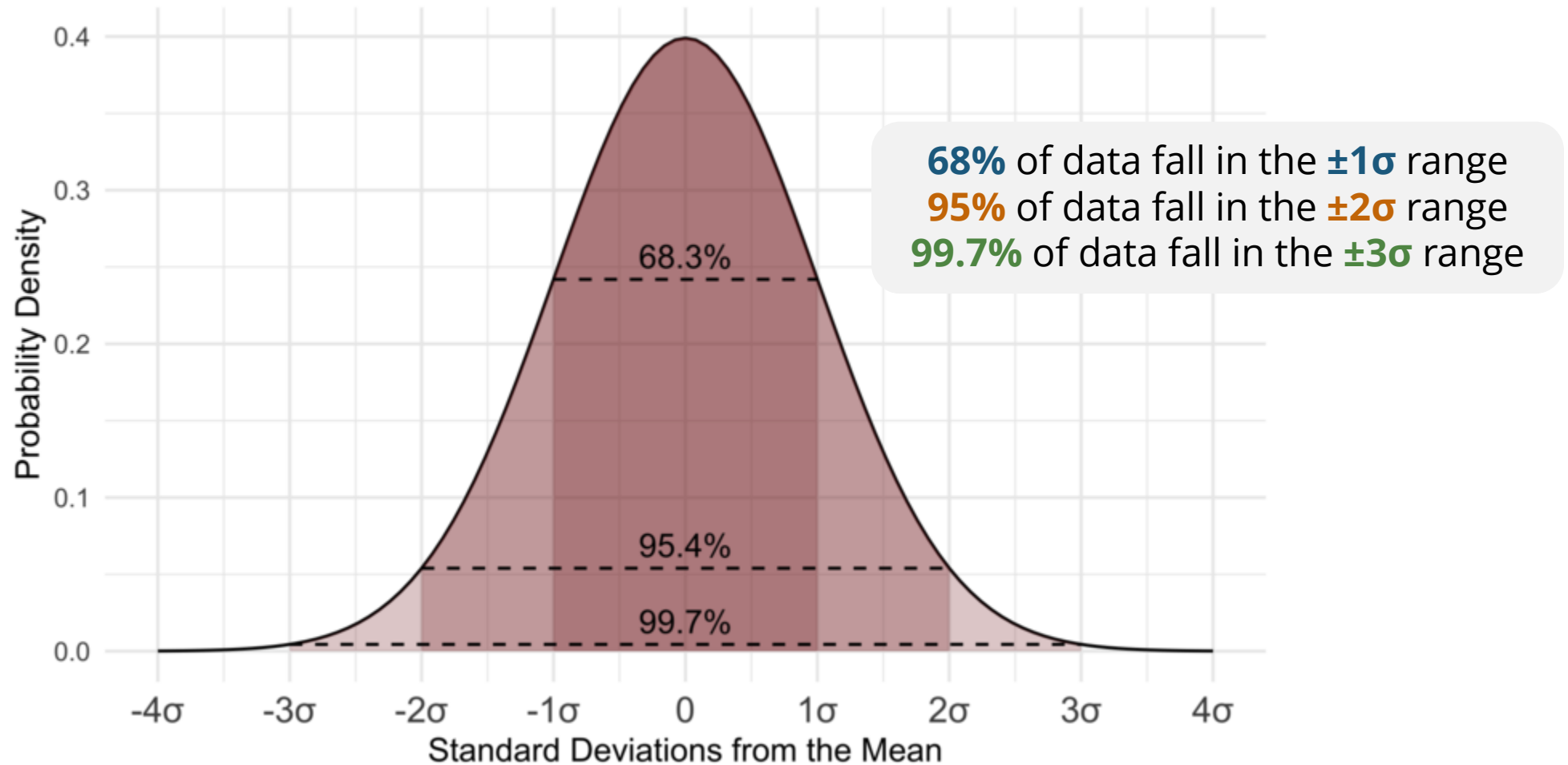
```
x = randomGaussian() * 10 + 200;
```

Standard
deviation

Mean



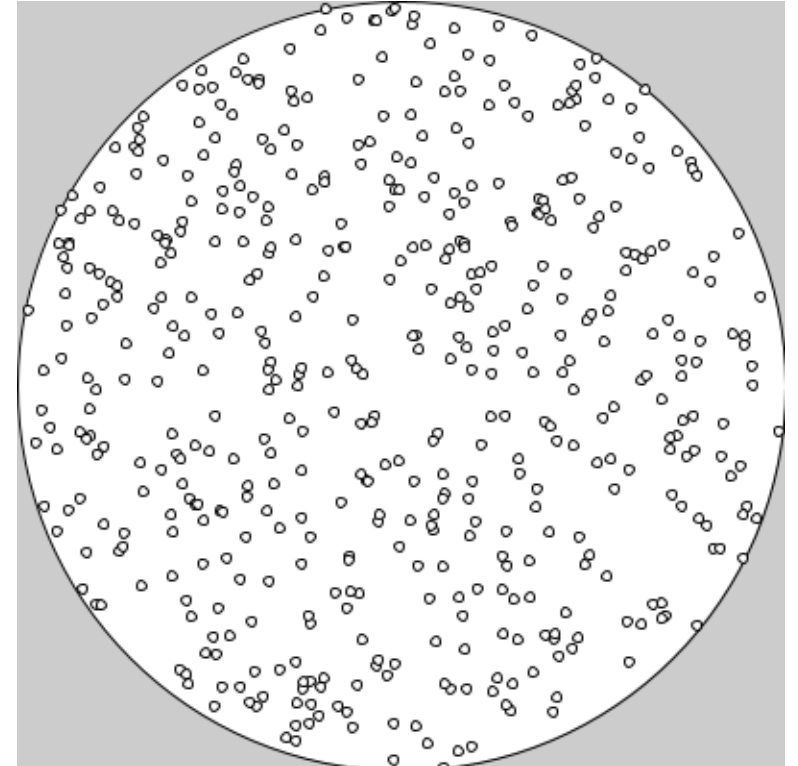
Gaussian Distribution & the 68–95–99.7 Rule



(Source: D Wells)

