

PAT 204/504 (Fall 2025)

Creative Coding

Lecture 9: Transformations & 3D Graphics

Instructor: Hao-Wen Dong

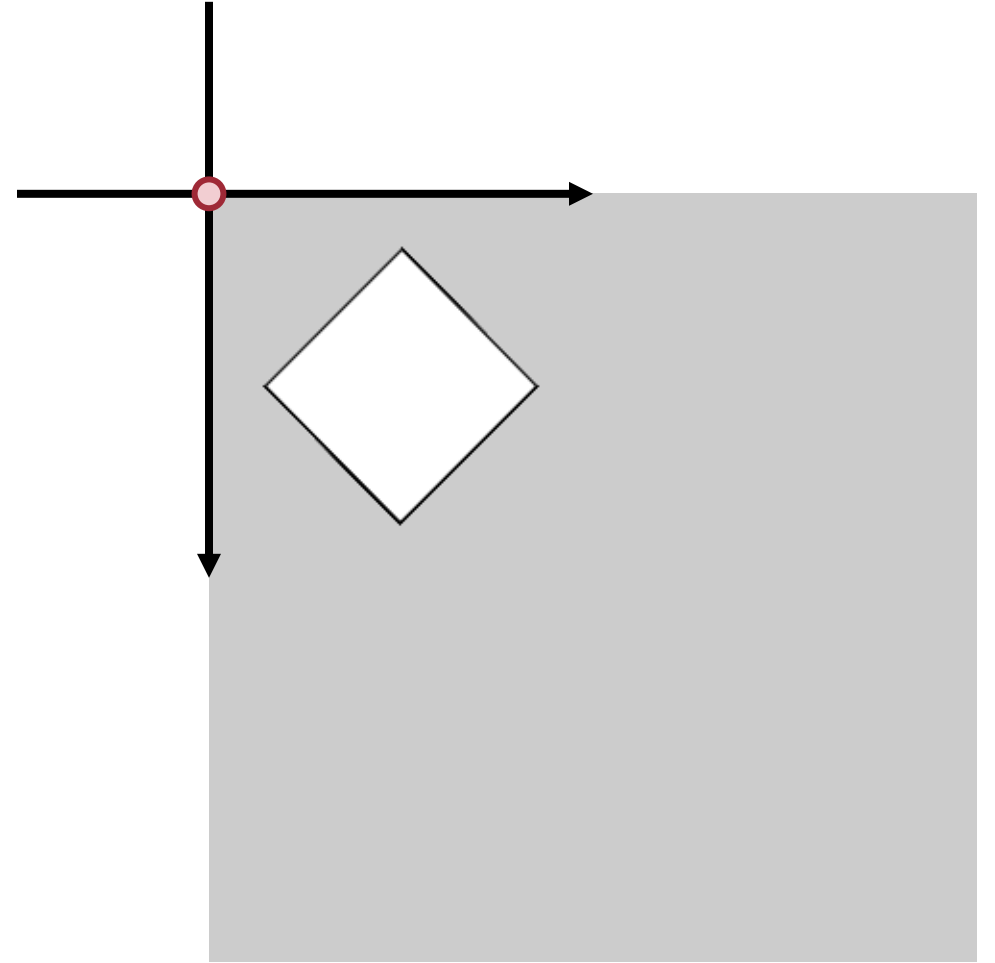
Transformations

| Transformations

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

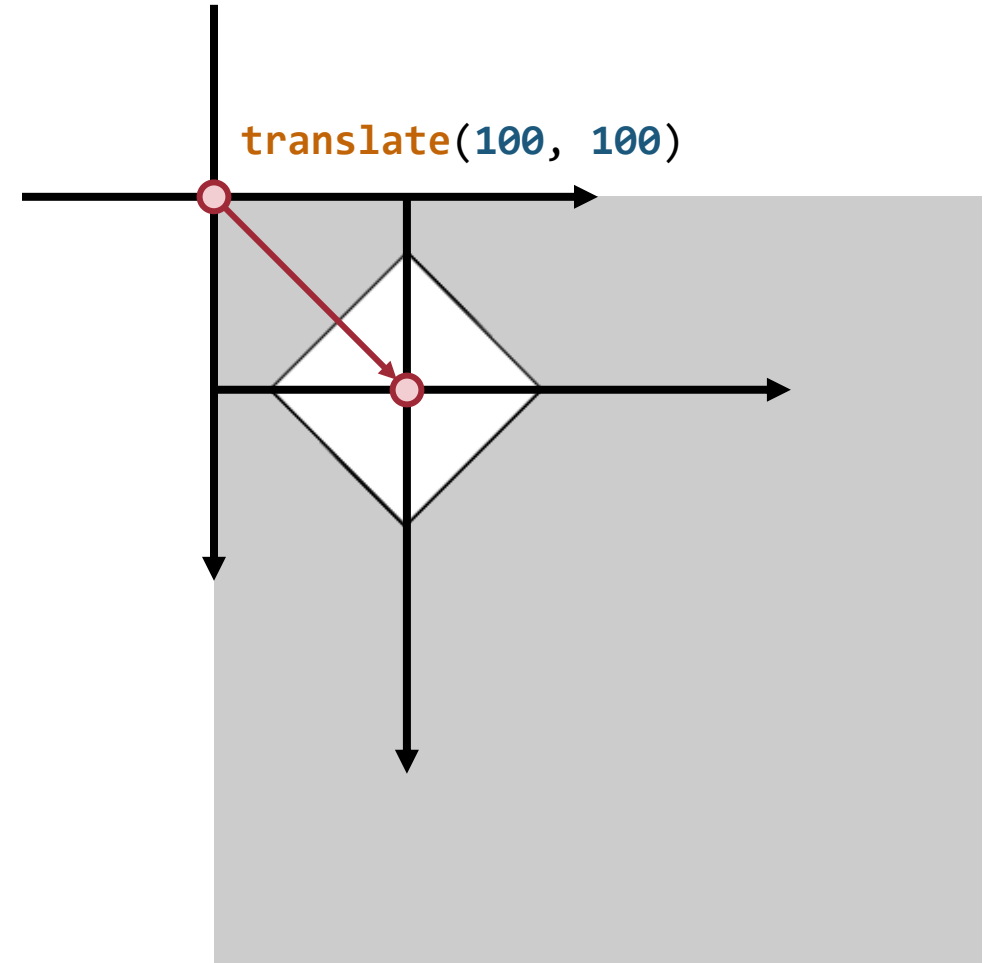
Example: Rotated Square

```
void setup() {  
  size(400, 400);  
}  
  
void draw() {  
  rectMode(CENTER);  
  
  translate(100, 100);  
  rotate(PI / 4);  
  square(0, 0, 100);  
}
```



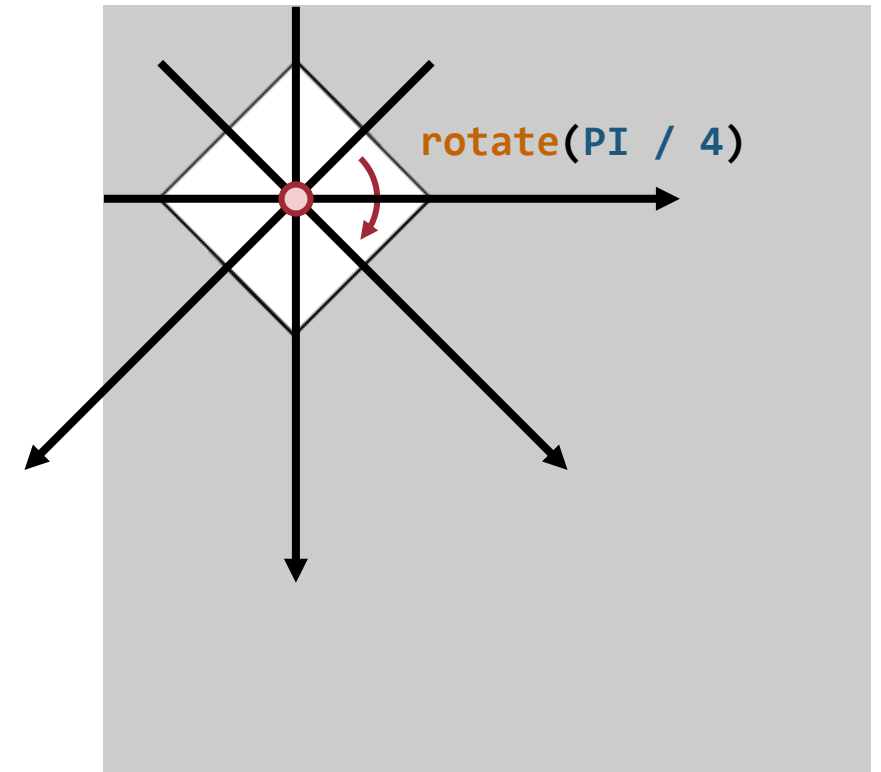
Example: Rotated Square

```
void setup() {  
  size(400, 400);  
}  
  
void draw() {  
  rectMode(CENTER);  
  
  translate(100, 100);  
  rotate(PI / 4);  
  square(0, 0, 100);  
}
```



