PAT 204/504 (Fall 2024)

# Creative Coding

## Lecture 9: Transformations & 3D Graphics

Instructor: Hao-Wen Dong

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

# Midterm Assignment: Build Your Own Music Visualizer

- **Open-ended** assignment
- Use everything you've learned from the class (and beyond!)
- Instructions will be released on Gradescope
- Due at **11:59pm ET** on **October 7**
- Late submissions: **NOT Accepted** **(Submit early and update later!)**

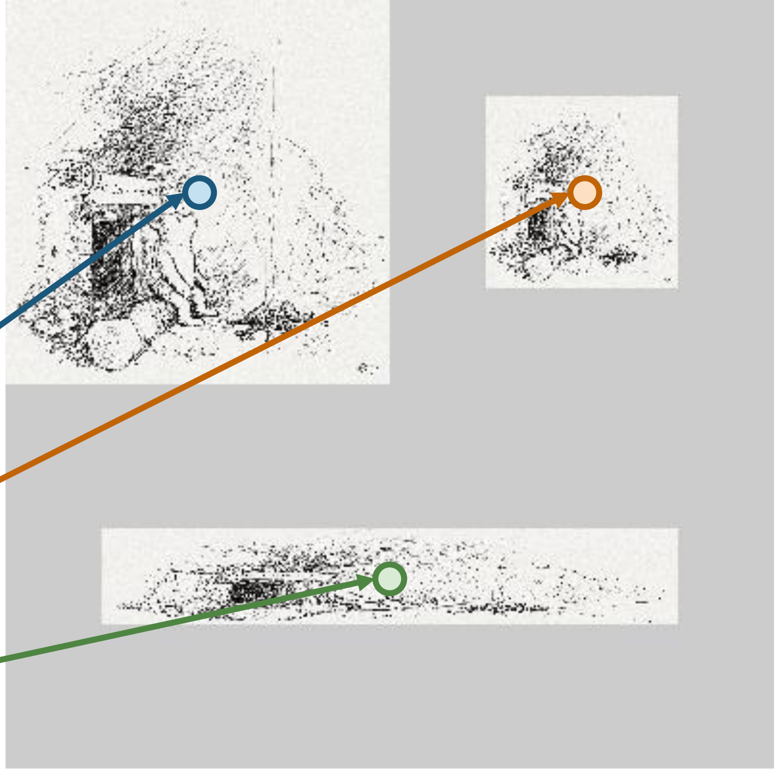
# Midterm Assignment – Rubrics

- Use **two of the following three concepts** (**10pt**)
  - **Loops and recursion**
  - **Data structures** (e.g., arrays, lists, dictionaries, etc.)
  - **Objects**
- **Clear documentation** in code (**5pt**)
- Live demo in class on **October 7** (**5pt**)

# (Recap) Example: Displaying Images

```
PImage img;

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

  img = loadImage("pooh.jpg");
}

void draw() {
  imageMode(CENTER);
  image(img, 100, 100, 200, 200);
  image(img, 300, 100, 100, 100);
  image(img, 200, 300, 300, 50);
}
```

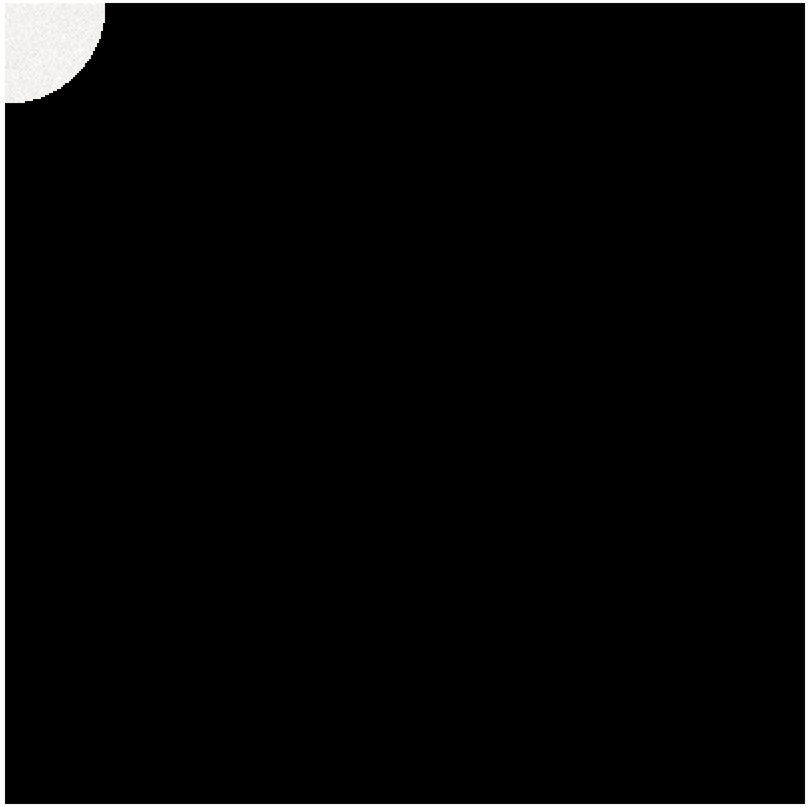
**Load the image**



# (Recap) **Exercise**: The Reveal Effect

```
void setup() {
  size(400, 400);
  img = loadImage("pooh.jpg");
  image(img, 0, 0, 400, 400);
  loadPixels();
  org = pixels.clone();
  background(0);
  loadPixels();
}

void draw() {
  for (int x = 0; x < width; x++) {
    for (int y = 0; y < height; y++) {
      int loc = x + y * width;
      float d = dist(x, y, mouseX, mouseY);
      if (d < 50) {
        pixels[loc] = org[loc];
      }
    }
  }
  updatePixels();
}
```

**Update the pixel values**



# (Recap) Example: Pointillism

```
PImage img;

void setup() {
  size(400, 400);
  img = loadImage("sakura.jpg");
  background(255);
  noLoop();
}

void draw() {
  for (int i = 0; i < 10000; i++) {
    int x = int(random(img.width));
    int y = int(random(img.height));
    int loc = x + y * img.width;

    img.loadPixels();
    float r = red(img.pixels[loc]);
    float g = green(img.pixels[loc]);
    float b = blue(img.pixels[loc]);

    noStroke();
    fill(r, g, b, 100);
    circle(x, y, 20);
  }
}
```

Pick a random pixel

Find the color of the pixel

Set the color of the circle

Draw the circle

# (Recap) Example: Loading a Movie

```
import processing.video.*;

Movie myMovie;

void setup() {
  size(640, 360);
  myMovie = new Movie(this, "movie.mov");
  myMovie.loop();
}

void movieEvent(Movie m) {
  m.read();
}

void draw() {
  image(myMovie, 0, 0);
}
```

**Import video library**

**Initialize the `movie` object**

**Called whenever a new frame is available to read**

# (Recap) Example: Webcam Capture

```
import processing.video.*;

Capture cam;

void setup() {
  size(640, 480);

  String[] cameras = Capture.list();
  if (cameras.length == 0) {
    println(“No cameras available for capture");
    exit();
  }
  cam = new Capture(this, cameras[0]);
  cam.start();
}

void draw() {
  if (cam.available() == true) cam.read();
  image(cam, 0, 0);
}
```

Get the webcam list

Use a specific webcam

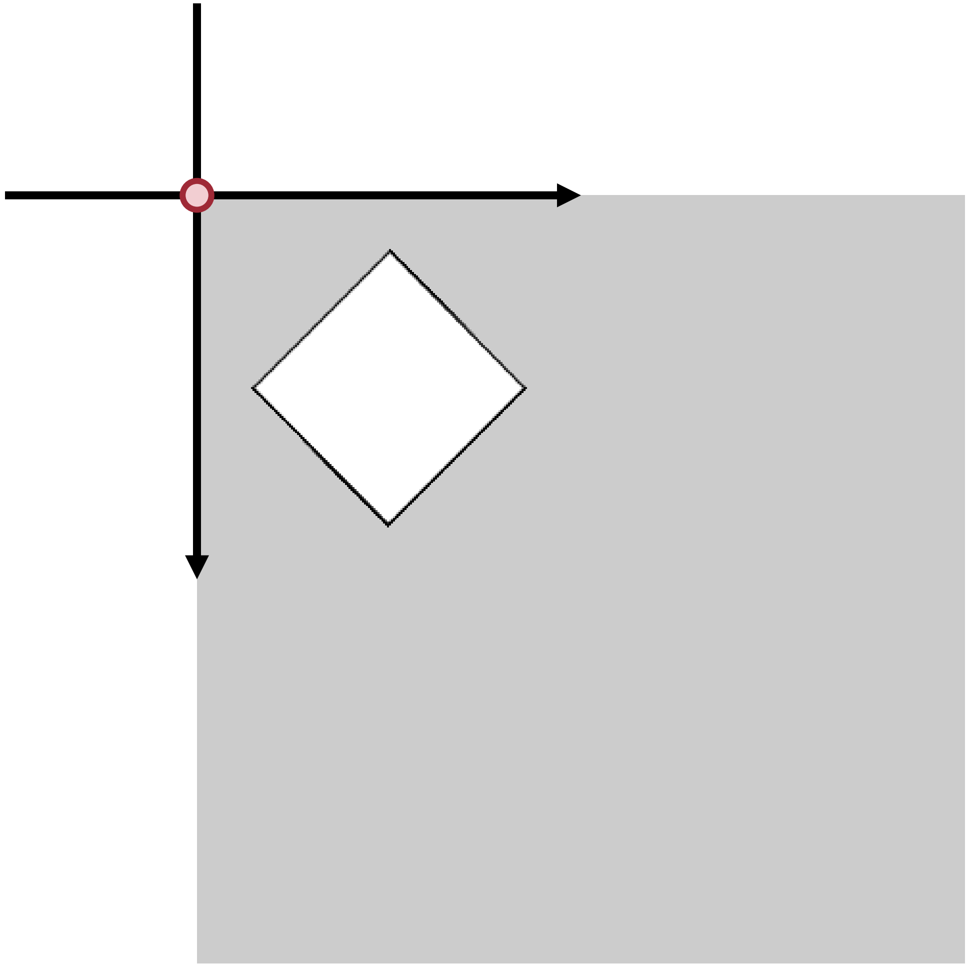
# Transformations

# Transformations

| | |
|---|---|
| • `translate(x, y)` | Translate the object |
| • `rotate(angle)` | Rotate the object |
| • `scale(s)` | Scale the object |
| • `scale(x, y)` | Scale the object |