🤔 Exercise: Random Points in a Circle

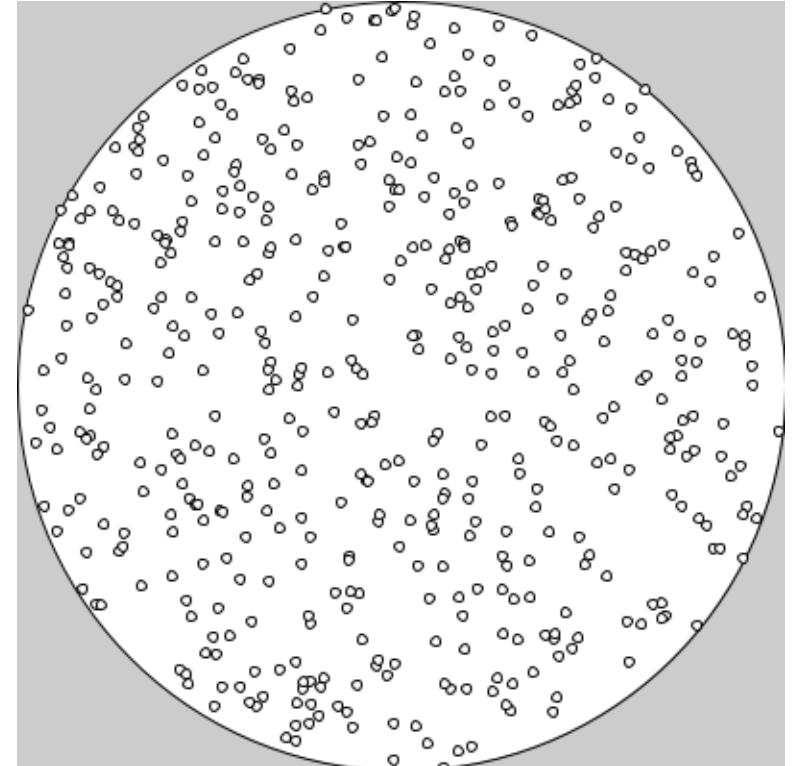
- How to randomly **sample a point in a circle**?
- **Strategy 1: Rejection Sampling**
 - Sample randomly and **keep only those fall in the circle**



🤔 Exercise: Rejection Sampling

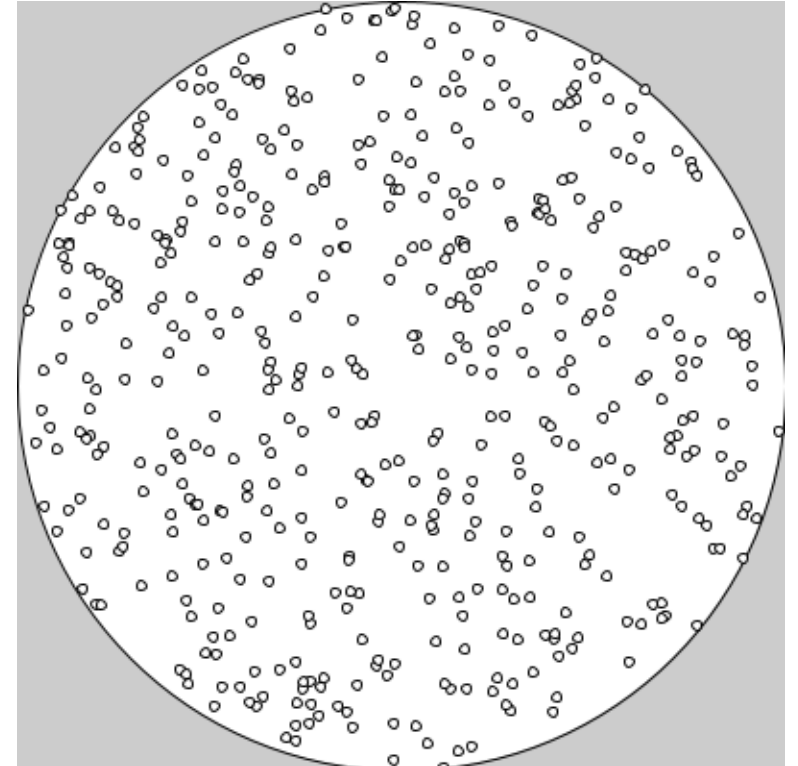
```
// Calculate the x- and y-positions
x = random(400);
y = random(400);

// Check if the point lies in the circle
if (dist(200, 200, x, y) < 200) {
  // Draw a circle
  circle(x, y, 5);
}
```

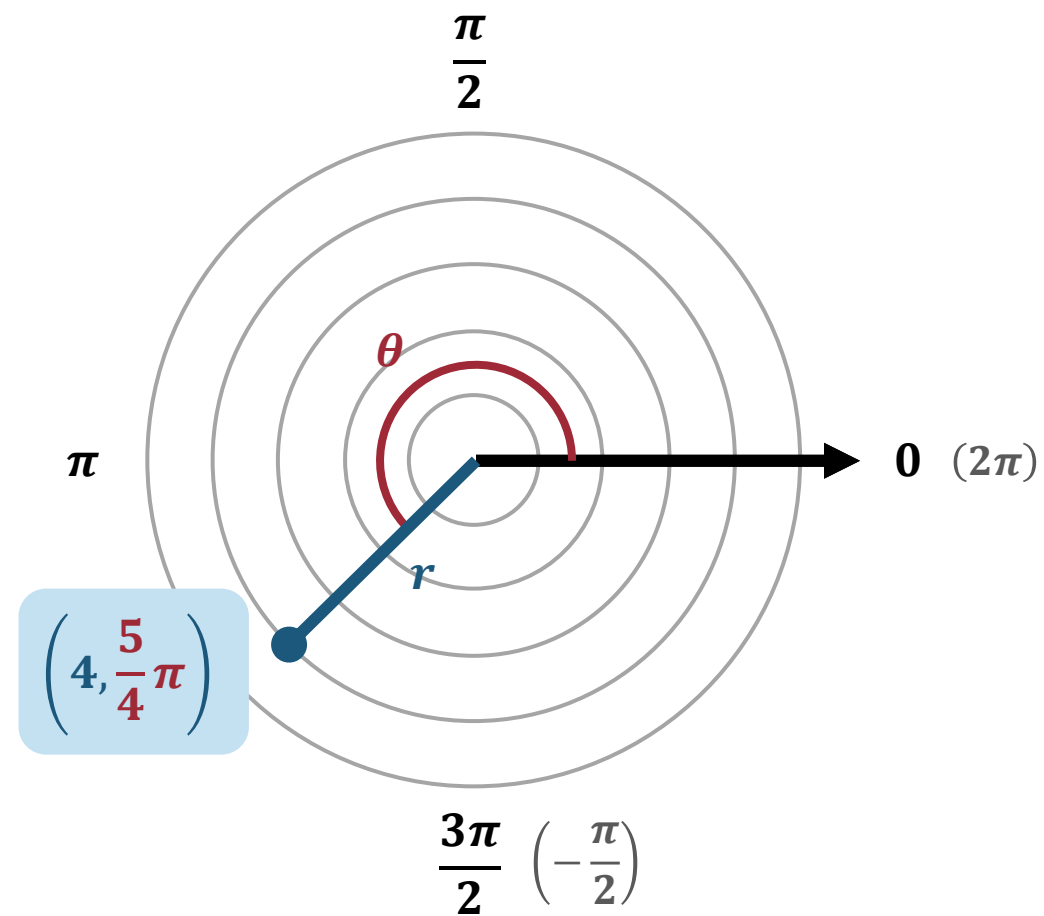
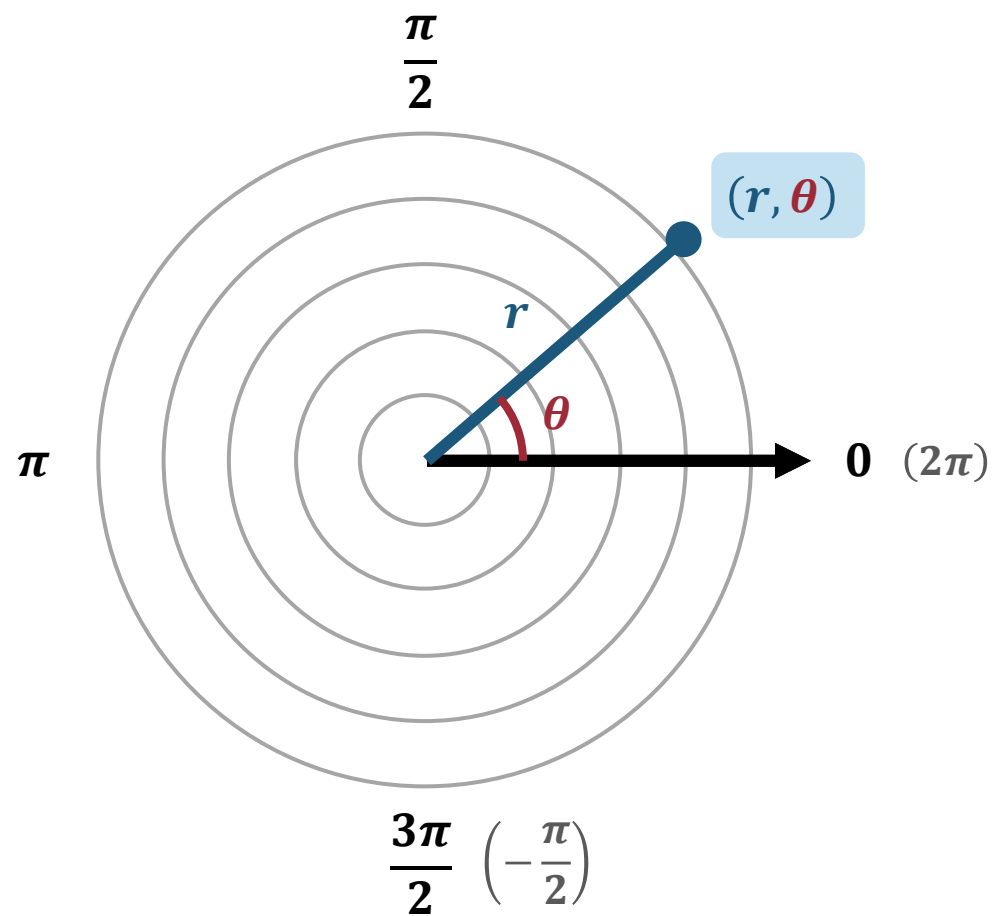