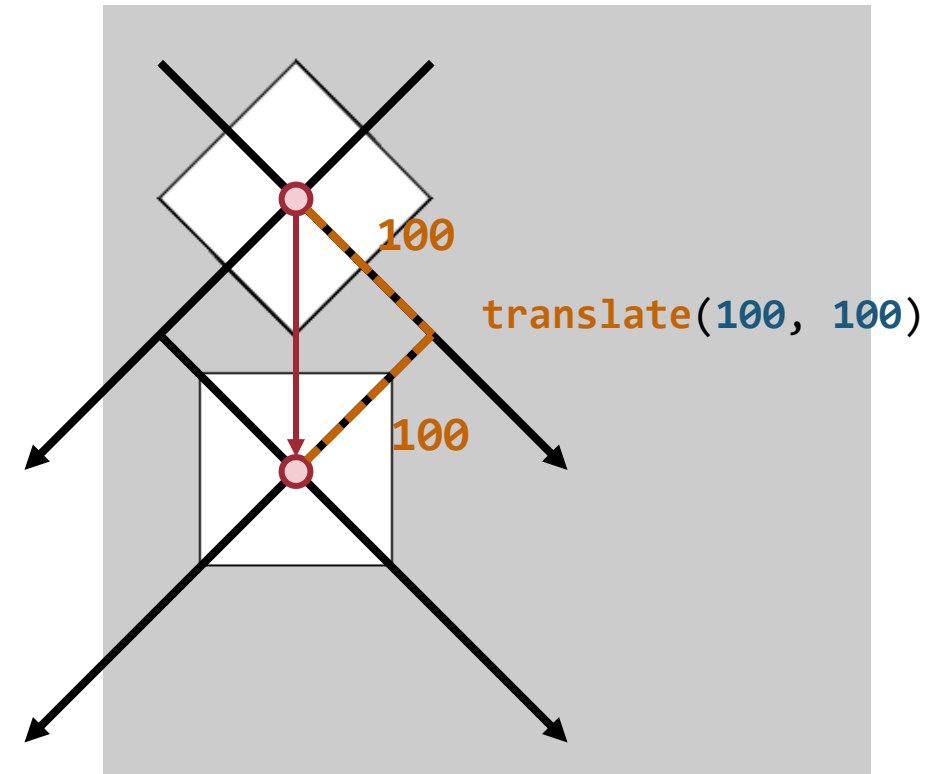
Example: Rotated Square

```
void setup() {  
  size(400, 400);  
}  
  
void draw() {  
  rectMode(CENTER);  
  
  translate(100, 100);  
  rotate(PI / 4);  
  square(0, 0, 100);  
}
```



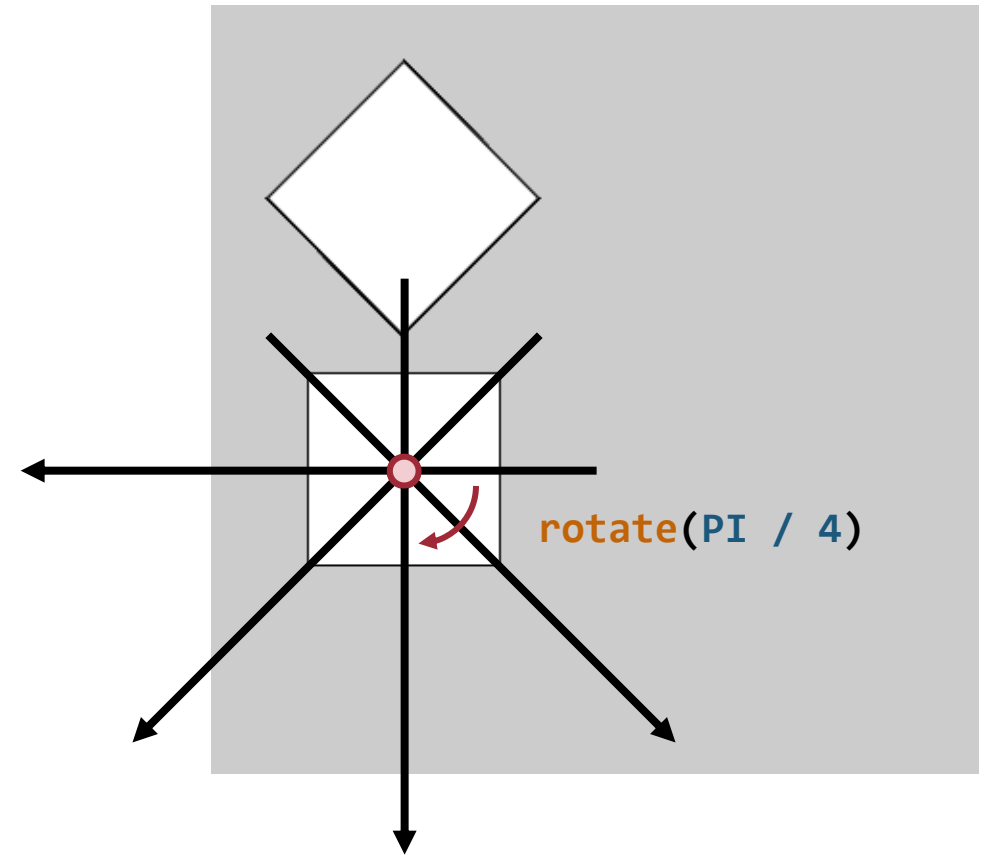
Example: Rotated Squares

```
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);  
}
```



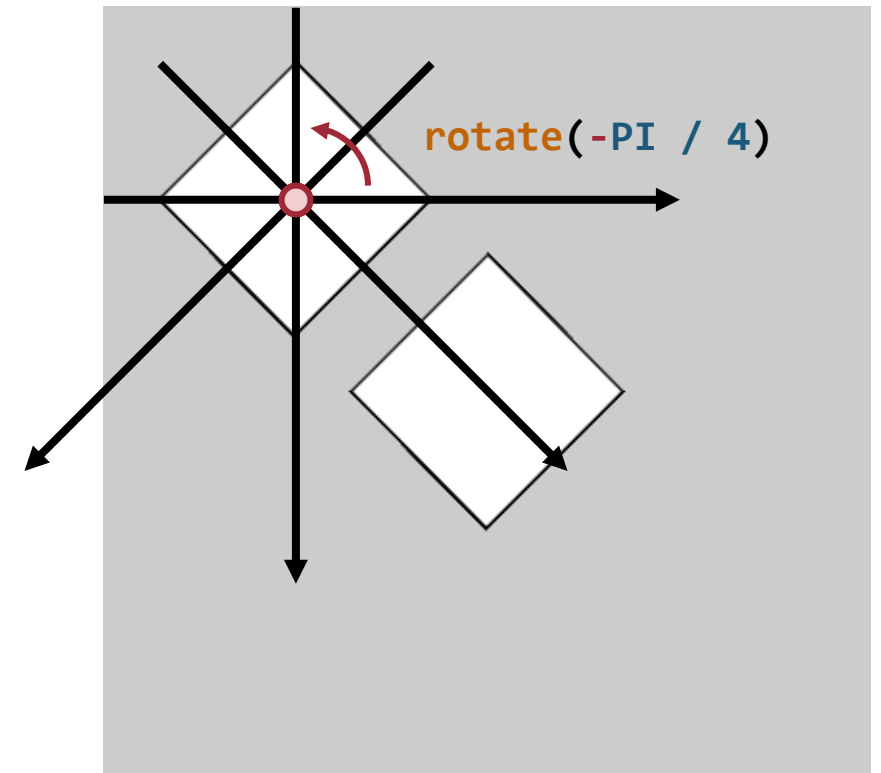
Example: Rotated Squares

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



Example: Rotated Squares

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

```
void draw() {  
  rectMode(CENTER);
```