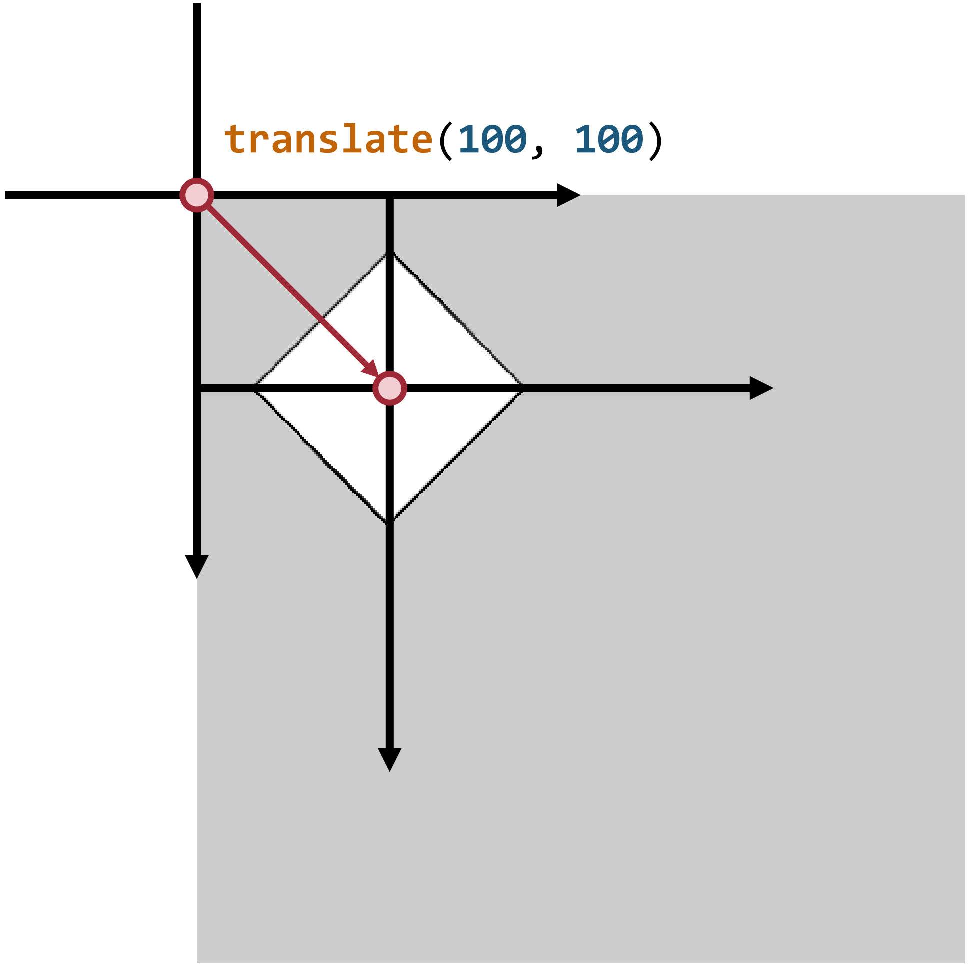
# Example: Rotated Square

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

void draw() {
  rectMode(CENTER);

  translate(100, 100);
  rotate(PI / 4);
  square(0, 0, 100);
}
```



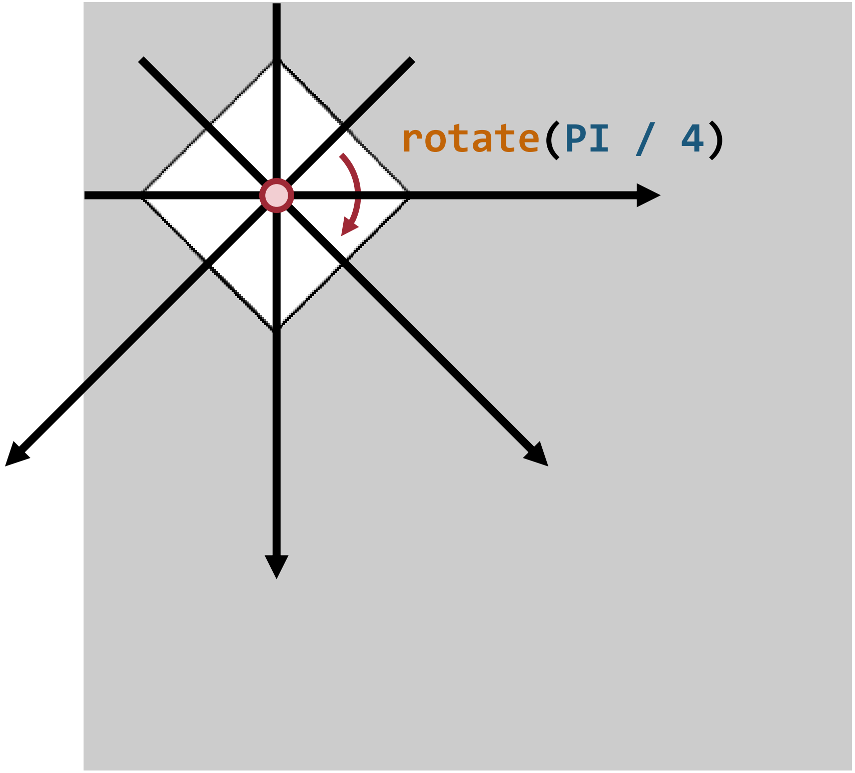
# Example: Rotated Square

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

void draw() {
  rectMode(CENTER);

  translate(100, 100);
  rotate(PI / 4);
  square(0, 0, 100);
}
```



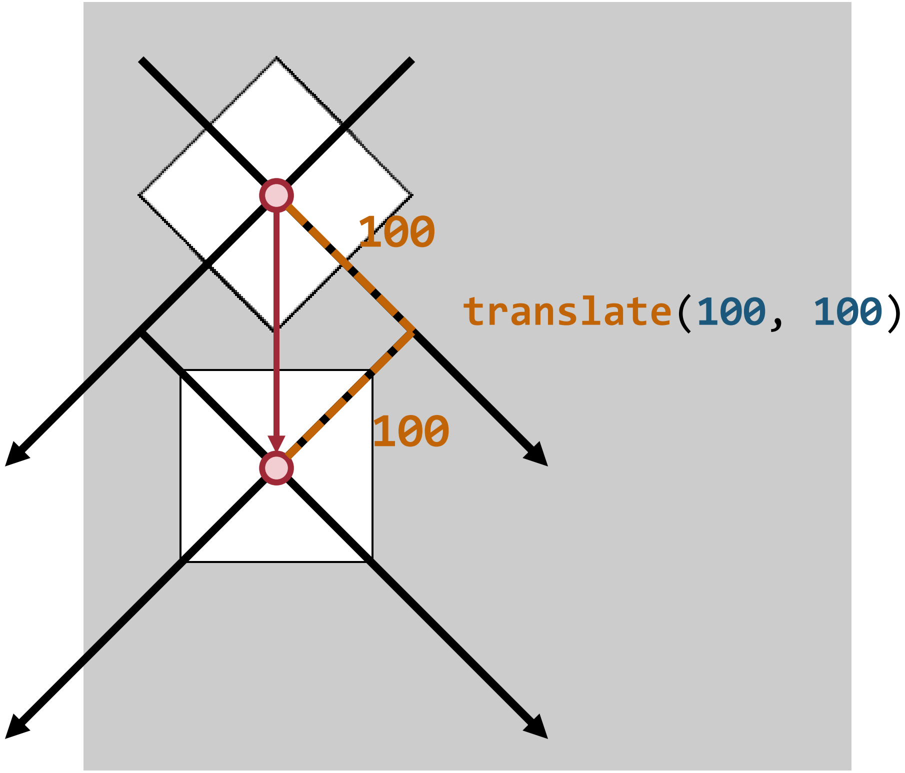
# Example: Rotated Square

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

void draw() {
  rectMode(CENTER);

  translate(100, 100);
  rotate(PI / 4);
  square(0, 0, 100);
}
```



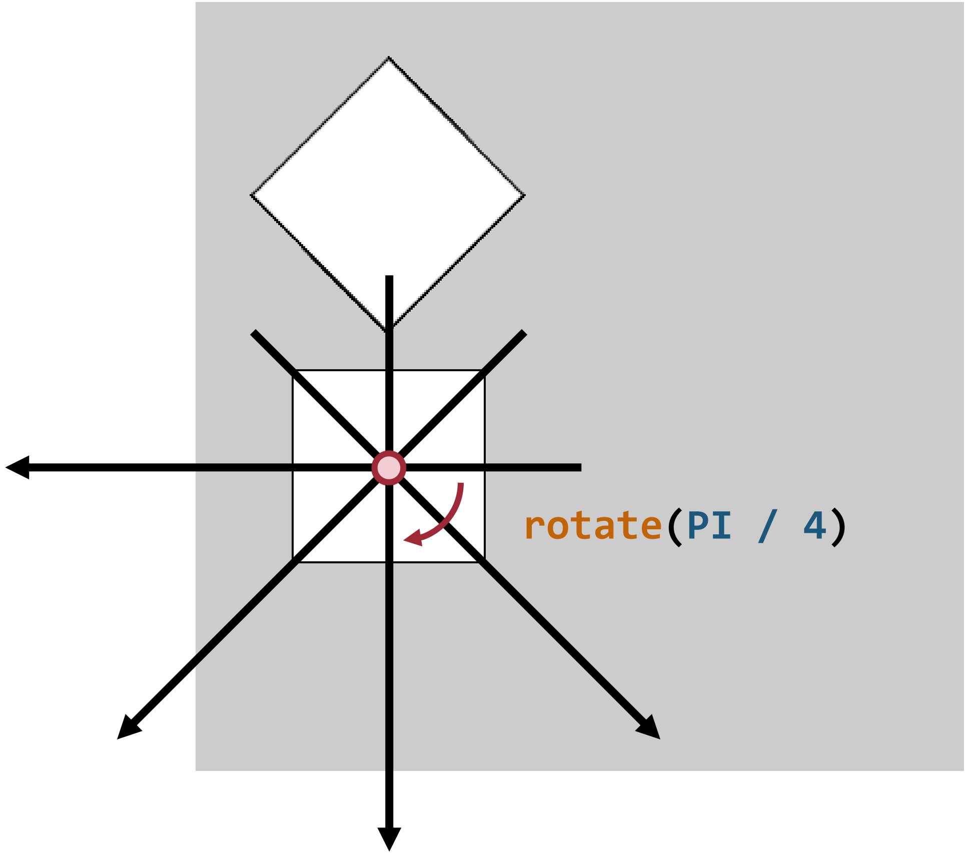
# Example: Rotated Squares

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

void draw() {
  rectMode(CENTER);

  translate(100, 100);
  rotate(PI / 4);
  square(0, 0, 100);

  translate(100, 100);
  rotate(PI / 4);
  square(0, 0, 100);
}
```



# Example: Rotated Squares

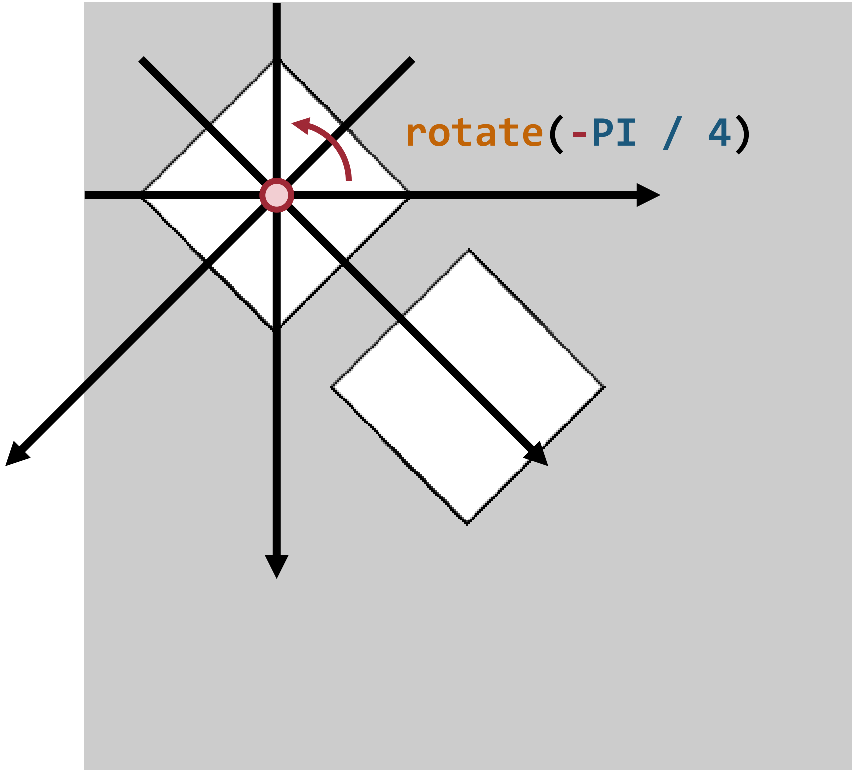
```
void setup() {
  size(400, 400);
}

void draw() {
  rectMode(CENTER);

  translate(100, 100);
  rotate(PI / 4);
  square(0, 0, 100);

  translate(100, 100);
  rotate(PI / 4);
  square(0, 0, 100);
}
```