🤔 Exercise: Random Points in a Circle

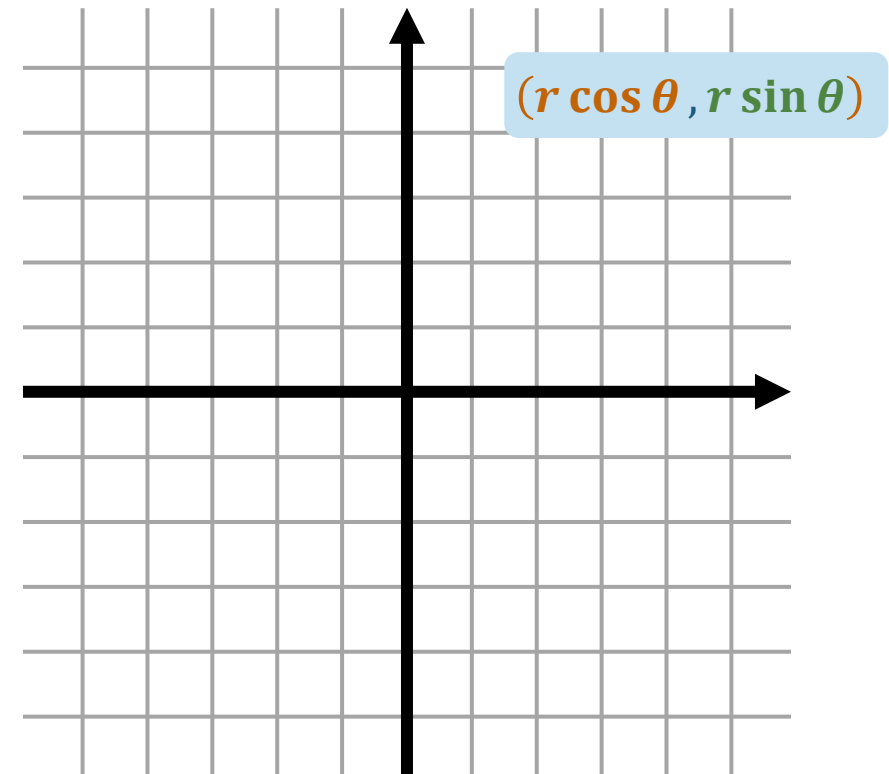
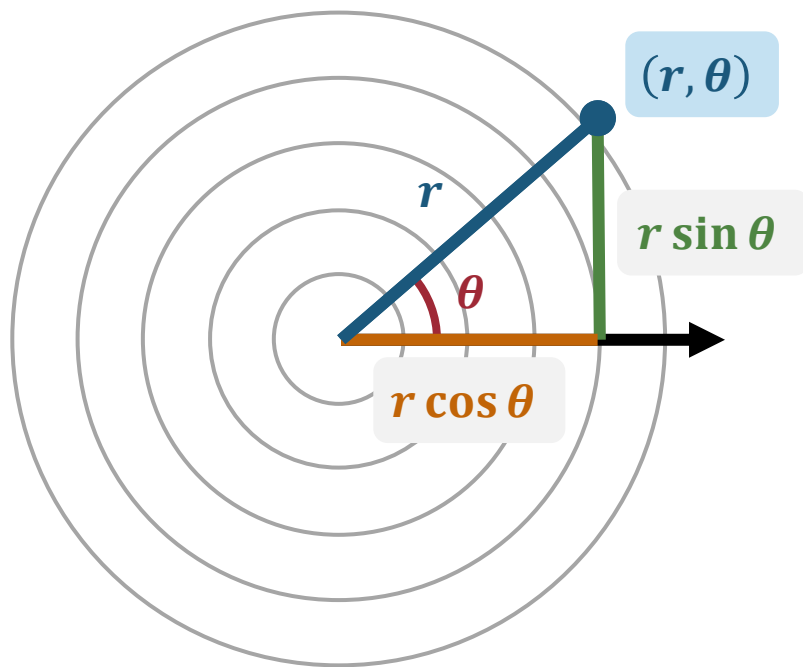
- How to randomly **sample a point in a circle**?
- Strategy 1: Rejection Sampling
 - Sample randomly and keep only those fall in the circle
- **Strategy 2: Polar Coordinate Sampling**
 - Sample points that **always fall in the circle**