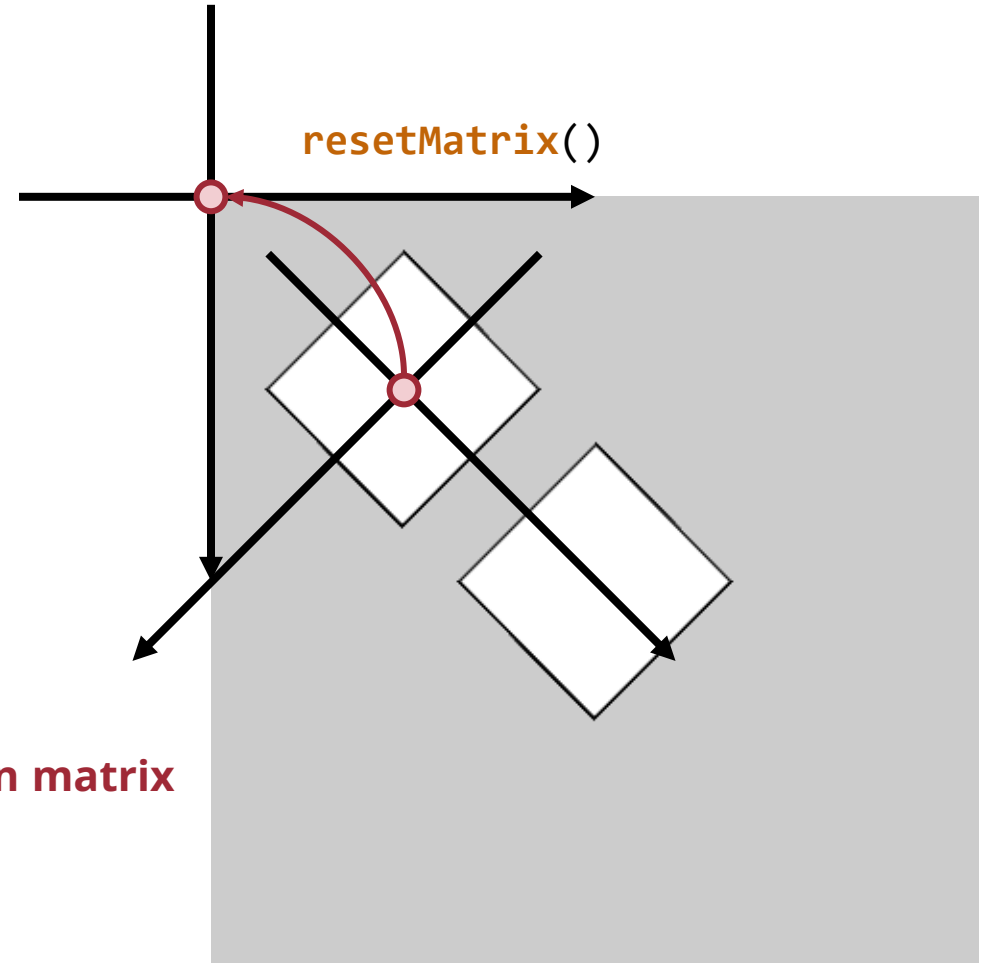
```
  translate(100, 100);  
  rotate(PI / 4);  
  square(0, 0, 100);
```

```
  resetMatrix(); Reset the transformation matrix
```