# Example: Rotated Squares

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

void draw() {
  rectMode(CENTER);

  translate(100, 100);
  rotate(PI / 4);
  square(0, 0, 100);

  rotate(-PI / 4);
  translate(100, 100);
  rotate(PI / 4);
  square(0, 0, 100);
}
```

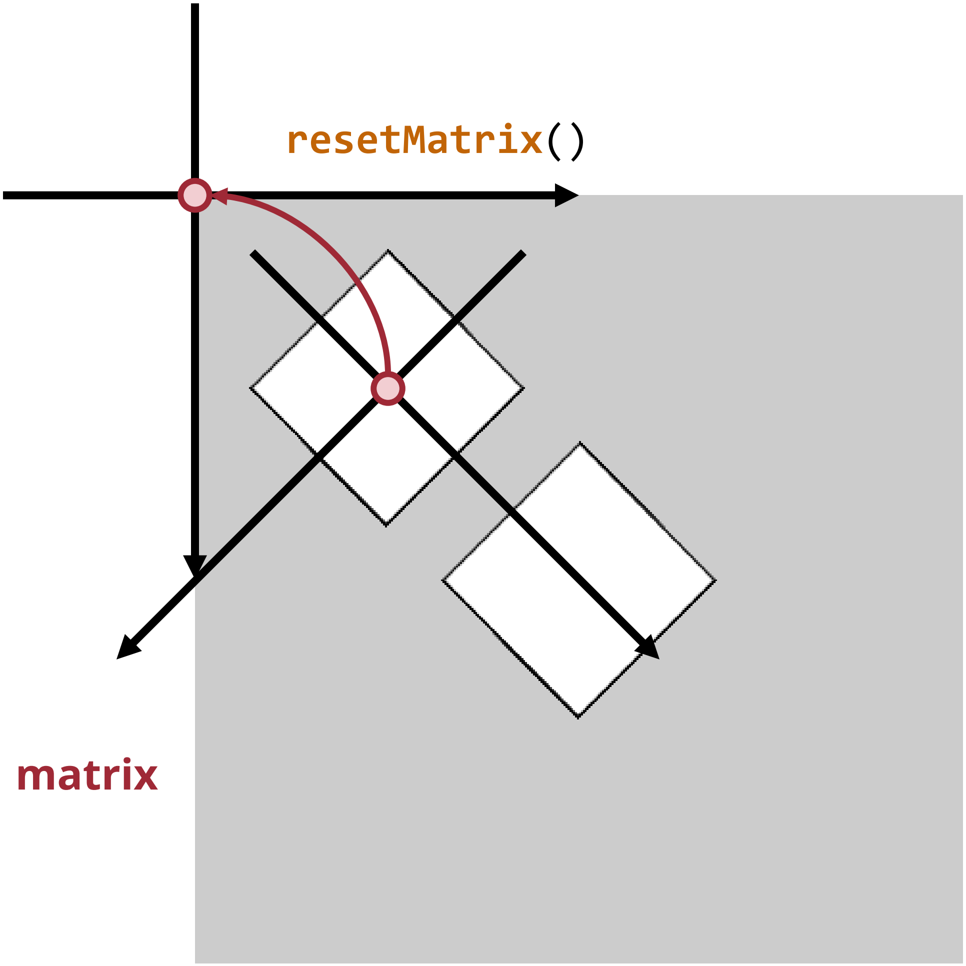
**Revert the rotation**



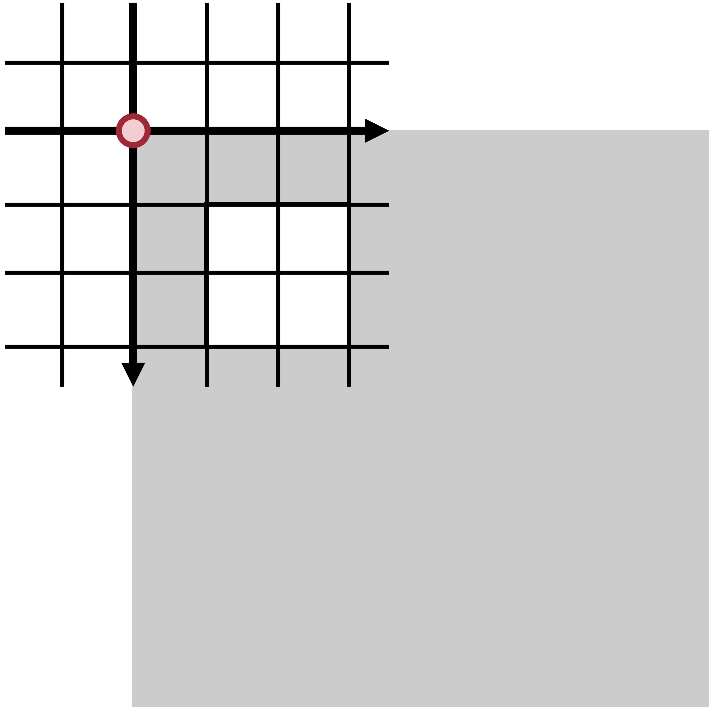
# Example: Rotated Squares

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

void draw() {
  rectMode(CENTER);

  translate(100, 100);
  rotate(PI / 4);
  square(0, 0, 100);

  resetMatrix();
  translate(200, 200);
  rotate(PI / 4);
  square(0, 0, 100);
}
```

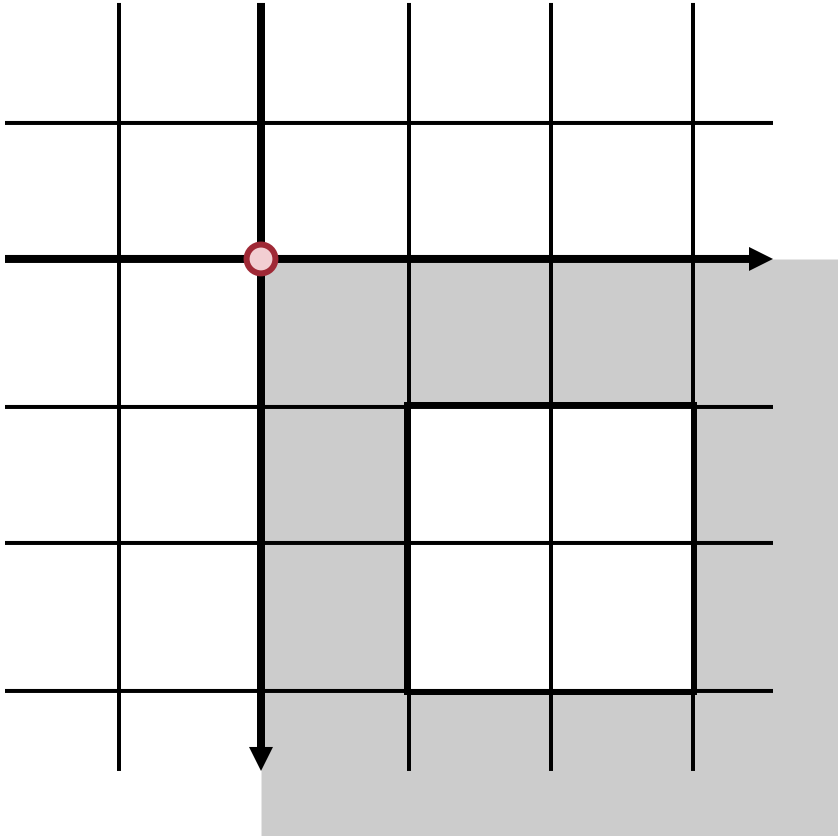
**Reset the transformation matrix**



# Transformation Matrix

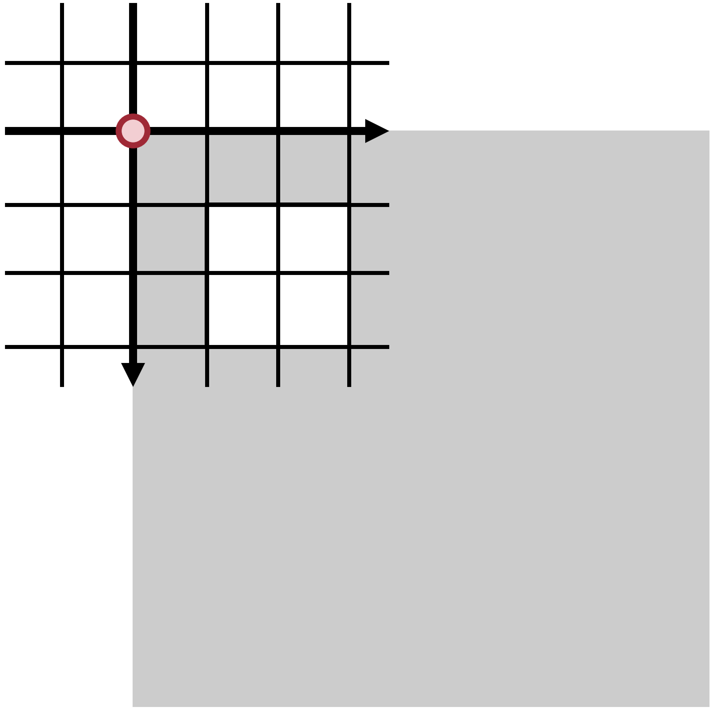


```
square(100, 100, 100);
```

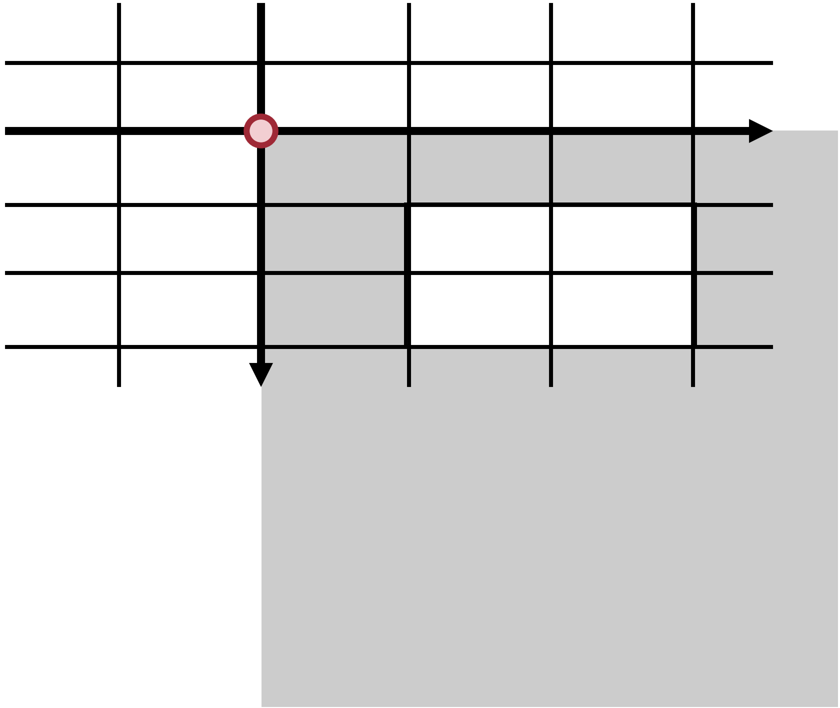


```
scale(2);
square(100, 100, 100);
```