Polar Coordinate



Polar-to-Cartesian Conversion



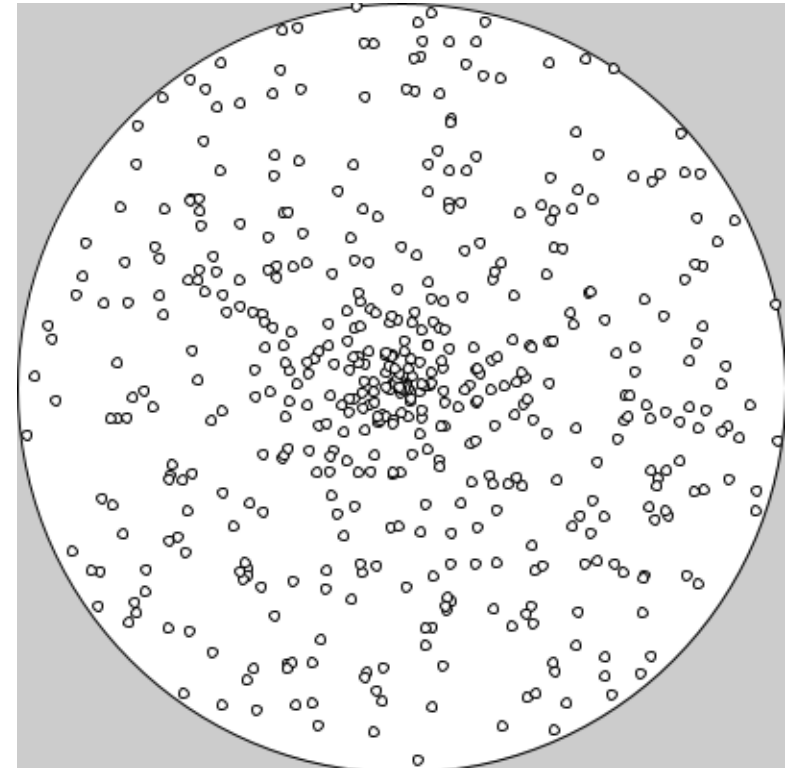
$$(r, \theta) \rightarrow (r \cos \theta, r \sin \theta)$$

🤔 Exercise: Polar Coordinate Sampling

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

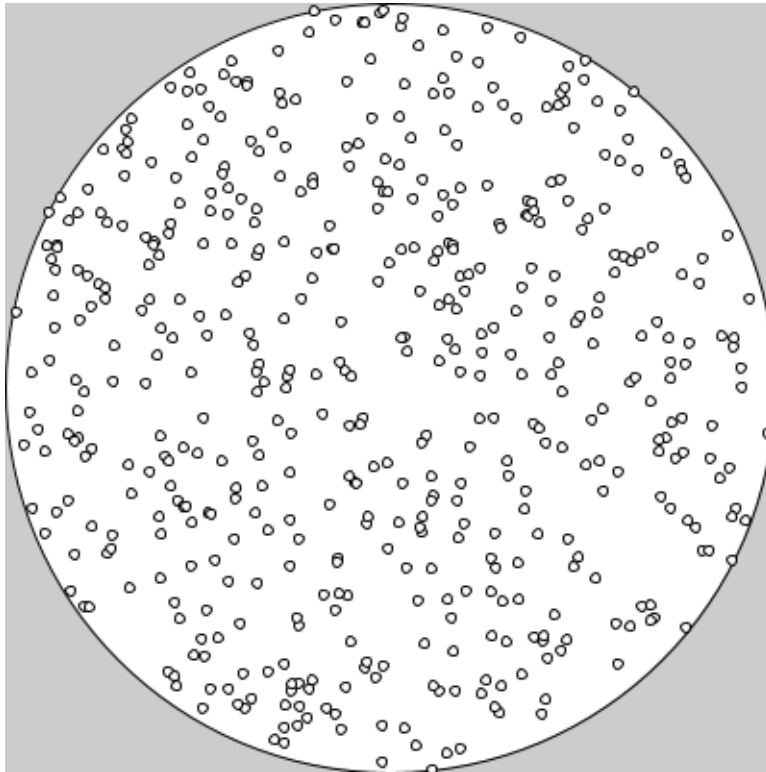
// Calculate the x- and y-positions
x = 200 + r * cos(theta);
y = 200 + r * sin(theta);