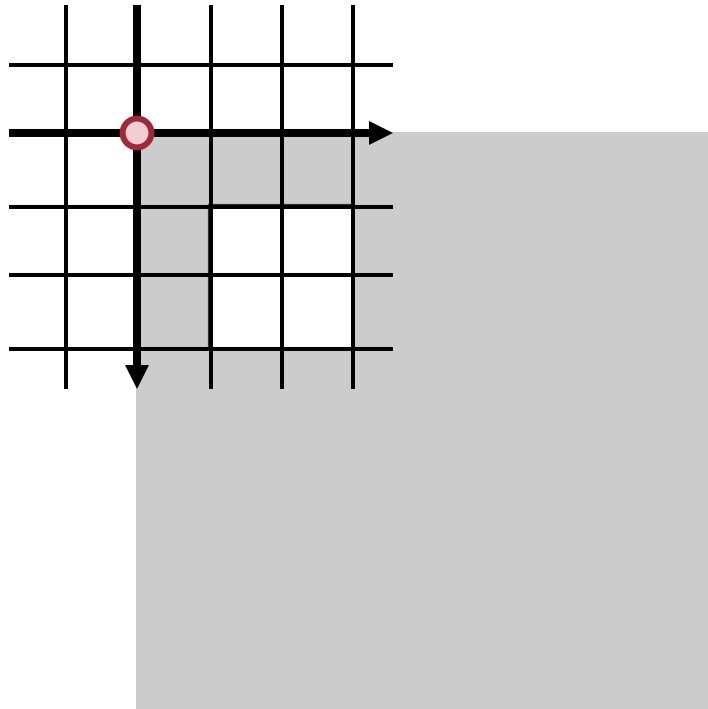
```
  translate(200, 200);
```

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

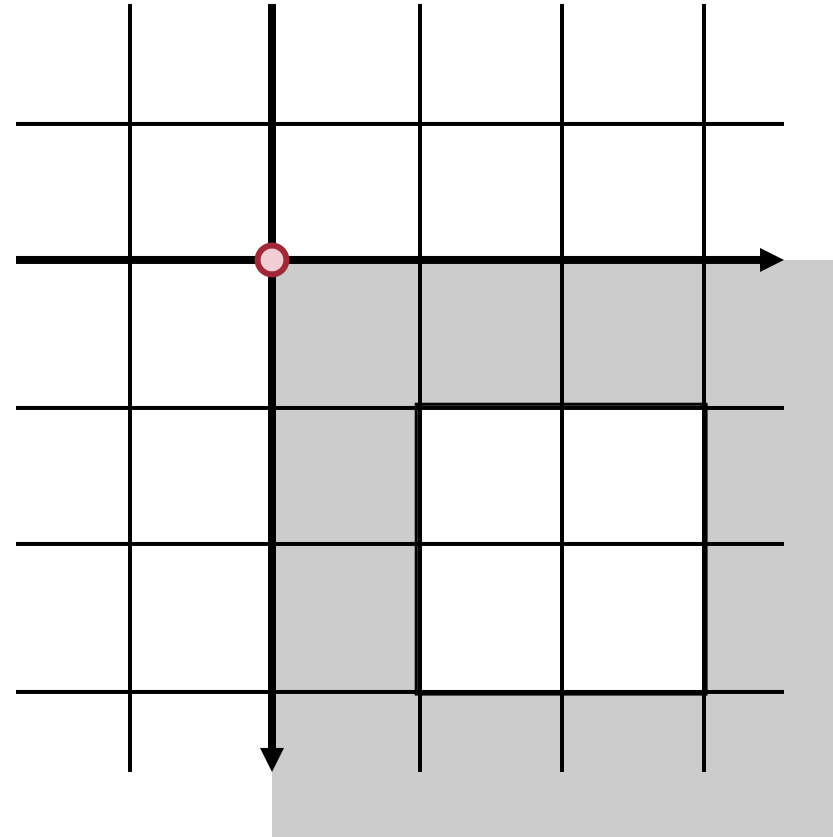
```
}
```



Transformation Matrix

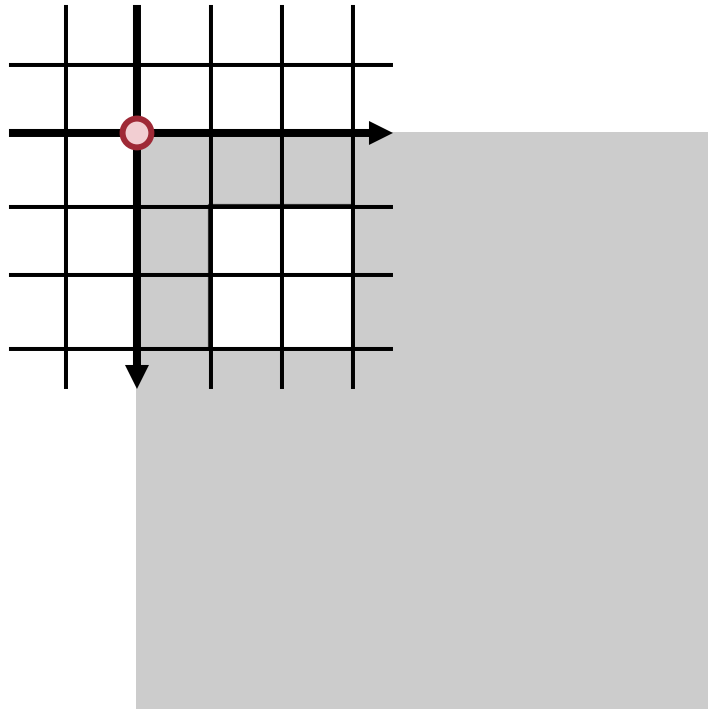


```
square(100, 100, 100);
```

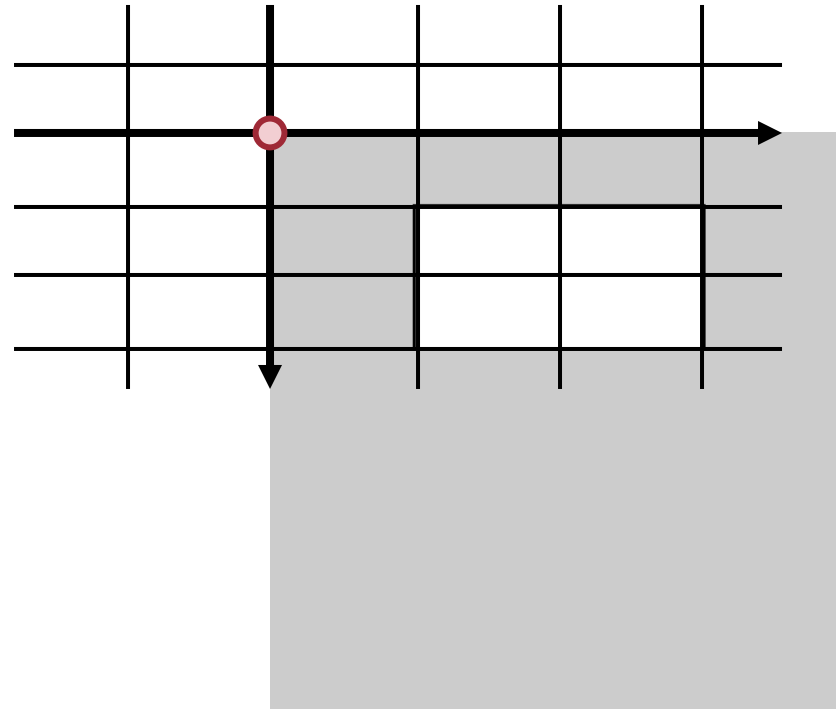


```
scale(2);  
square(100, 100, 100);
```

Transformation Matrix



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



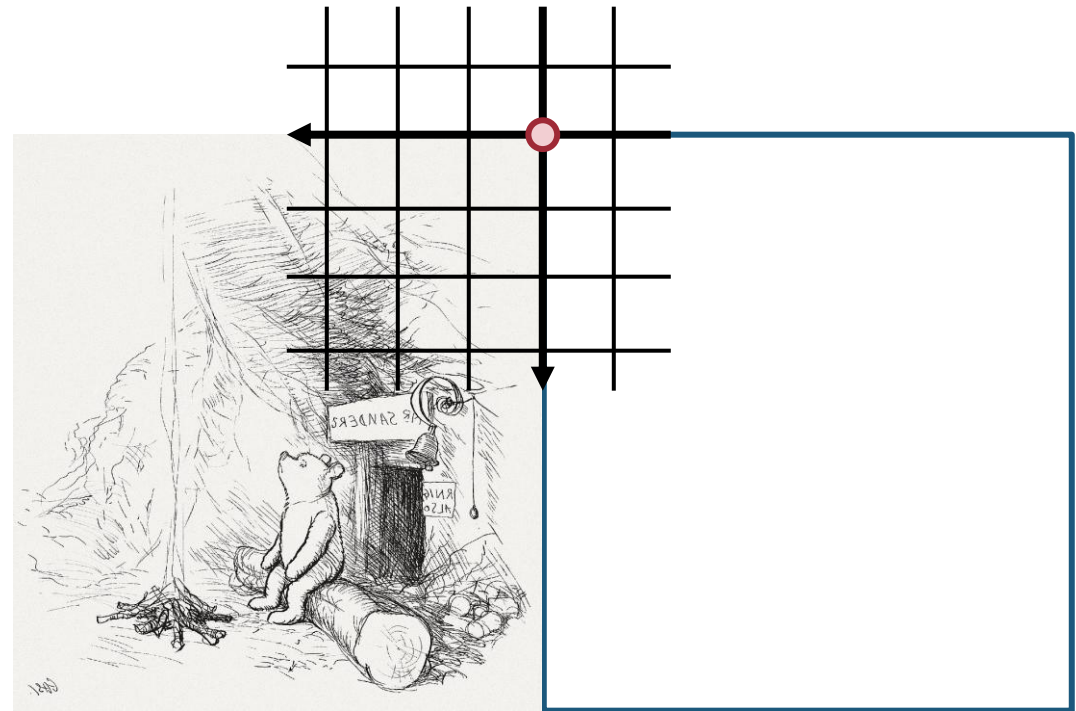
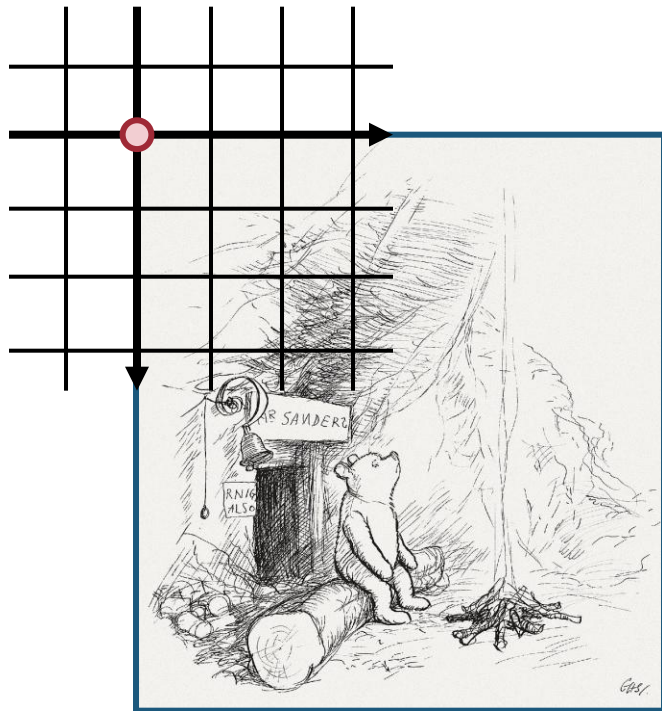
```
scale(2, 1);  
square(100, 100, 100);
```

Example: Mirroring an Image

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



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

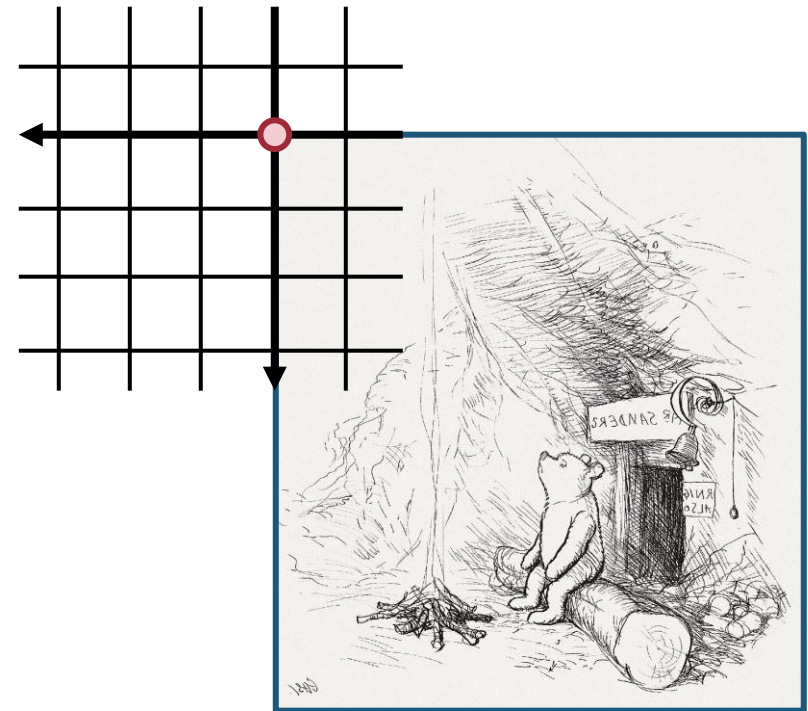
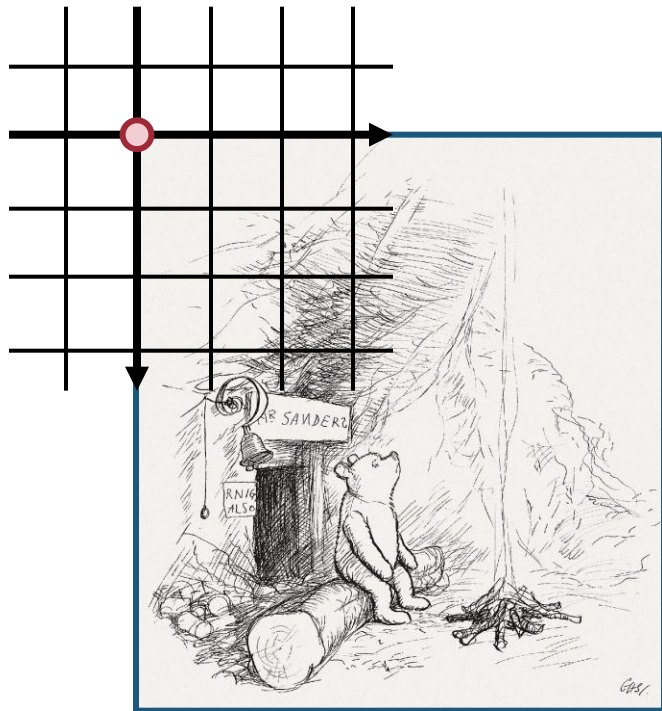