# Transformation Matrix



```
square(100, 100, 100);
```

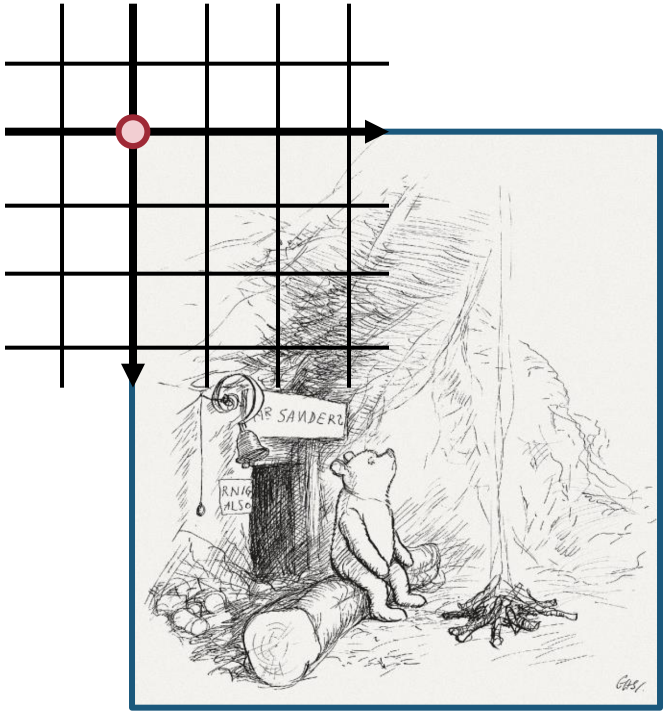


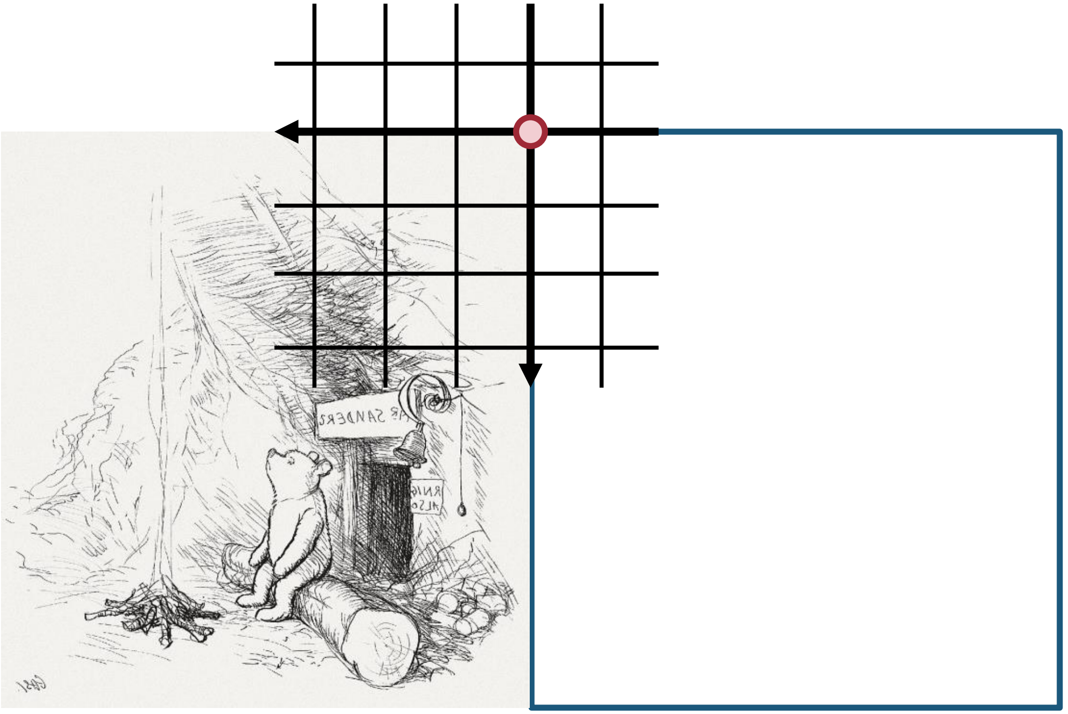
```
scale(2, 1);
square(100, 100, 100);
```

# Example: Mirroring Capture

```
void draw() {
  image(video, 0, 0);
}
```

```
void draw() {
  scale(-1, 1);
  image(video, 0, 0);
}
```



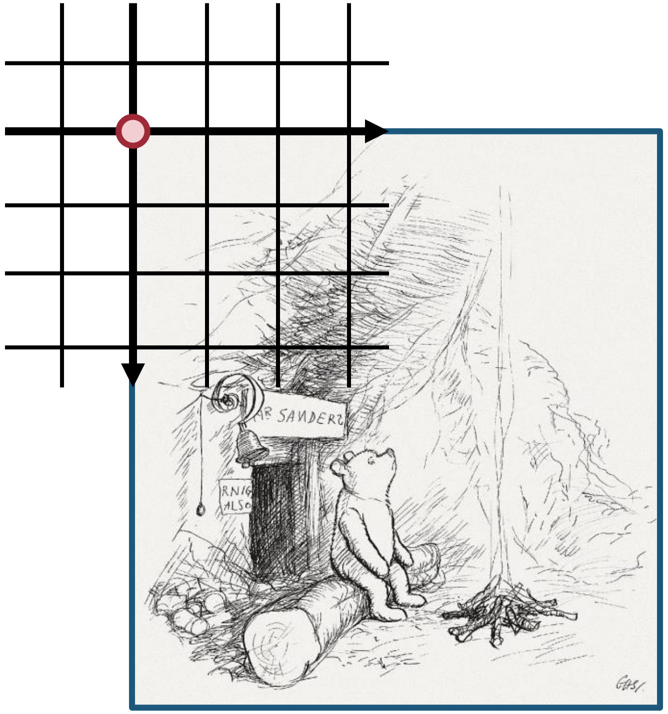


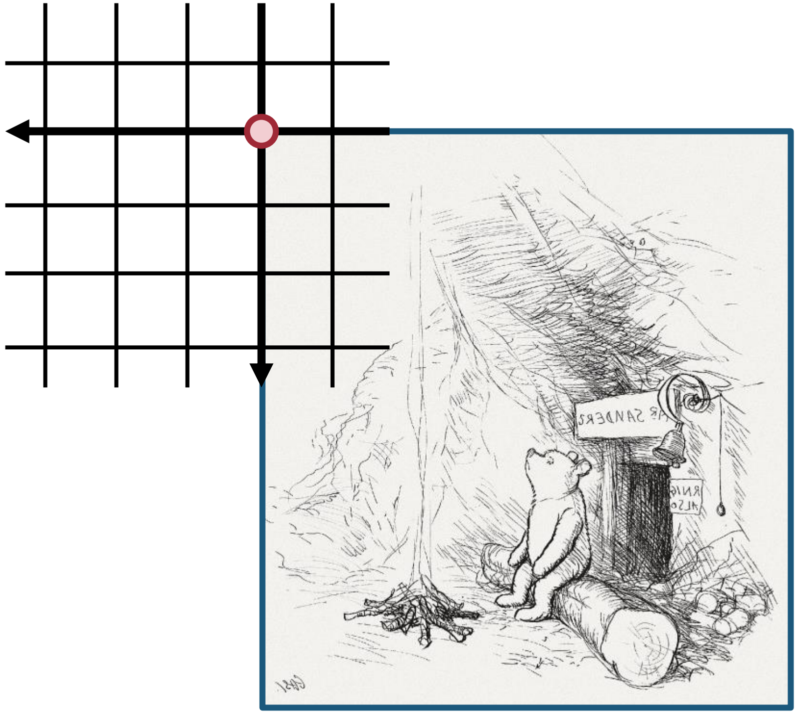
# Example: Mirroring Capture

```
void draw() {
  image(video, 0, 0);
}
```

→

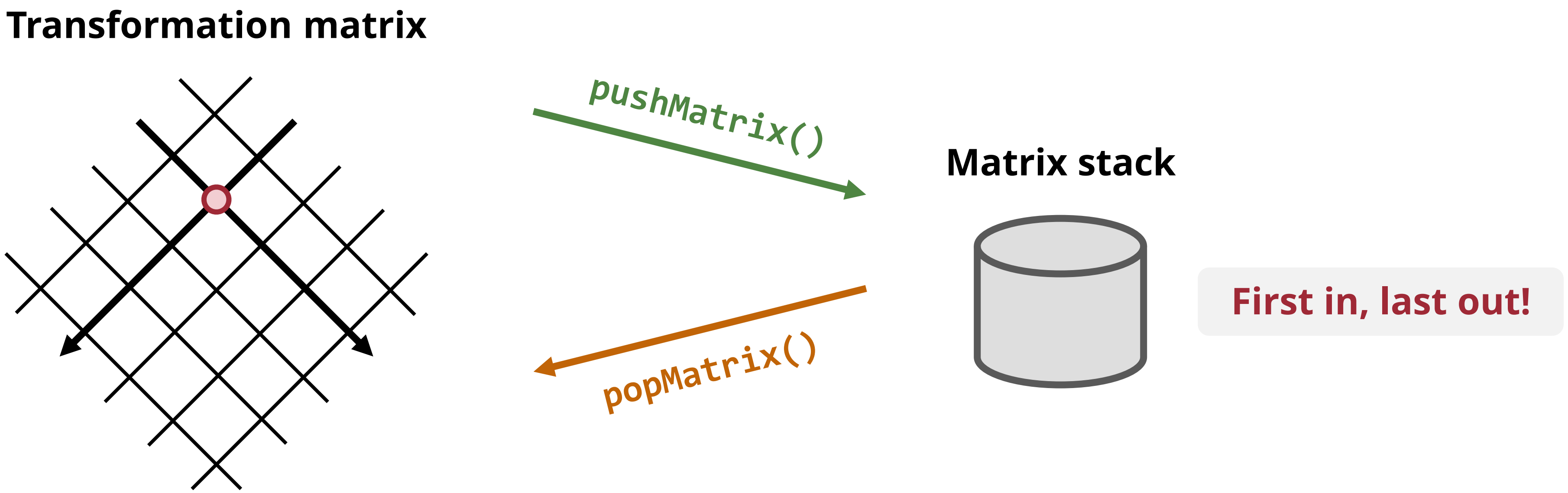
```
void draw() {
  scale(-1, 1);
  image(video, -video.width, 0);
}
```





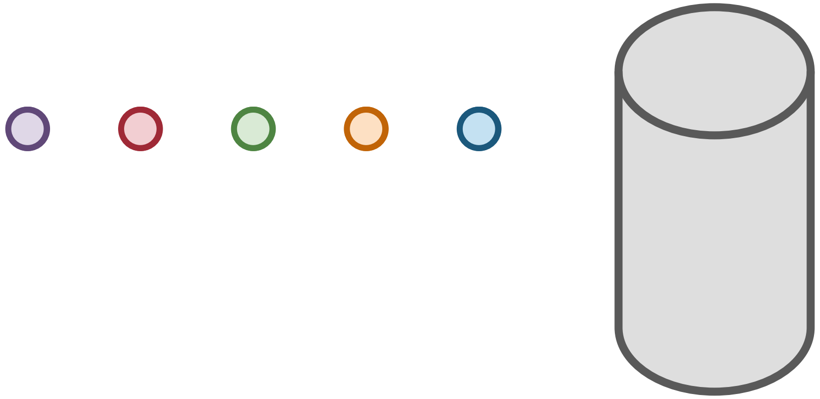
# Matrix Transforms

- `resetMatrix()` Reset to identity matrix
- `pushMatrix()` Push the current transformation matrix to the stack
- `popMatrix()` Pop the latest transformation matrix off the stack