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

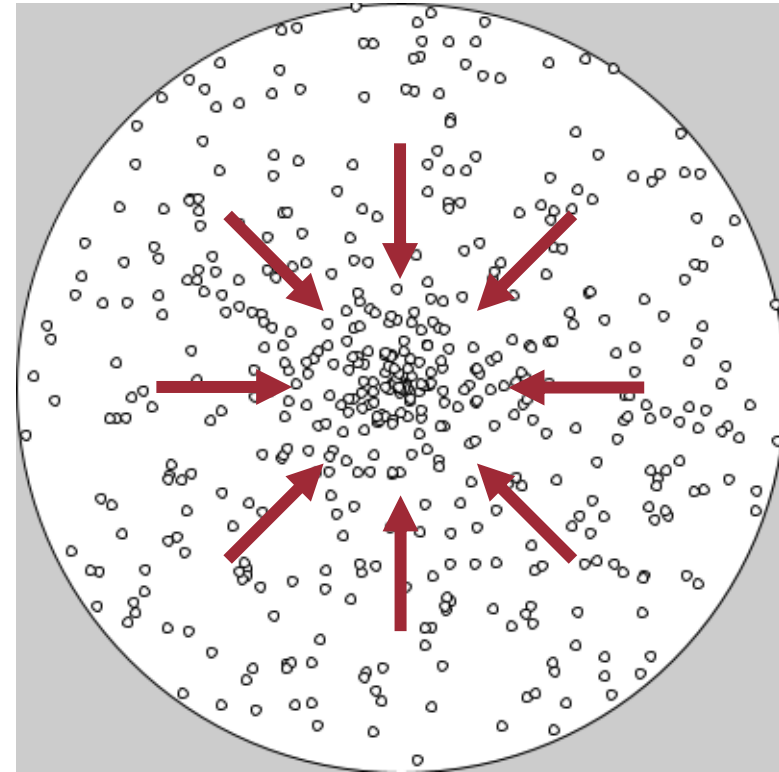


Polar Coordinate Sampling vs. Rejection Sampling

Rejection Sampling



Polar Coordinate Sampling

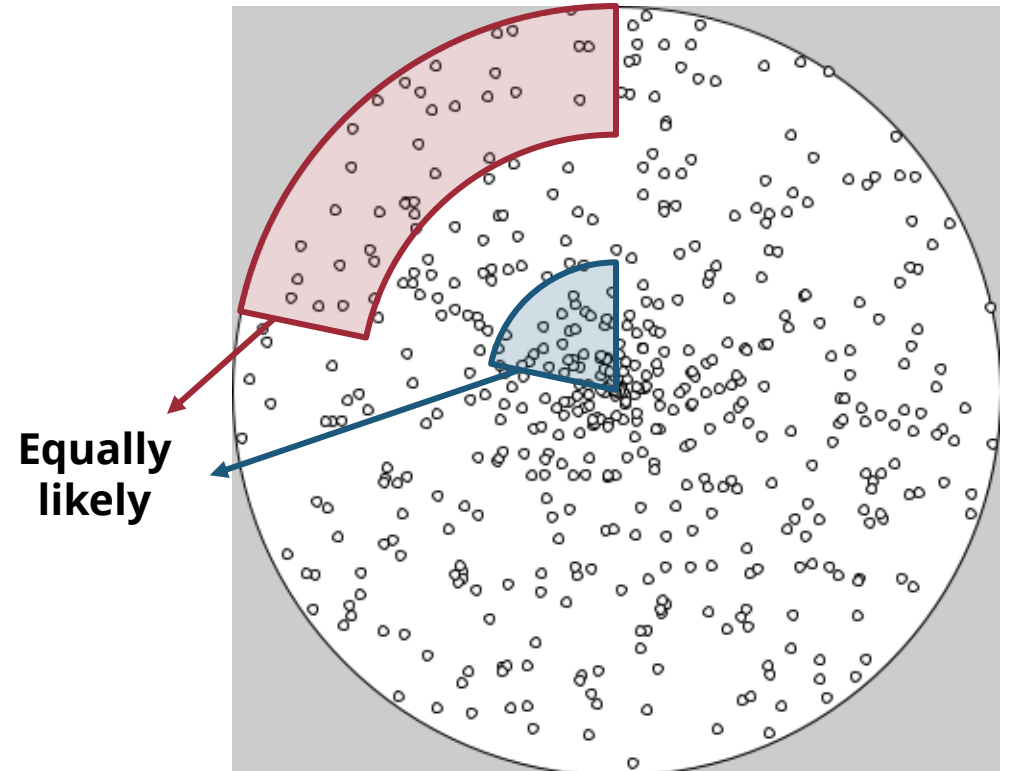


🤔 Exercise: Polar Coordinate Sampling

```
// 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);
```

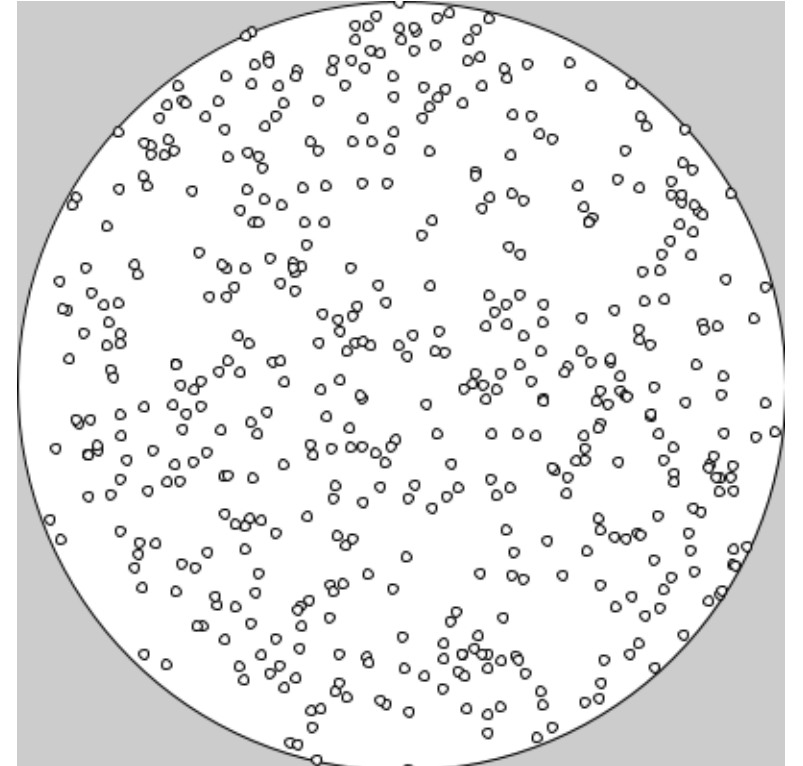


🤔 Exercise: Inversion Transform Sampling

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

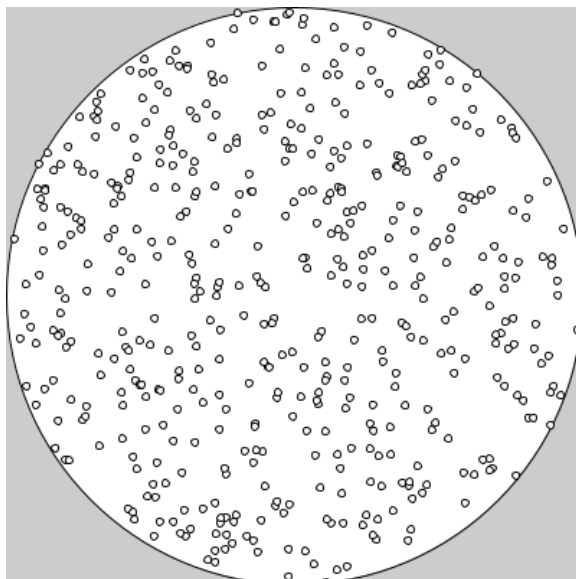
// Calculate the x- and y-positions
x = 200 + r * cos(theta);
y = 200 + r * sin(theta);

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



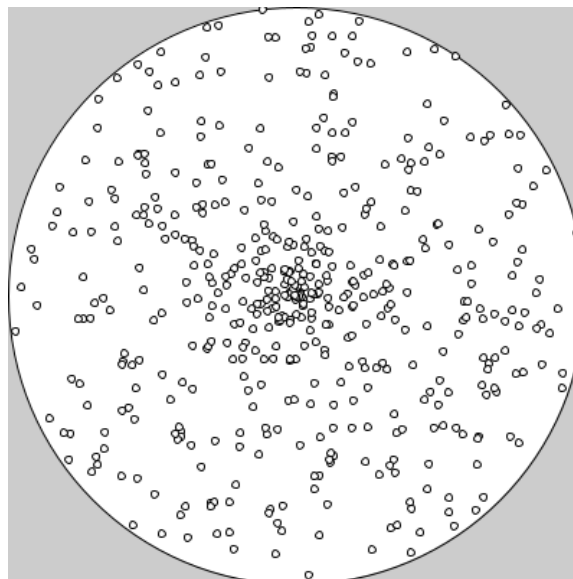
| Three Approaches

Rejection Sampling



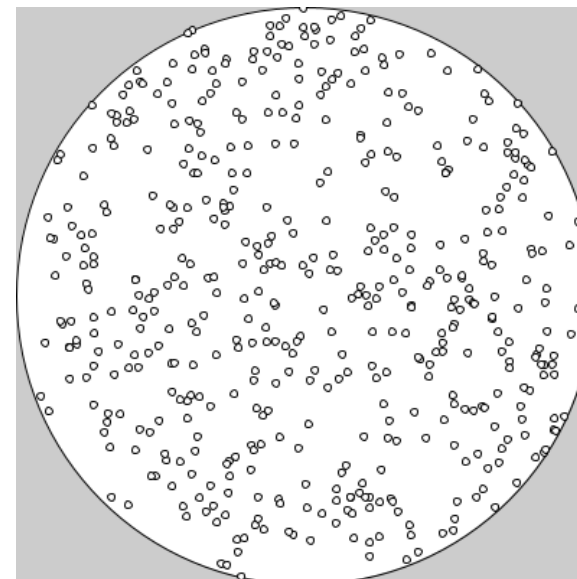
**Does not always produce
a point at each step**