Example: Mirroring an Image

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



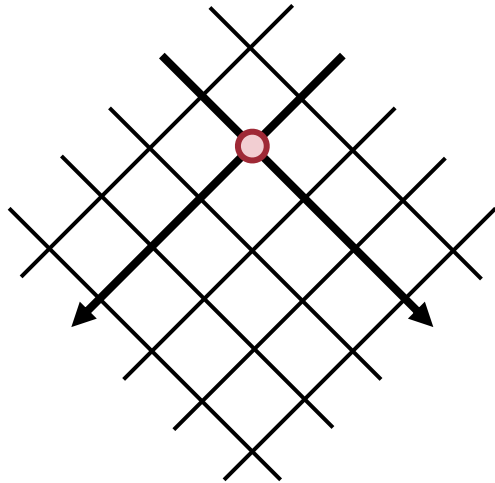
```
void draw() {  
  scale(-1, 1);  
  image(video, -video.width, 0);  
}
```



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

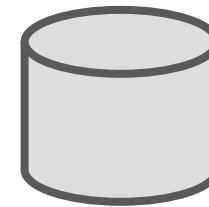
Transformation matrix



pushMatrix()

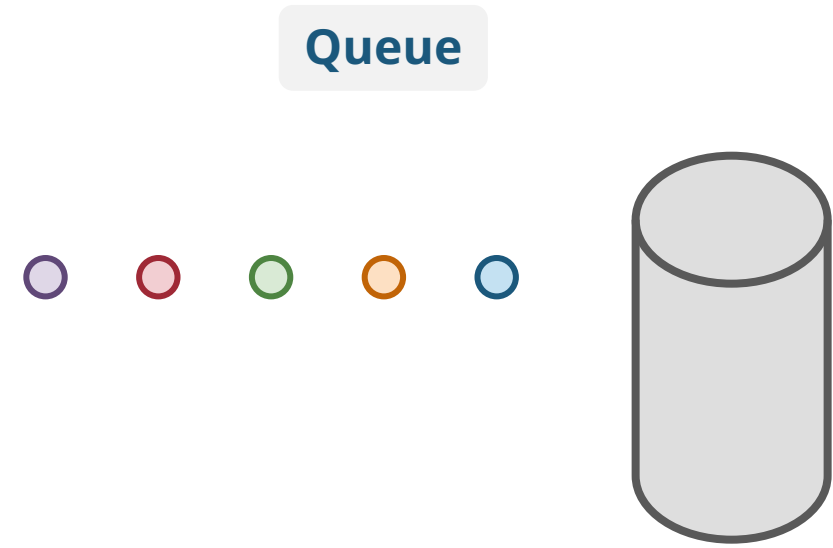
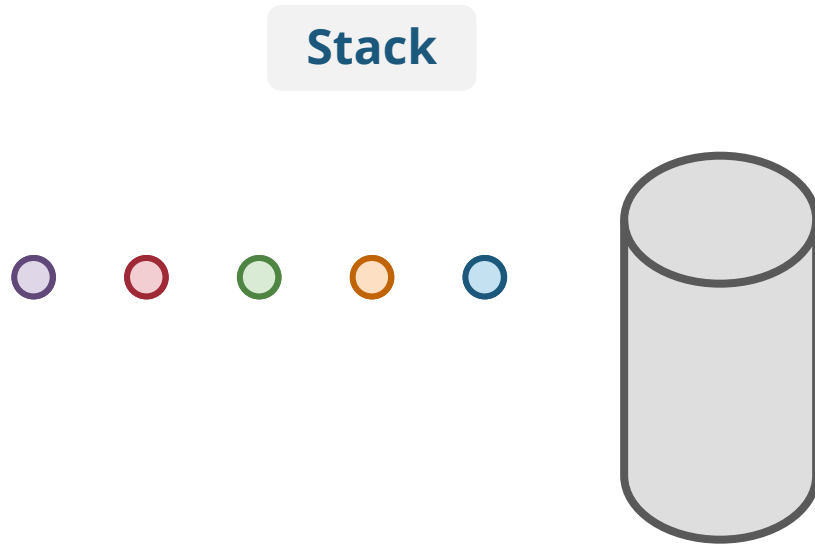
popMatrix()

Matrix stack

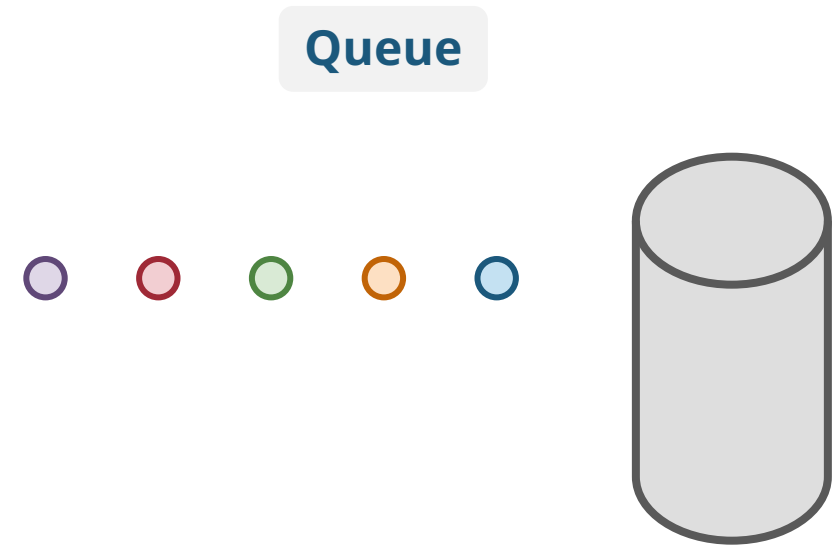
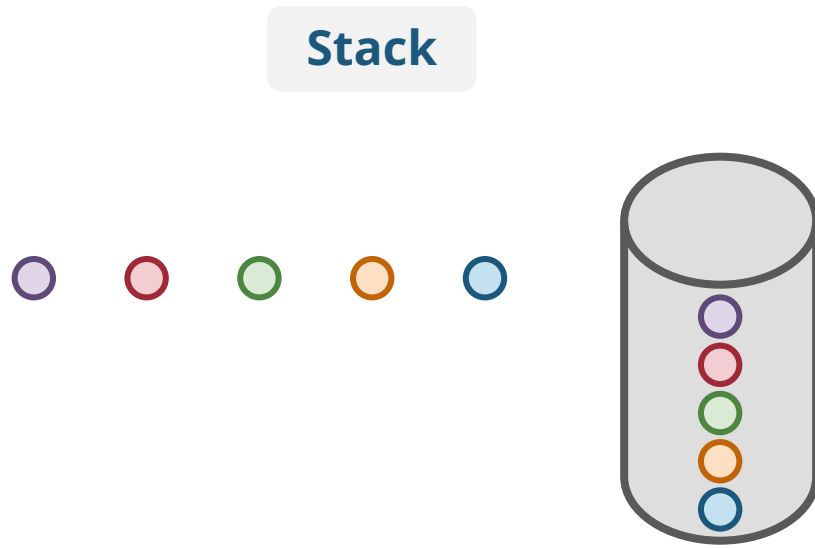


First in, last out!