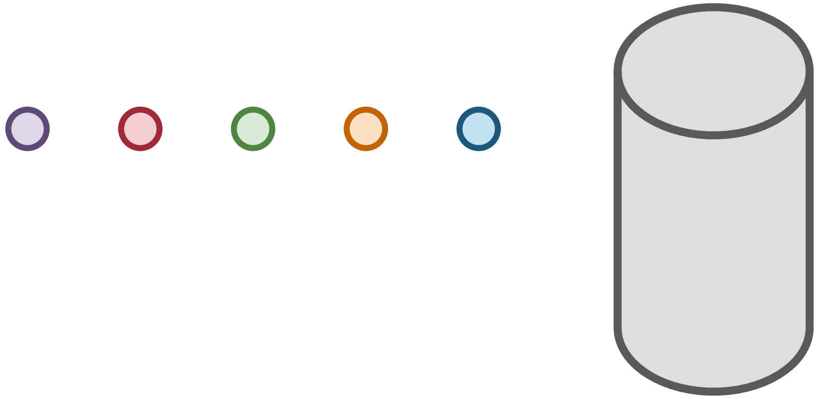


# Stack vs Queue

**Stack**

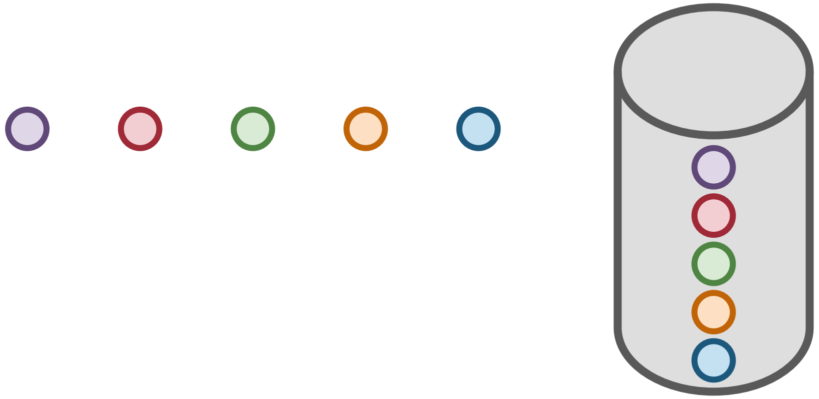


**Queue**

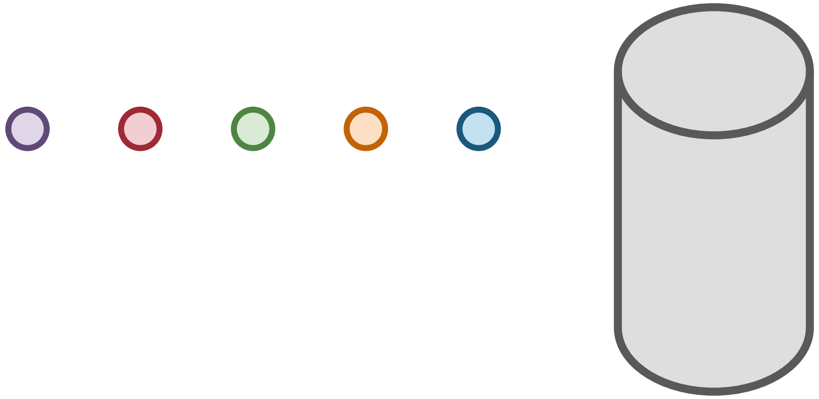


# Stack vs Queue

Stack

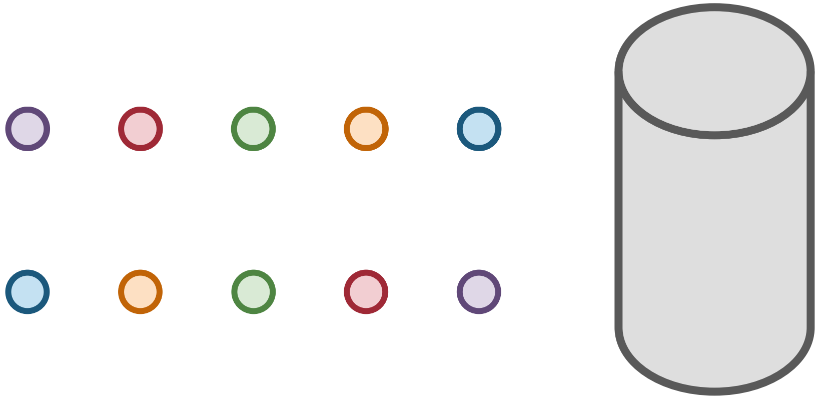


Queue

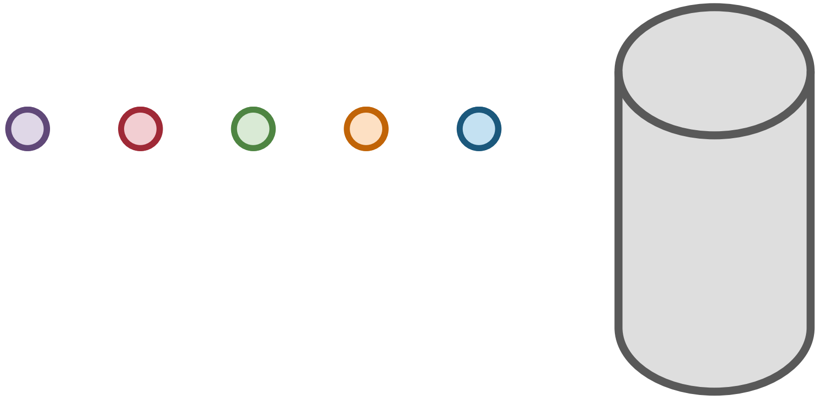


# Stack vs Queue

**Stack**

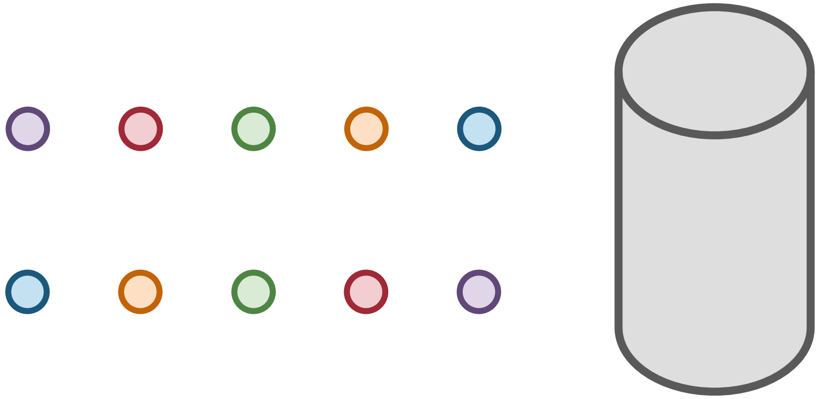


**Queue**

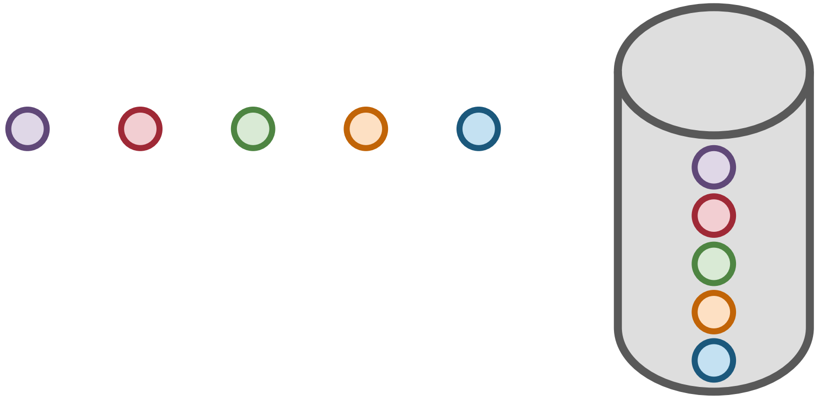


# Stack vs Queue

**Stack**

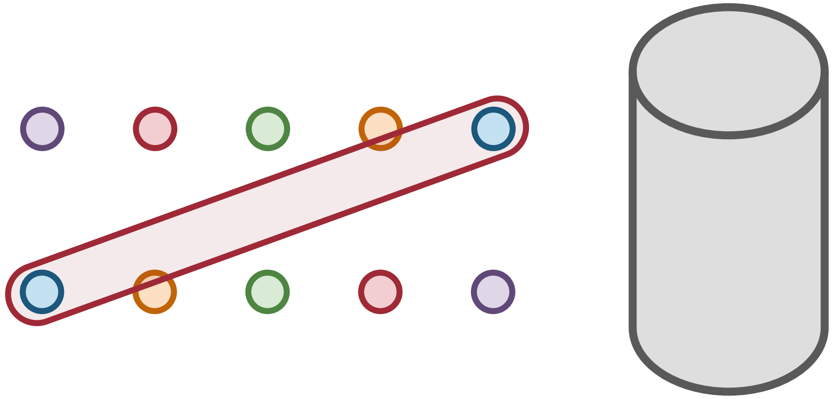


**Queue**



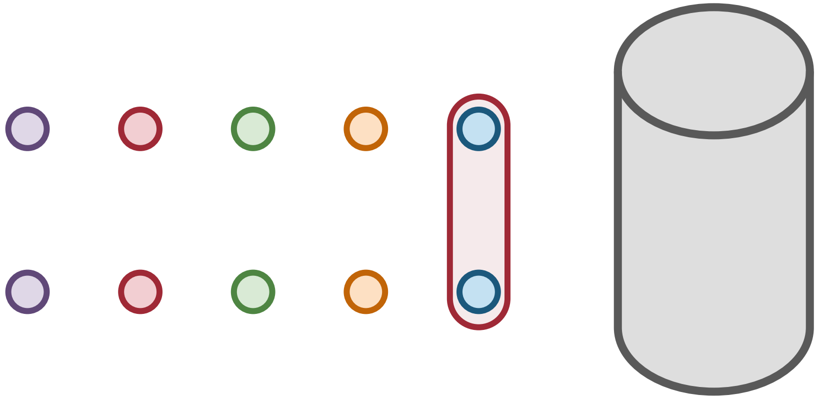
# Stack vs Queue

## Stack



**First in last out**

## Queue



**First in first out**

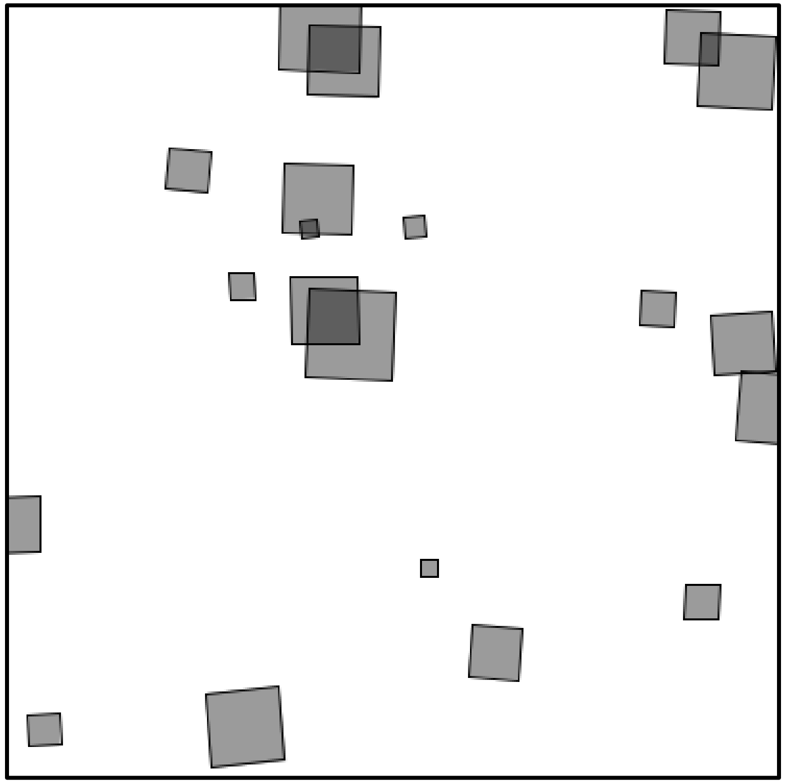
# Example: Spinning Objects

```
Rotater[] rotaters = new Rotater[20];
float x, y, speed, w;

void setup() {
  size(400, 400);
  for (int i = 0; i < rotaters.length; i++) {
    x = random(width);
    y = random(height);
    speed = random(-0.1, 0.1);
    w = random(5, 50);
    rotaters[i] = new Rotater(x, y, speed, w);
  }
}

void draw() {
  background(255);
  for (Rotater rotater: rotaters) {
    rotater.spin();
    rotater.display();
  }
}
```

**Declare an array of rotater objects**

**Initialize each rotater with a random position, size and rotation speed**

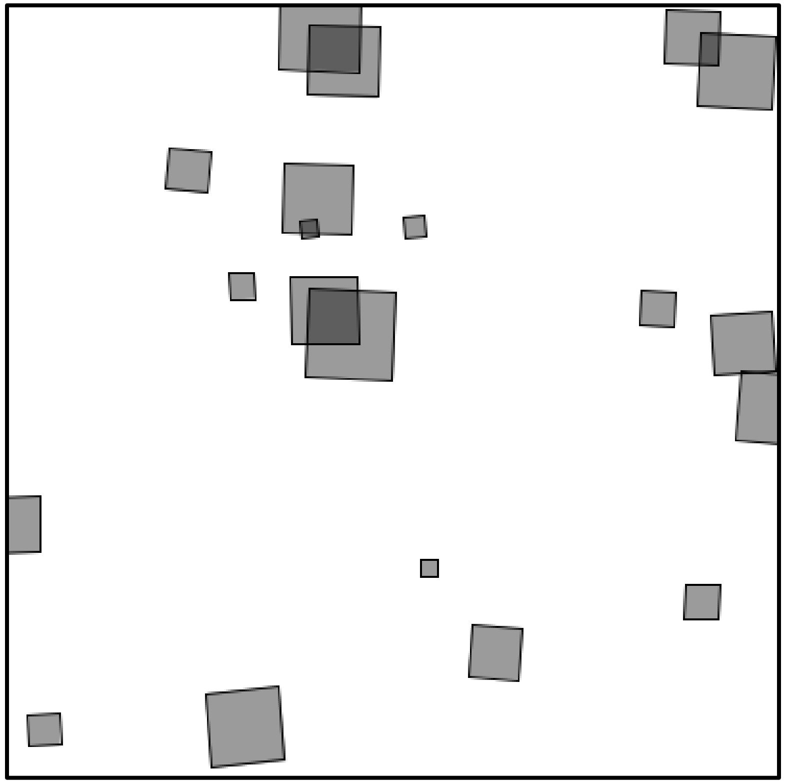
**Spin and show the rotaters!**



# Example: Spinning Objects

```
class Rotater {
  float x, y;    // x,y location
  float theta;   // angle of rotation
  float speed;   // speed of rotation
  float w;       // size of rectangle

  Rotater(float x, float y, float speed, float w) {
    this.x = x;
    this.y = y;
    theta = 0;
    this.speed = speed;
    this.w = w;
  }

  void spin() {
    theta += speed;
  }

  ...
}
```

Spin the rotater!