Polar Coordinate Sampling



**Not uniformly sampled
(denser near the center)**

Inverse Transform Sampling



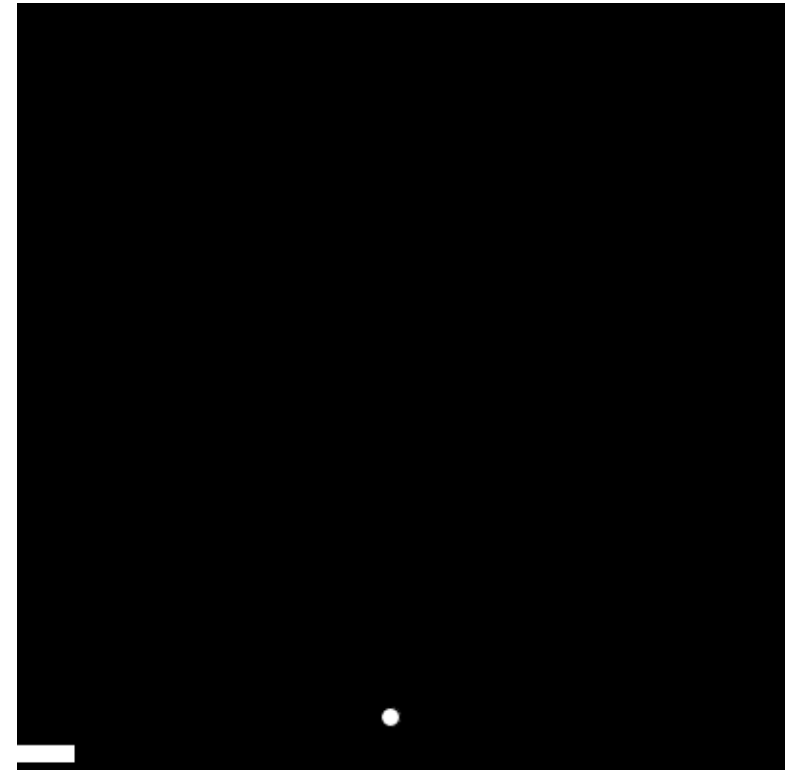
**Require some math
(not always tractable)**

| randomSeed(): Controlling the Randomness

- What if I want **the same random numbers every time**?
- random() is not truly random, but *random enough* for most use cases
 - Technically, we call these algorithms *pseudorandom algorithms*
- Pseudorandom algorithms take a **random seed** as input
 - The random seed determines the sequence of random numbers it generates
 - If you set the random seed to a fixed number, you'll get the exact same sequence of random numbers when calling random() and randomGaussian()
 - Use randomSeed(**seed**) to set the random seed
 - Any integer works: randomSeed(0) or randomSeed(1000) or randomSeed(1024)

🔥 HW 2: Paddle Ball Game

- Instructions will be sent by **emails** and released on the **course website**
- 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



| 🤔 Shooting the Ball to a Random Direction

```
speedX = speed * random(-1, 1);  
speedY = speed * random(-1, 1);
```

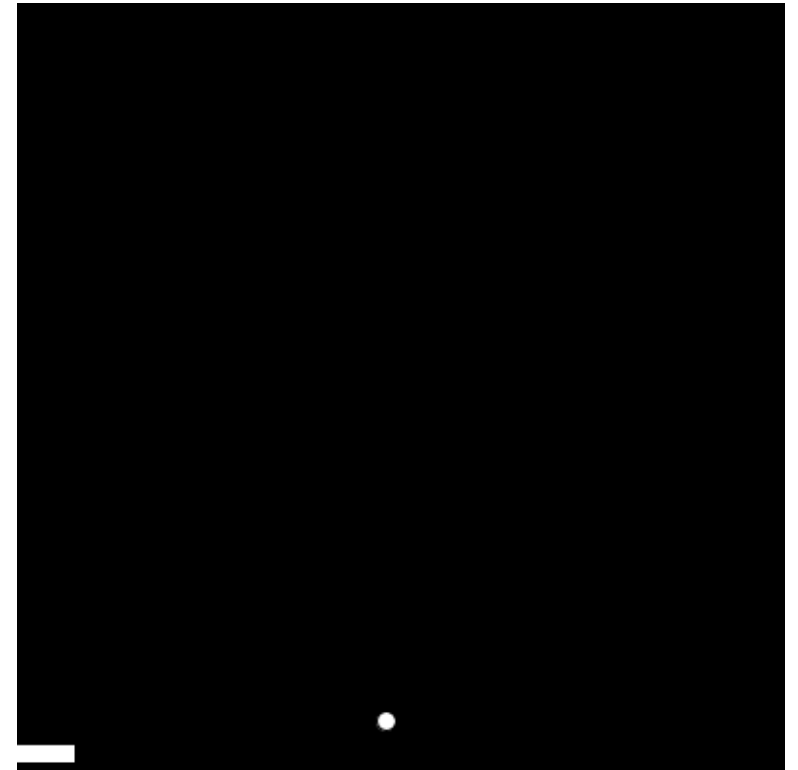
```
a = random(0, 1);  
speedX = speed * a;  
speedY = speed * sqrt(1 - a * a);
```

```
theta = random(0, TWO_PI);  
speedX = speed * cos(theta);  
speedY = speed * sin(theta);
```

🔥 HW 2: Sticky & Shrinking Paddle Ball Game

- To make the game more fun and challenging
- **Shrink the paddle bar size** after each catch
 - Until the bar reaches a reasonable minimum width
- Apply **stickiness between the paddle and ball**
 - A fixed ratio of the bar velocity will be added to the ball
 - Use **pmouseX** and **mouseX** to calculate the paddle velocity