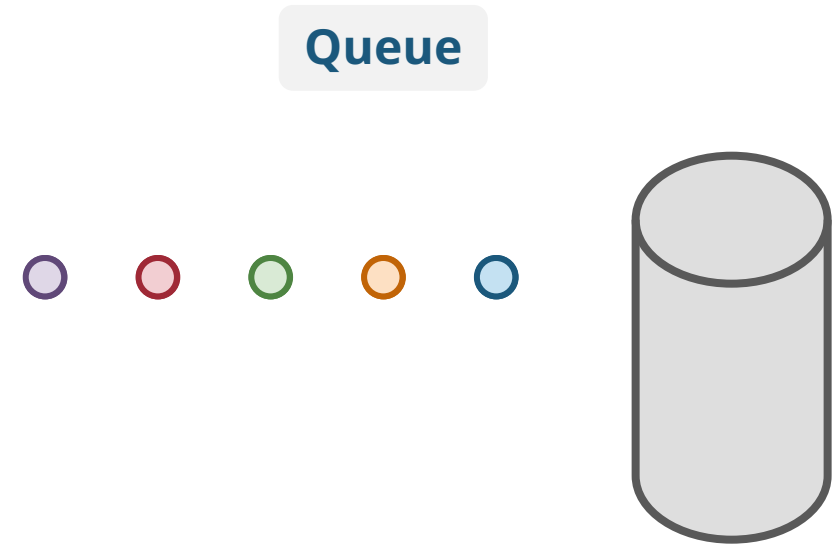
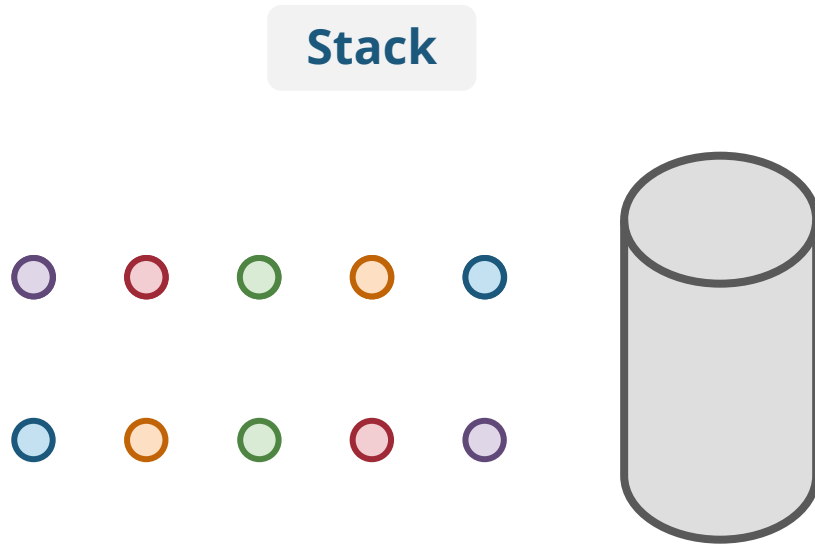
| Stack vs Queue



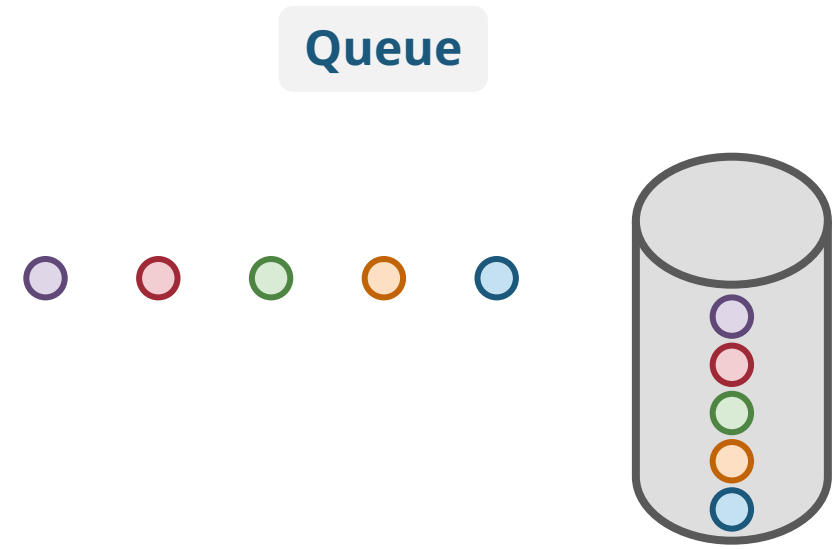
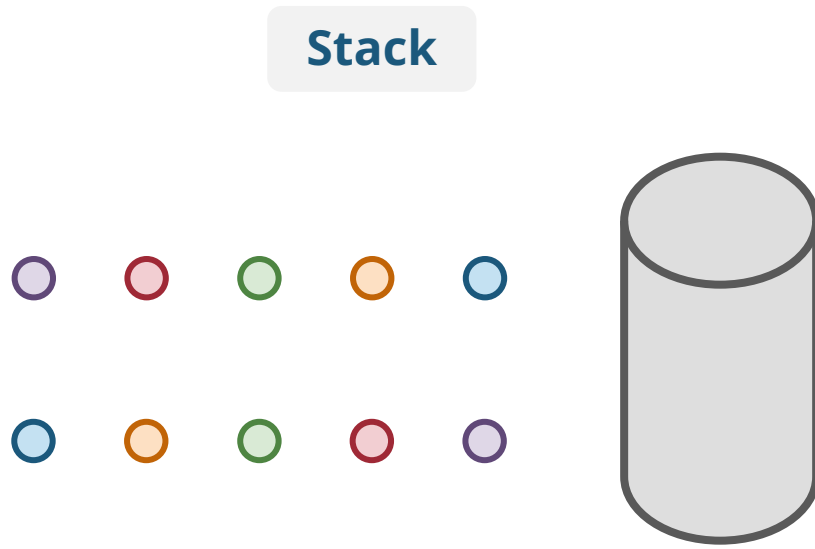
| Stack vs Queue



| Stack vs Queue

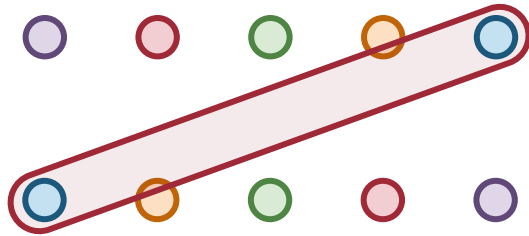


| Stack vs Queue



| Stack vs Queue

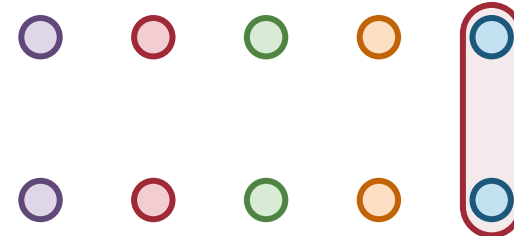
Stack



First in last out



Queue



First in first out



Example: Spinning Objects

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

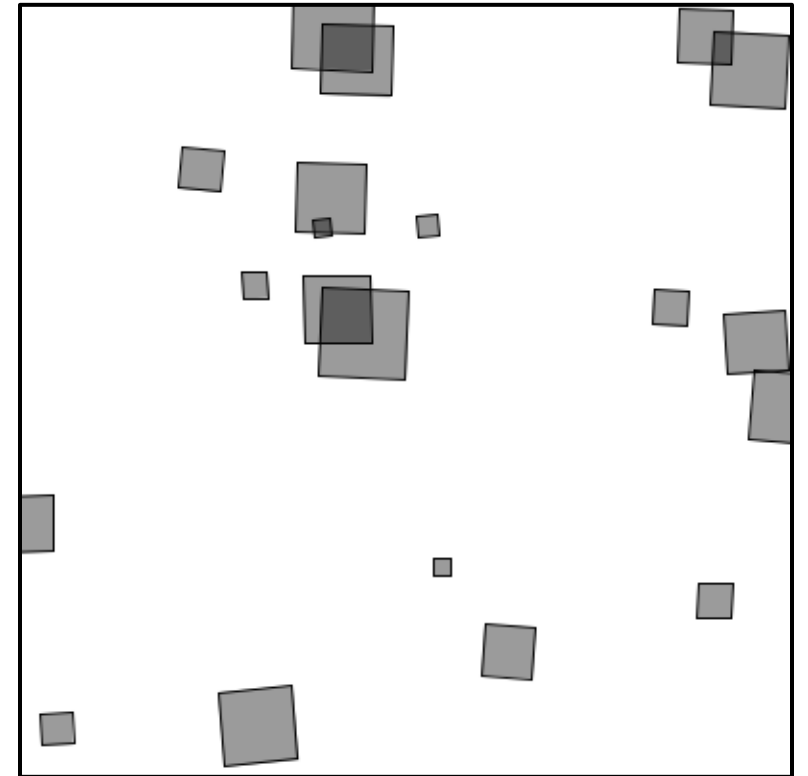
Declare an array of
rotater objects

```
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);  
  }  
}
```