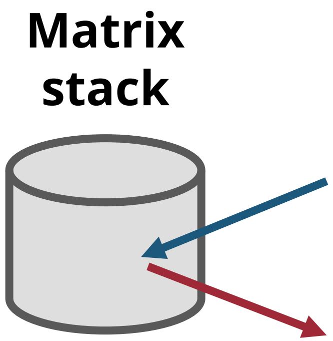


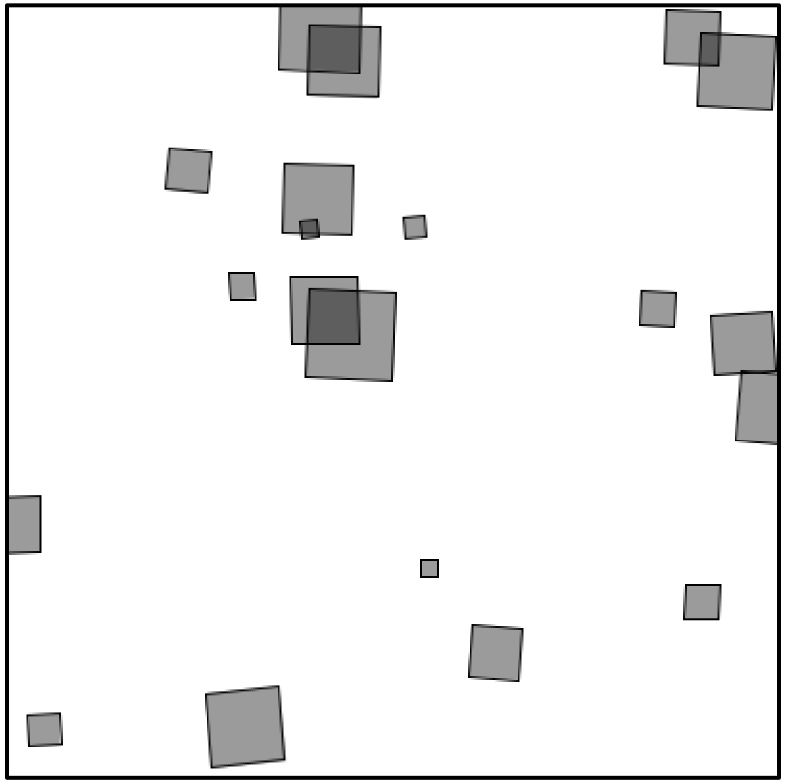
# Example: Spinning Objects

```
class Rotater {
  ...

  void spin() {
    theta += speed;
  }

  void display() {
    rectMode(CENTER);
    stroke(0);
    fill(0, 100);

    pushMatrix();
    translate(x, y);
    rotate(theta);
    rect(0, 0, w, w);
    popMatrix();
  }
}
```

Store the current matrix

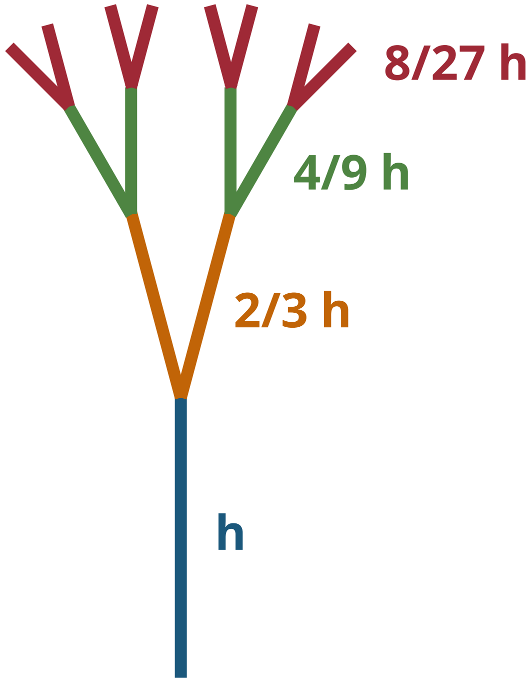
Restore the stored matrix

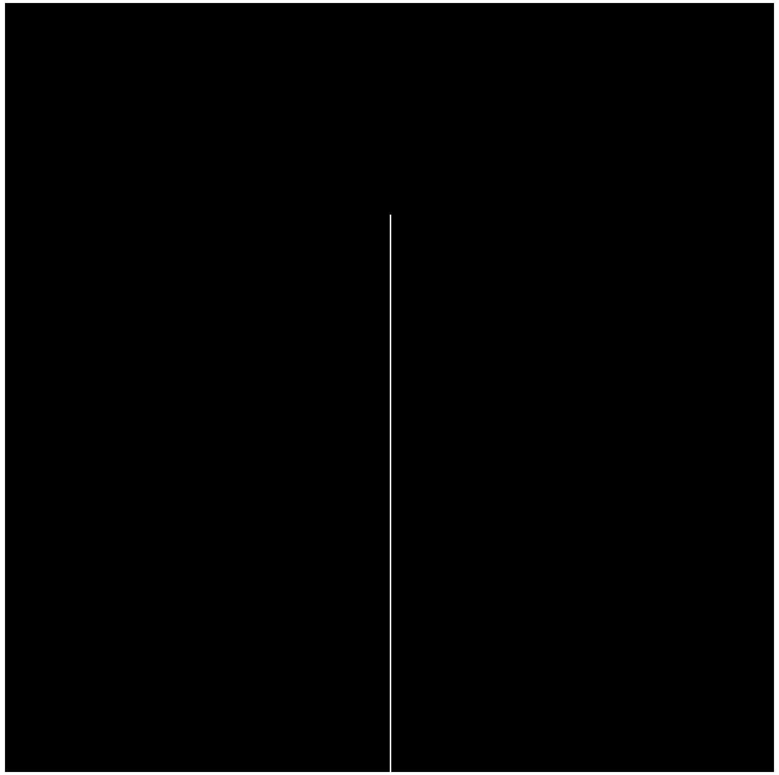




# (Recap) Example: Recursive Tree

- 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

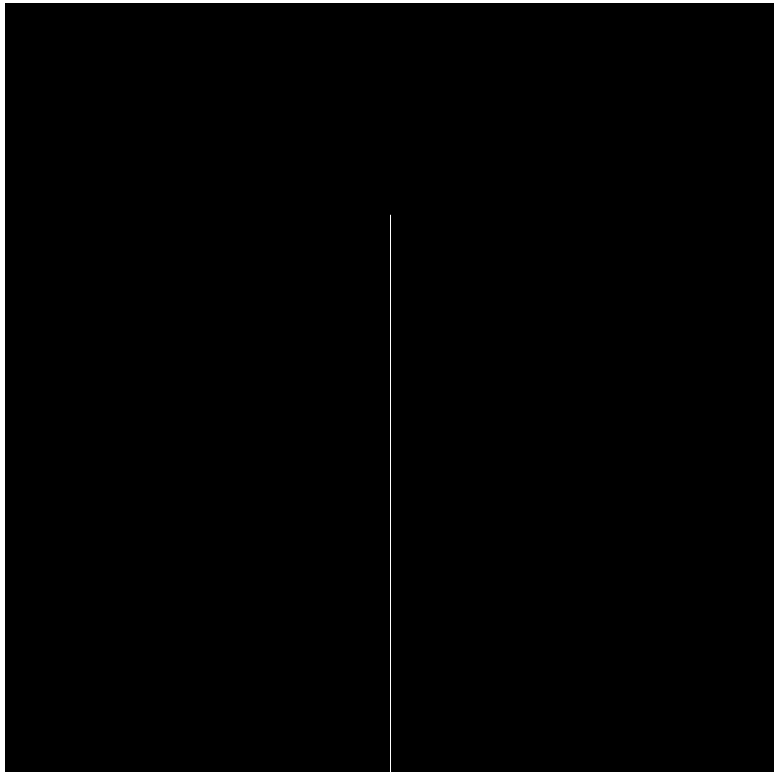




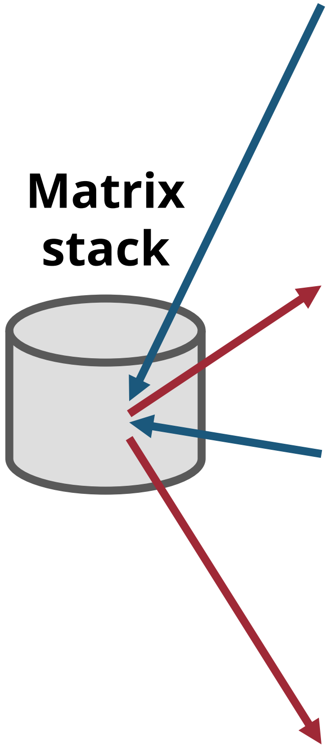
# (Recap) Example: Recursive Tree

```
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();
}
```

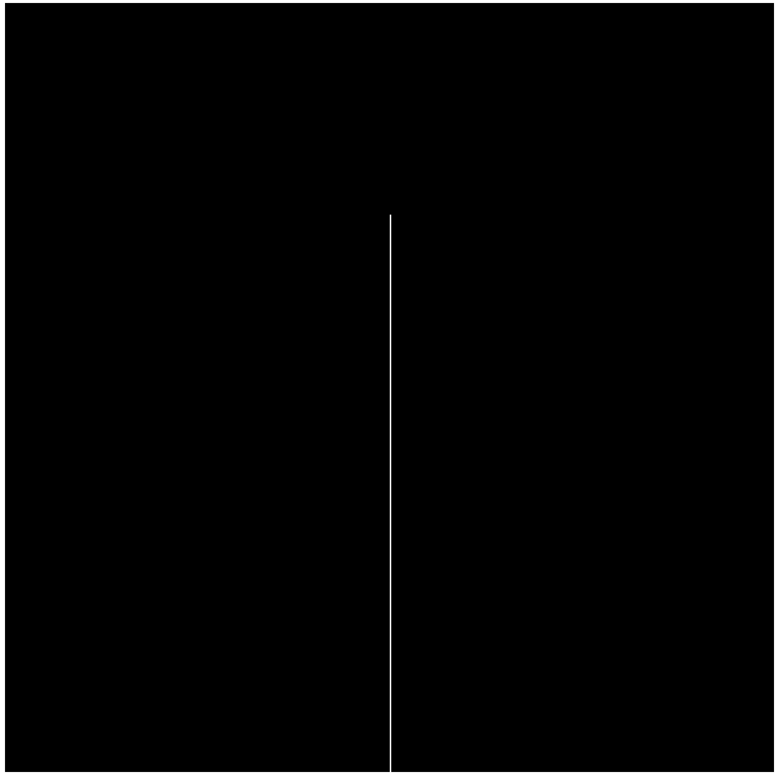
**Stop condition**



# 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();
}
```



# 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();
}
```