The value of mouseX
in the previous frame



Recap

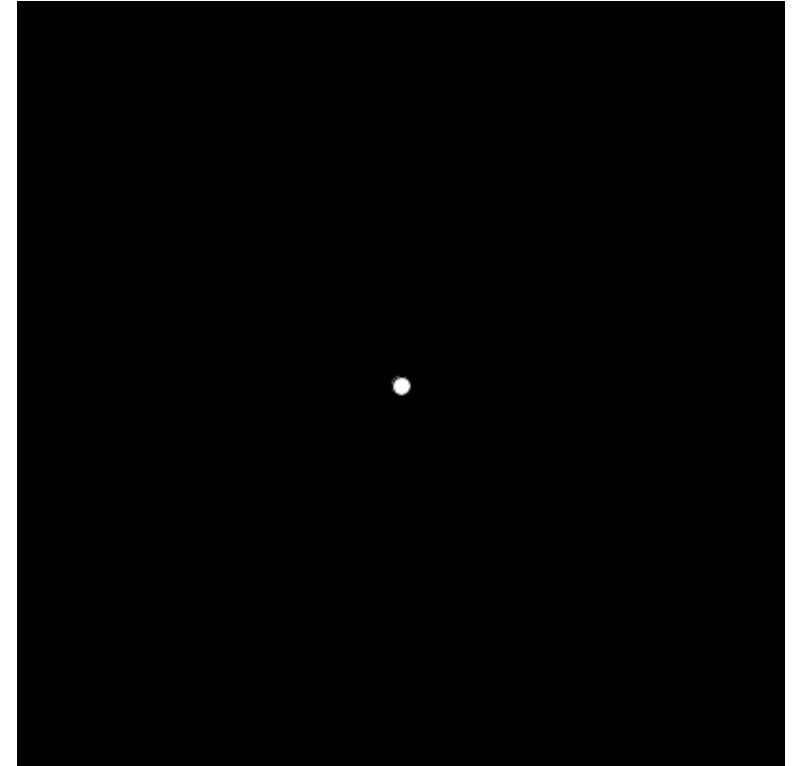
Built-in Keyboard Functions

Function	Called when
<code>keyPressed()</code>	a key is <i>pressed</i>
<code>keyReleased()</code>	a key is <i>released</i>
<code>keyTyped()</code>	a key is <i>clicked</i> (called repeatedly if held down)

🤔 Exercise: Use the Arrow Keys to Control a Ball

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

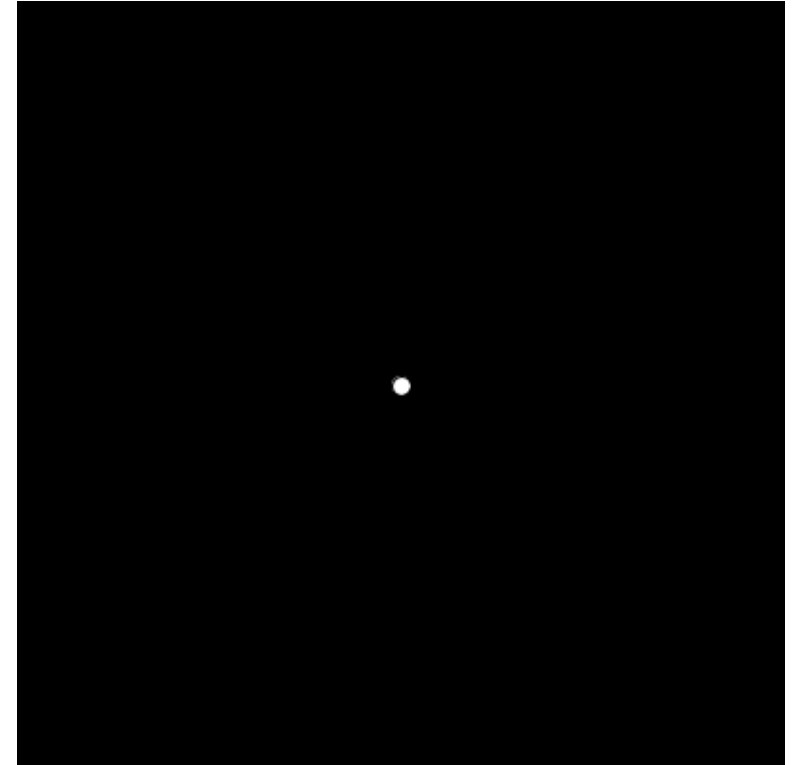
void keyPressed() {
  if (key == CODED) {
    if (keyCode == LEFT) {
      x = x - step;
    }
    if (keyCode == RIGHT) {
      x = x + step;
    }
    if (keyCode == UP) {
      y = y - step;
    }
    if (keyCode == DOWN) {
      y = y + step;
    }
  }
}
```



🤔 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;
    }
  }
}
```



switch Statement

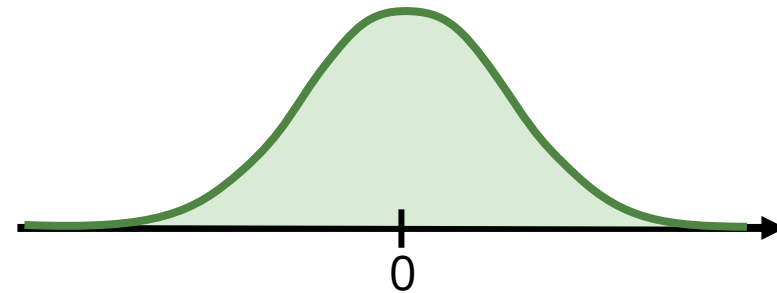
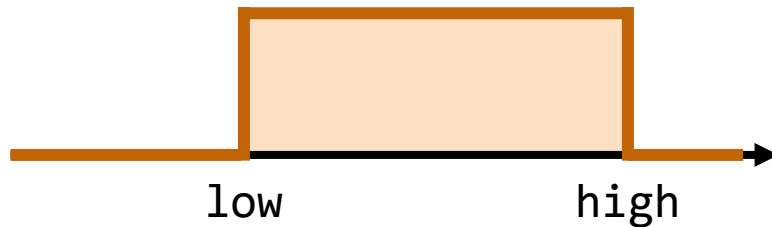
```
if (keyCode == LEFT) {  
    x = x - step;  
} else if (keyCode == RIGHT) {  
    x = x + step;  
} else if (keyCode == UP) {  
    y = y - step;  
} else if (keyCode == DOWN) {  
    y = y + step;  
}
```