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

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

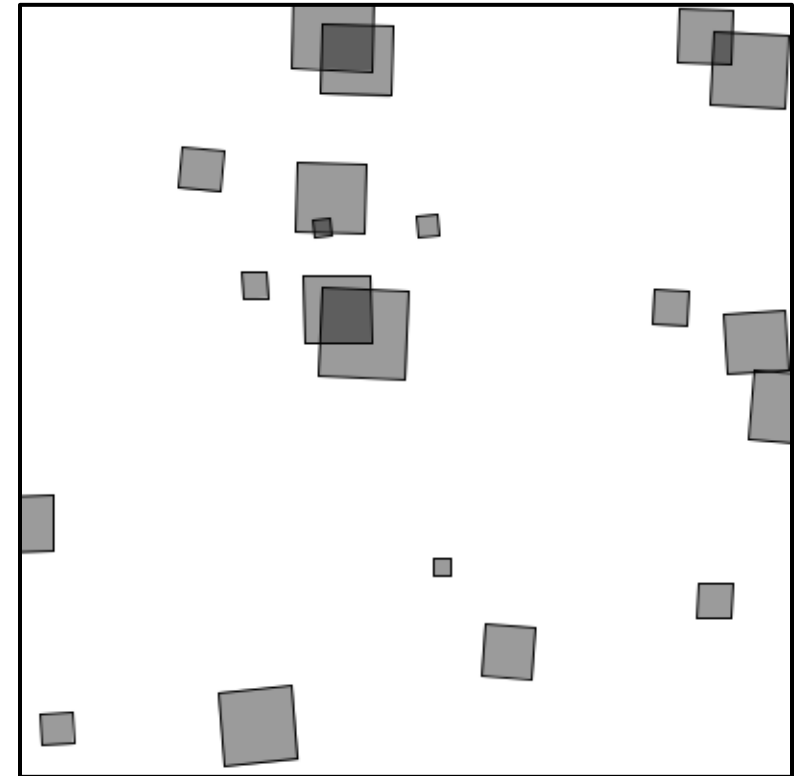
Spin and show
the rotaters!



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(); Store the current matrix
```

```
        translate(x, y);
```

```
        rotate(theta);
```

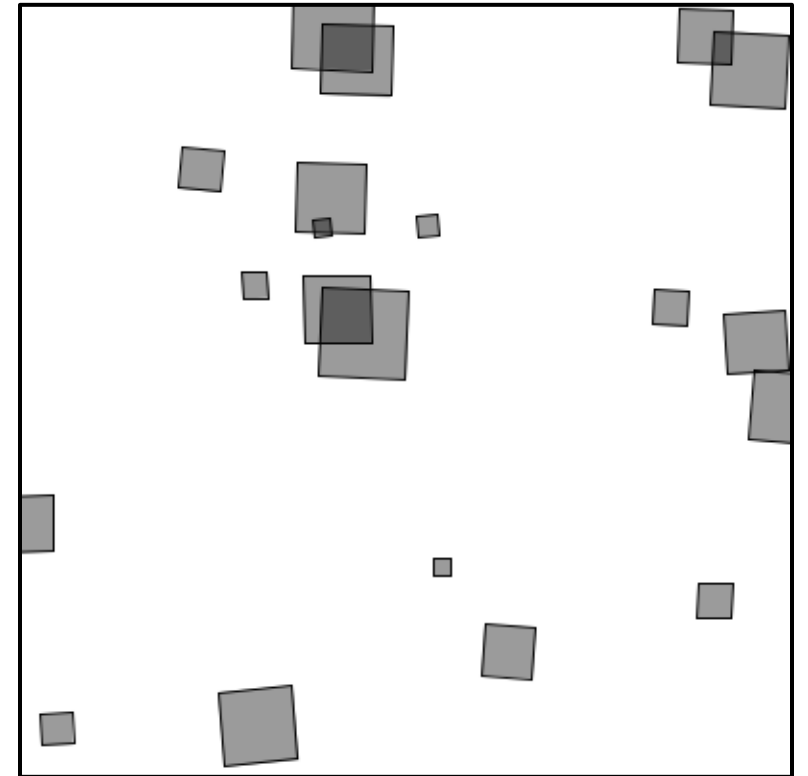
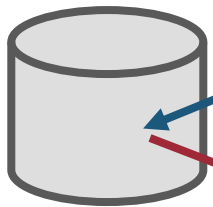
```
        rect(0, 0, w, w);
```

```
        popMatrix(); Restore the stored matrix
```

```
    }
```

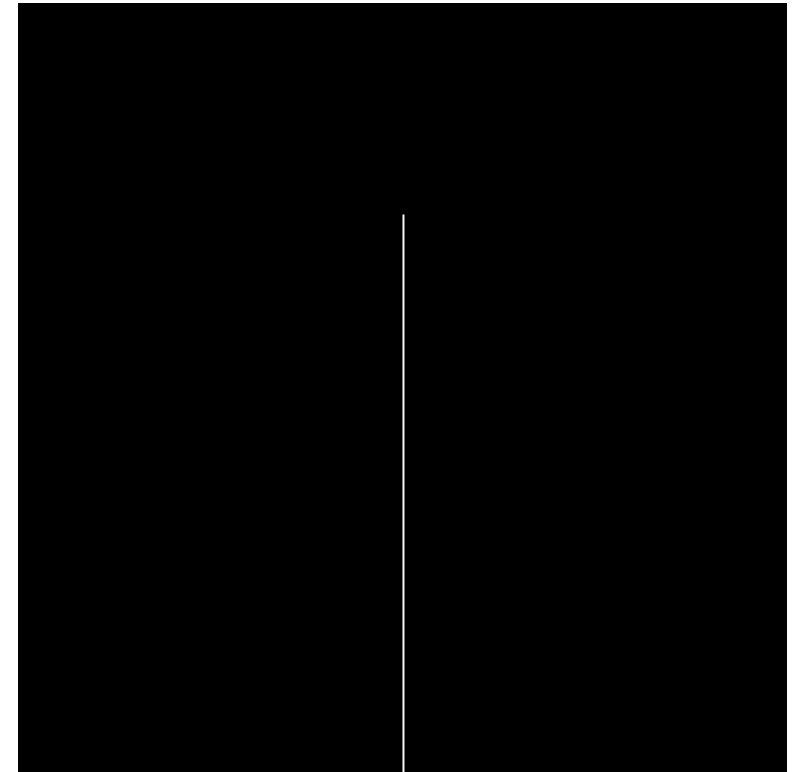
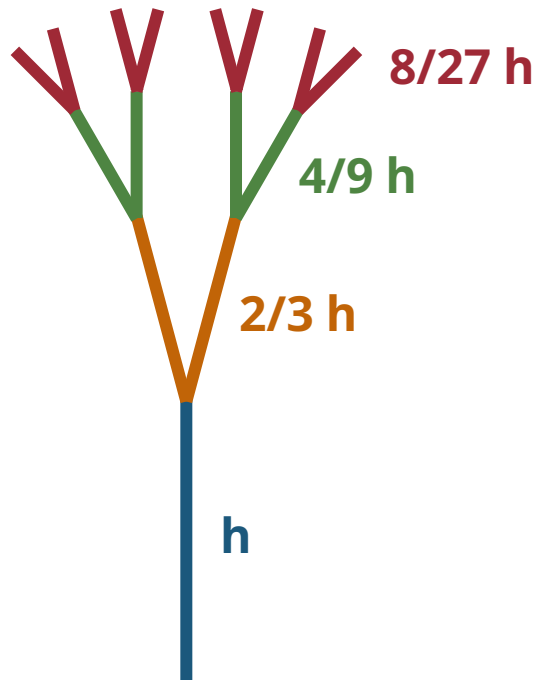
```
}
```

Matrix
stack



Example: Recursive Tree

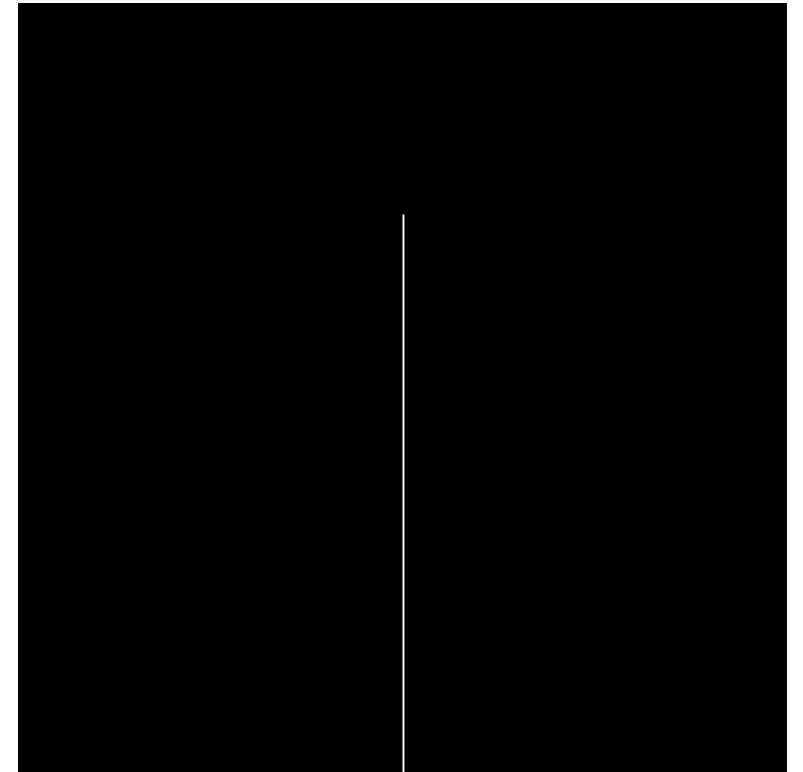
- Symmetric branches of $2/3$ length of its root
 - One branch is rotated counterclockwise for a fixed angle
 - The other branch is rotated clockwise for a fixed angle



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();  
}
```