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

  // Right branch
```

**Why not?**

```
  resetMatrix();
  rotate(theta);
  line(0, 0, 0, -h * scale);
  translate(0, -h * scale);
  branch(h * scale);

  // Left branch
```

**Why not?**

```
  resetMatrix();
  rotate(-theta);
  line(0, 0, 0, -h * scale);
  translate(0, -h * scale);
  branch(h * scale);
}
```

# 3D Graphics

# 3D Graphics in Processing

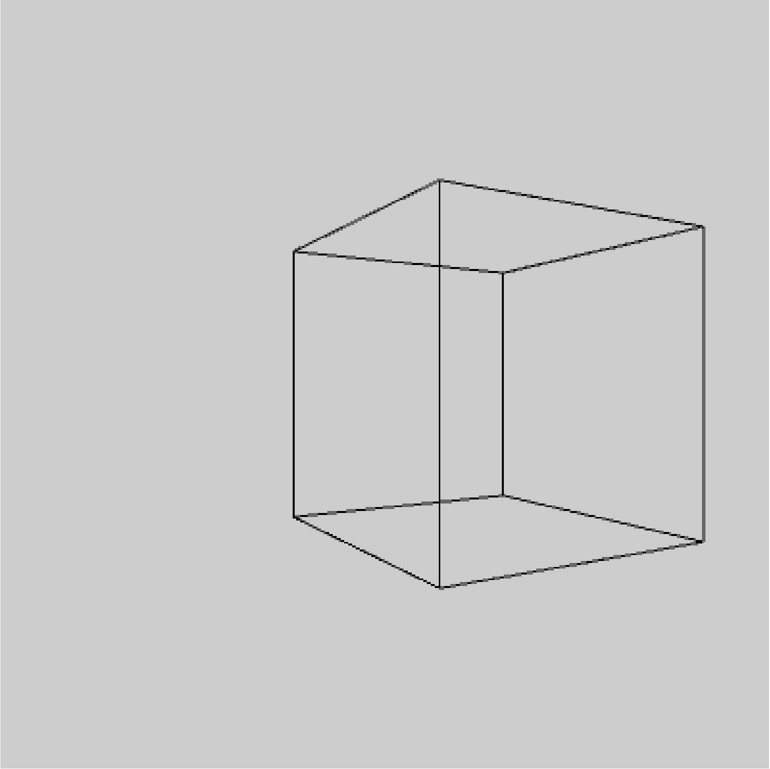
- Enable by using the 3D renderer
  - `size(800, 800, P3D)`

- 3D Primitives
  - `Box(size)`
  - `Box(width, height, depth)`
  - `Sphere(radius)`

- Heavily rely on **transformation**!

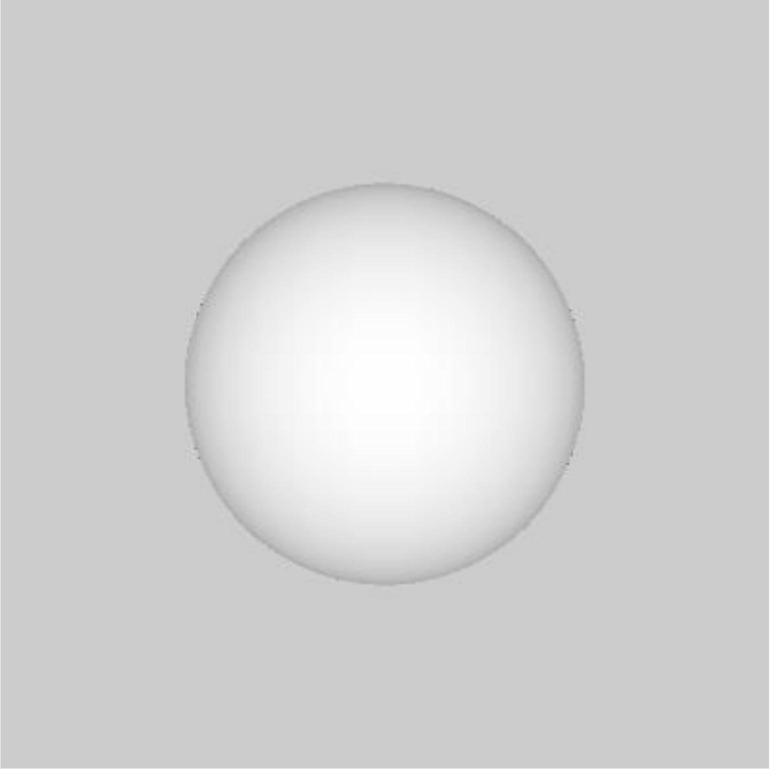
# Example: Box

```
size(400, 400, P3D);
translate(250, 200, 0);
rotateY(0.5);
fill(0, 10);
box(150);
```



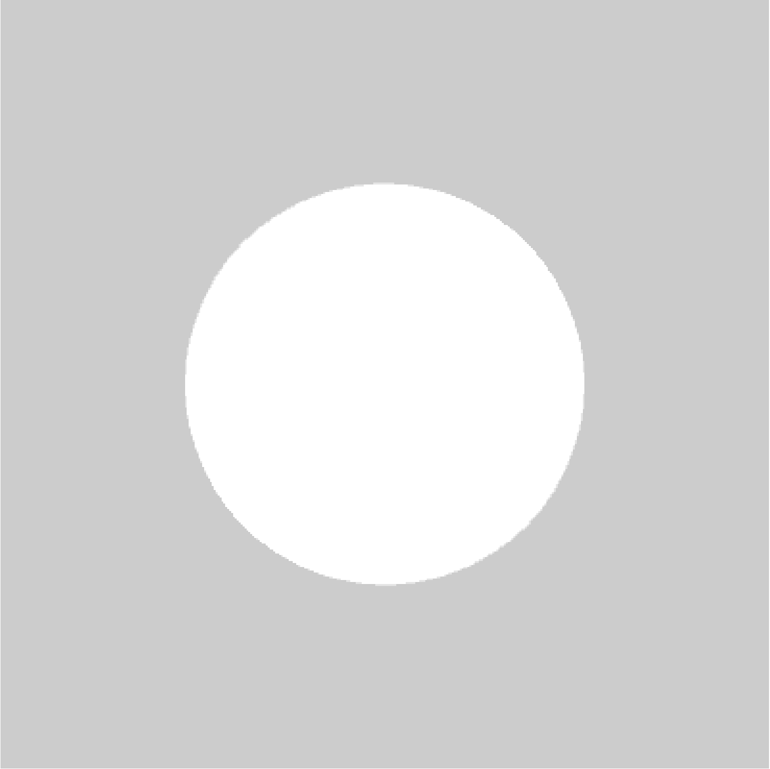
# Example: Sphere

```
size(400, 400, P3D);
noStroke();
lights();
translate(200, 200, 0);
sphere(100);
```



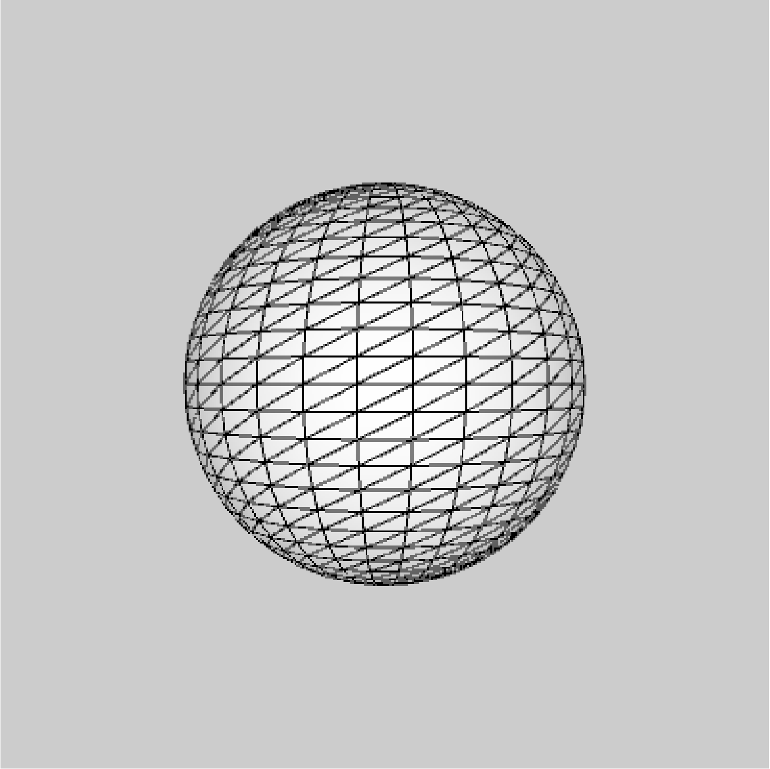
# Example: Sphere

```
size(400, 400, P3D);
noStroke();
~~lights();~~
translate(200, 200, 0);
sphere(100);
```



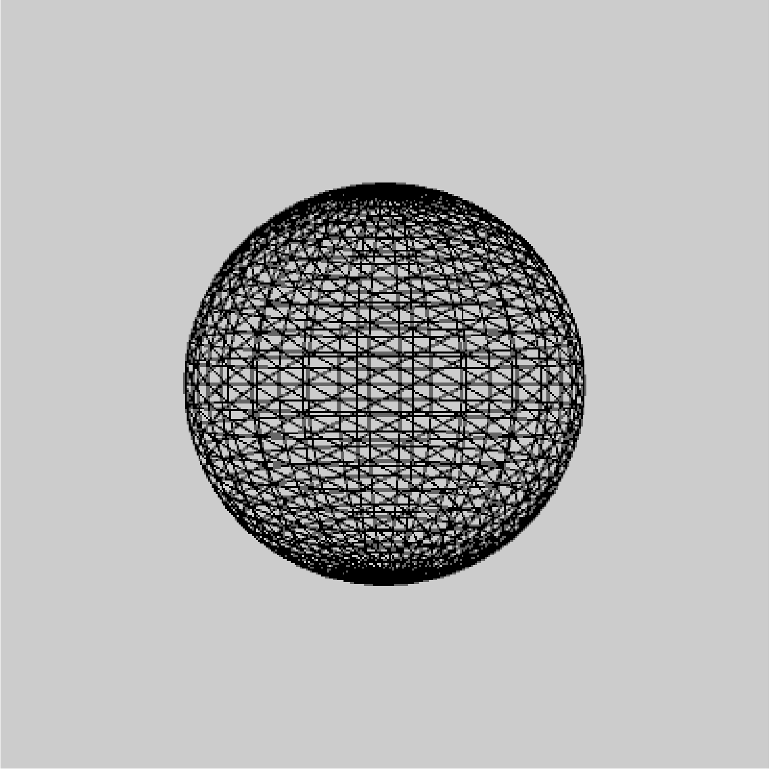
# Example: Sphere