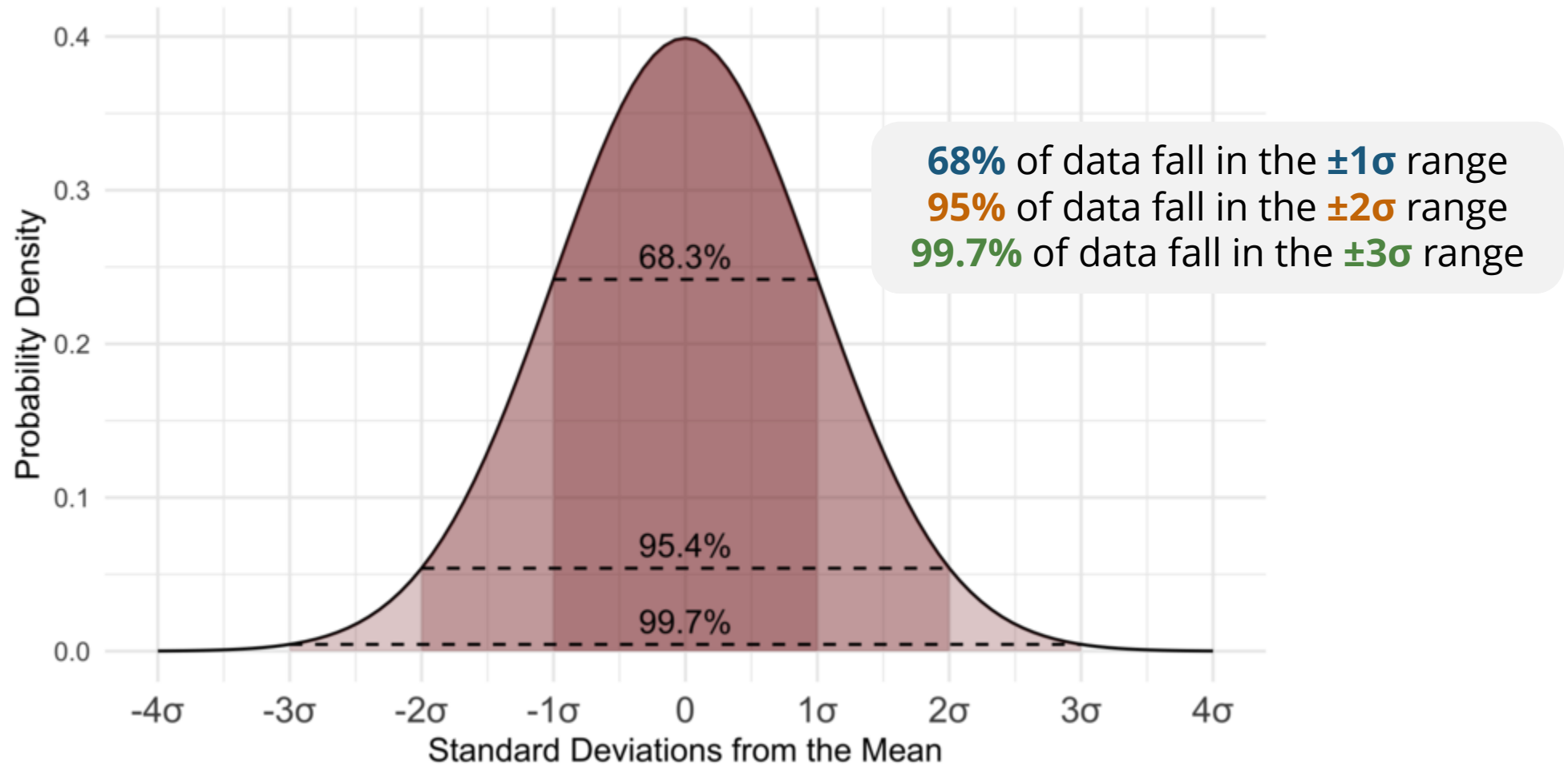
```
switch(keyCode) {  
    case LEFT:  
        x = x - step;  
        break;  
  
    case RIGHT:  
        x = x + step;  
        break;  
  
    case UP:  
        y = y - step;  
        break;  
  
    case DOWN:  
        y = y + step;  
        break;  
}
```

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 $N[0, 1]$



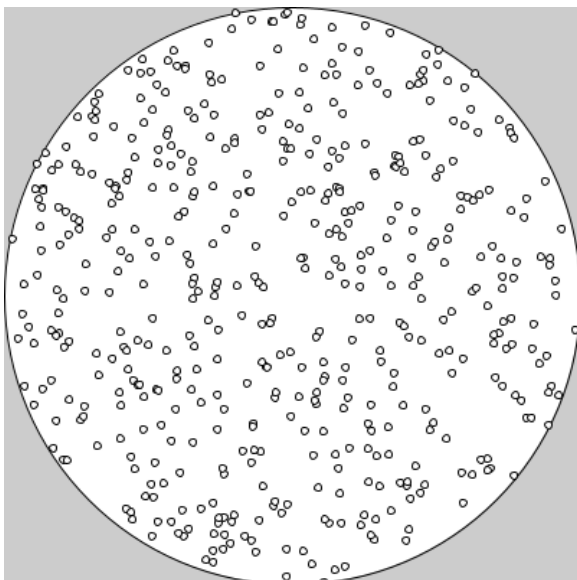
Gaussian Distribution & the 68–95–99.7 Rule



(Source: D Wells)

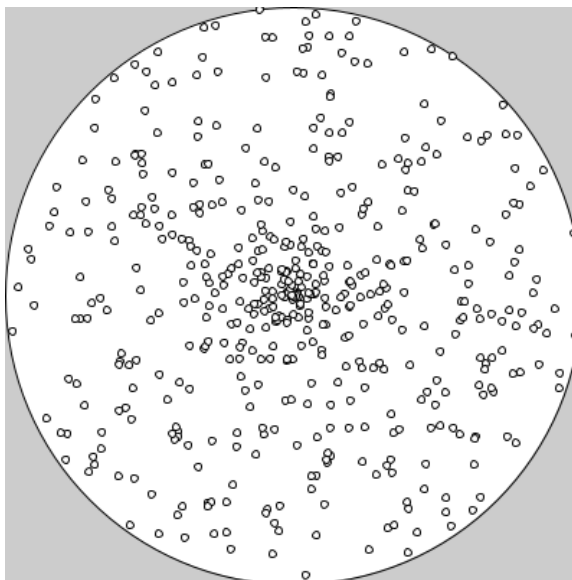
Three Approaches

Rejection Sampling



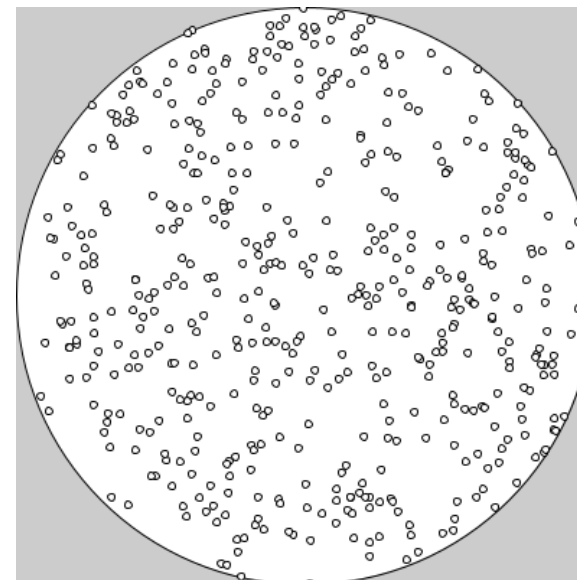
**Does not always produce
a point at each step**

Polar Coordinate Sampling



**Not uniformly sampled
(denser near the center)**

Inverse Transform Sampling



**Require some math
(not always tractable)**

Next Lecture

Loops