```
size(400, 400, P3D);
~~noStroke();~~
lights();
translate(200, 200, 0);
sphere(100);
```



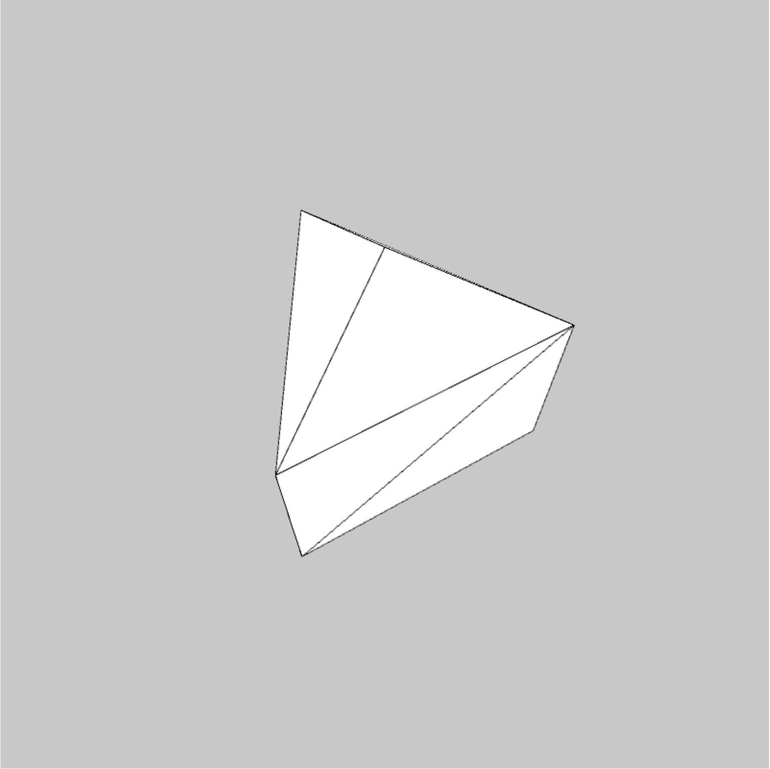
# Example: Sphere

```
size(400, 400, P3D);
noStroke();
noFill();
translate(200, 200, 0);
sphere(100);
```



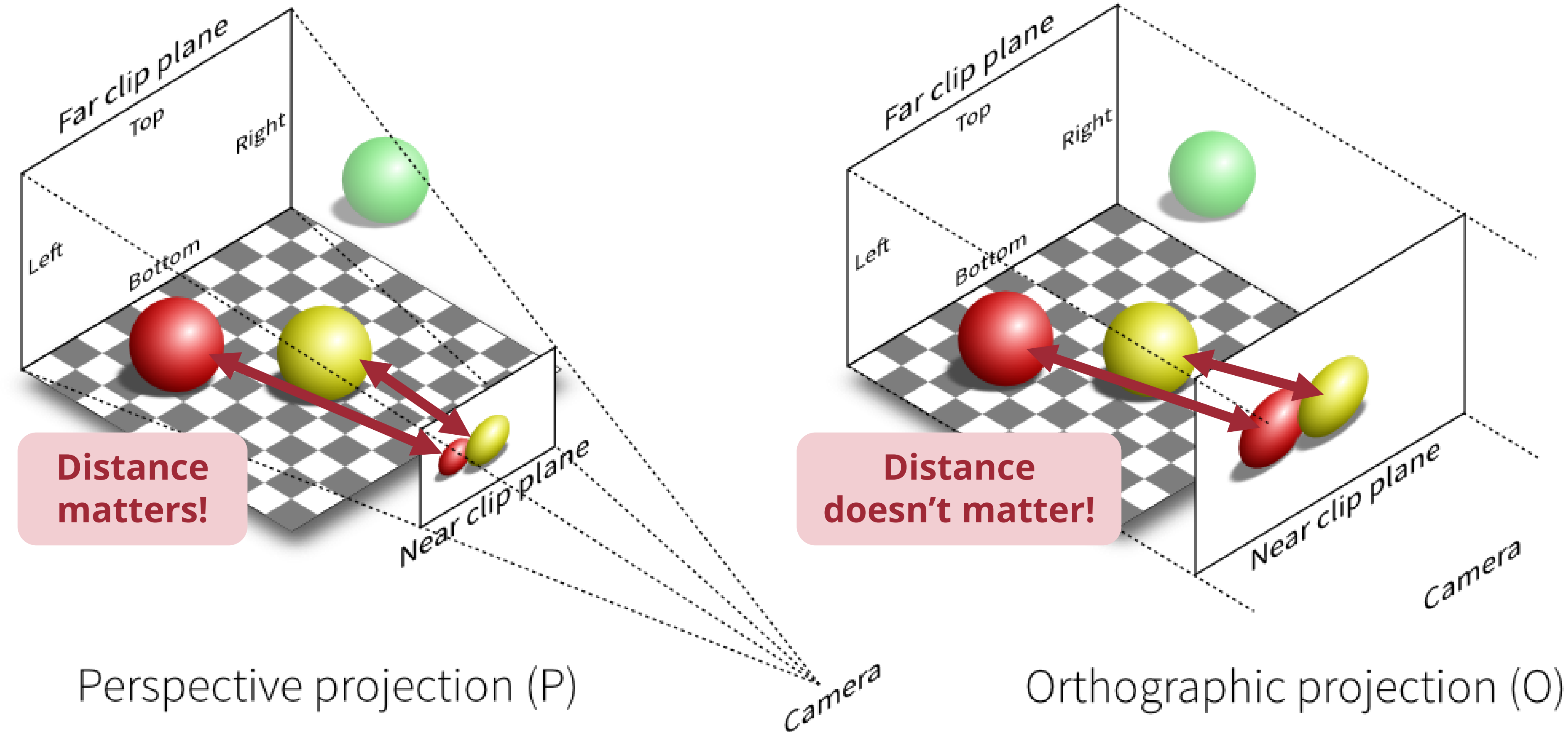
# Sphere Details

```
int res = 3;

void setup() {
  size(800, 800, P3D);
}

void draw() {
  background(200);
  fill(255);
  stroke(0);

  translate(400, 400, 0);
  rotateX(-1);

  sphereDetail(res);
  sphere(200);

  res += 1;
  if (res > 200) exit();
}
```



# Perspective vs Orthographic Projections

**perspective()** **ortho()**



**Distance matters!**

**Distance doesn't matter!**

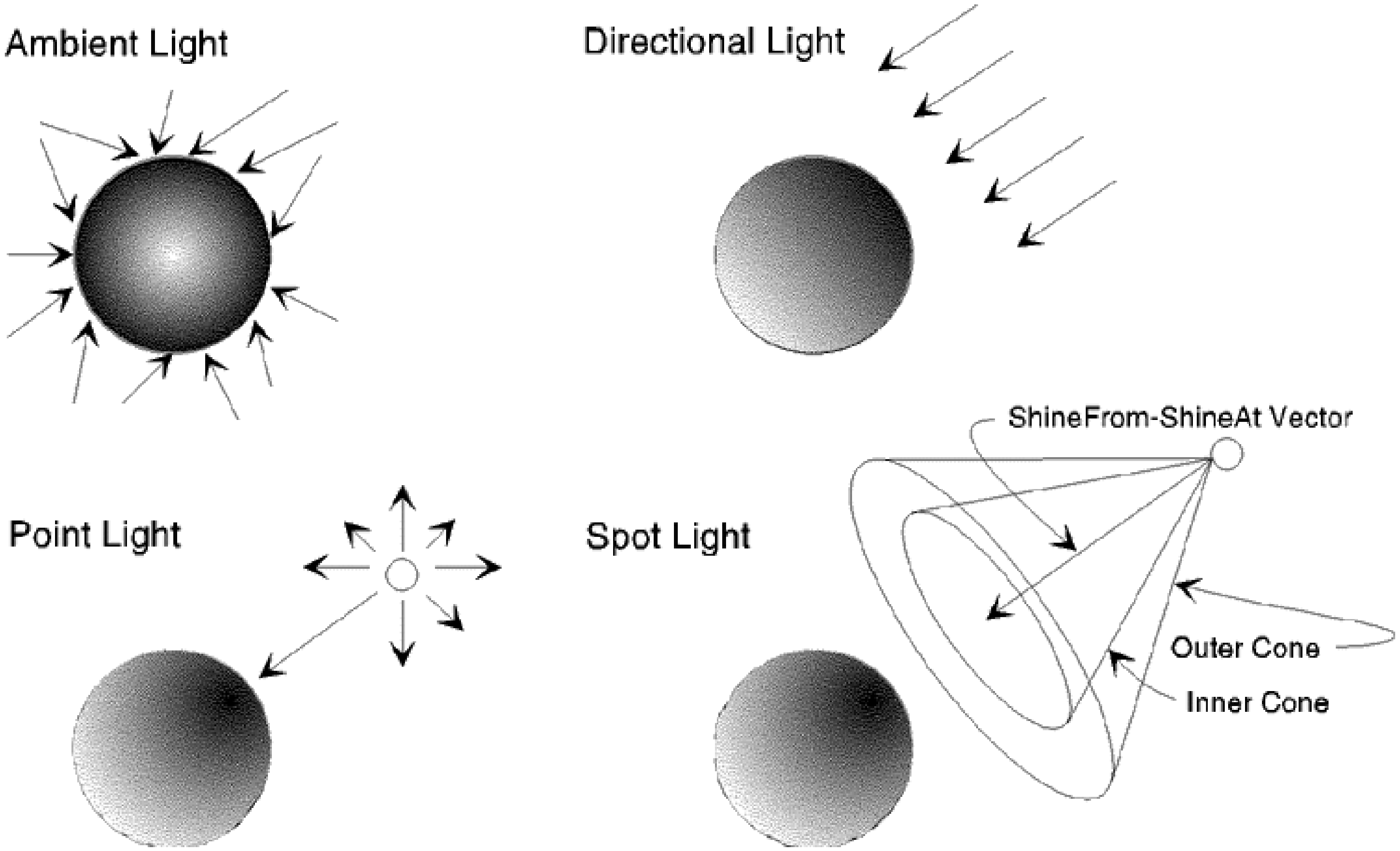
Perspective projection (P)

Orthographic projection (O)

glumpy.github.io/modern-gl.html

# Lights

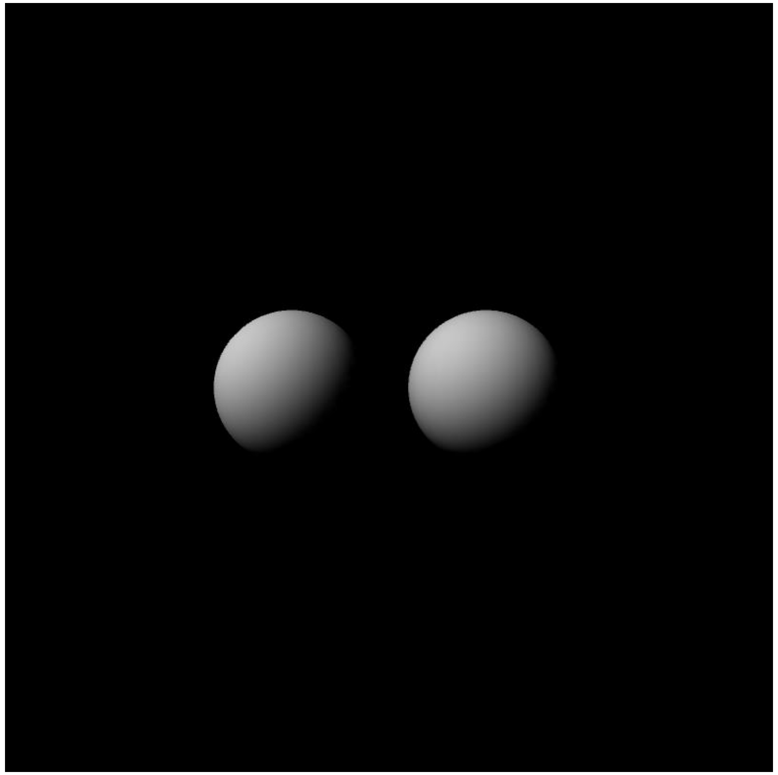
- `ambientLight()`
- `directionalLight()`
- `spotlight()`
- `pointLight()`



okino.com

# Example: Creepy Eyes 3D

```
void setup() {
  size(800, 800, P3D);
}

void draw() {
  background(0);

  float dirX = (mouseX - width / 2) / (width / 2.0);
  float dirY = (mouseY - height / 2) / (height / 2.0);
  directionalLight(200, 200, 200, -dirX, -dirY, -1);

  fill(255);
  noStroke();
  translate(300, 400, 0);
  sphere(80);
  translate(200, 0, 0);
  sphere(80);
}
```



# Material

- `ambient()` Set the ambient reflectance
- `emissive()` Set the emissive color
- `shininess()` Set the amount of gloss
- `specular()` Set the specular color