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);
```

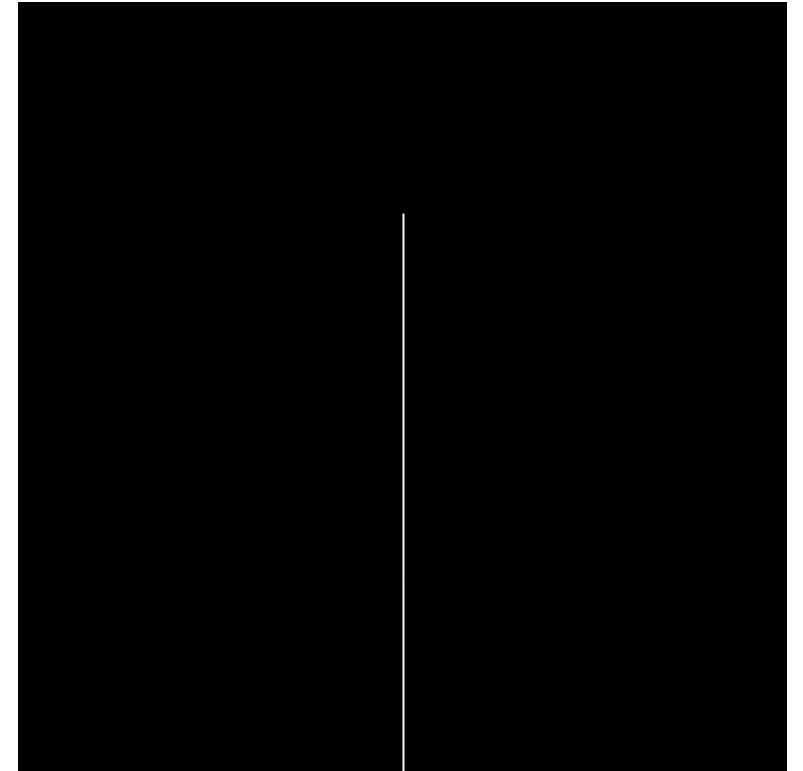
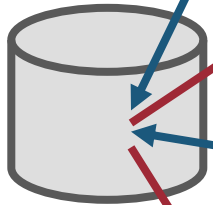
```
    translate(0, -h * scale);
```

```
    branch(h * scale);
```

```
    popMatrix();
```

```
}
```

Matrix
stack



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();  
}
```

Why not?

```
void branch(float h) {  
    if (h < 2) break;  
  
    // Right branch  
    resetMatrix();  
    rotate(theta);  
    line(0, 0, 0, -h * scale);  
    translate(0, -h * scale);  
    branch(h * scale);  
  
    // Left branch  
    resetMatrix();  
    rotate(-theta);  
    line(0, 0, 0, -h * scale);  
    translate(0, -h * scale);  
    branch(h * scale);  
}
```

Why not?

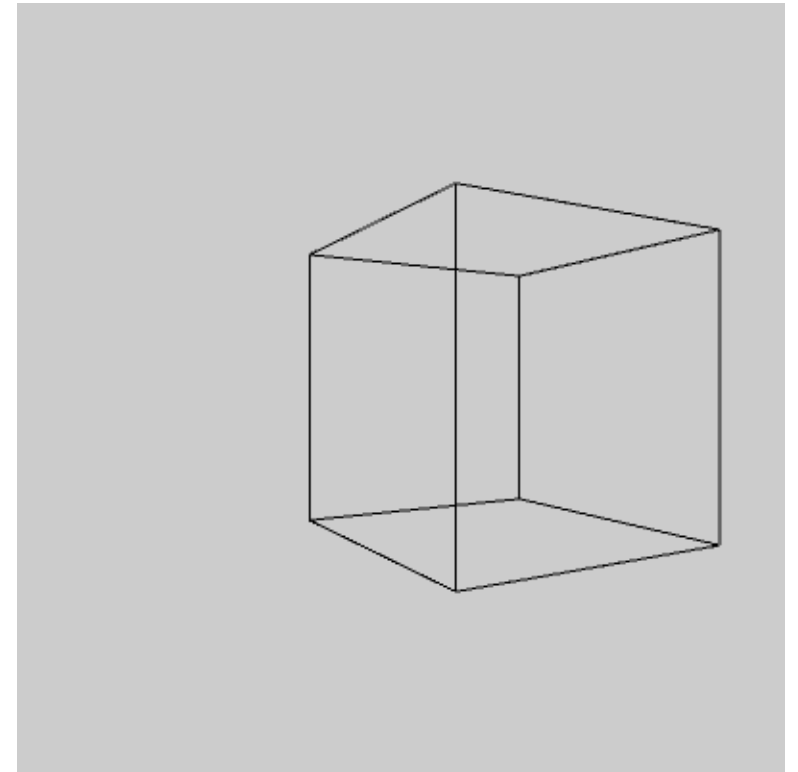
3D Graphics

| 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!**

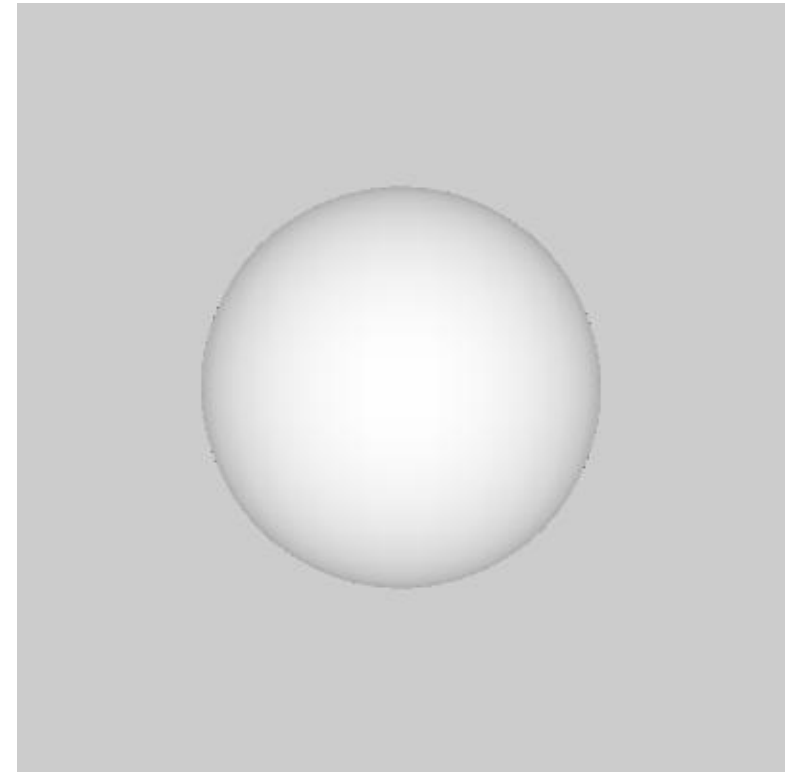
| Example: Box

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



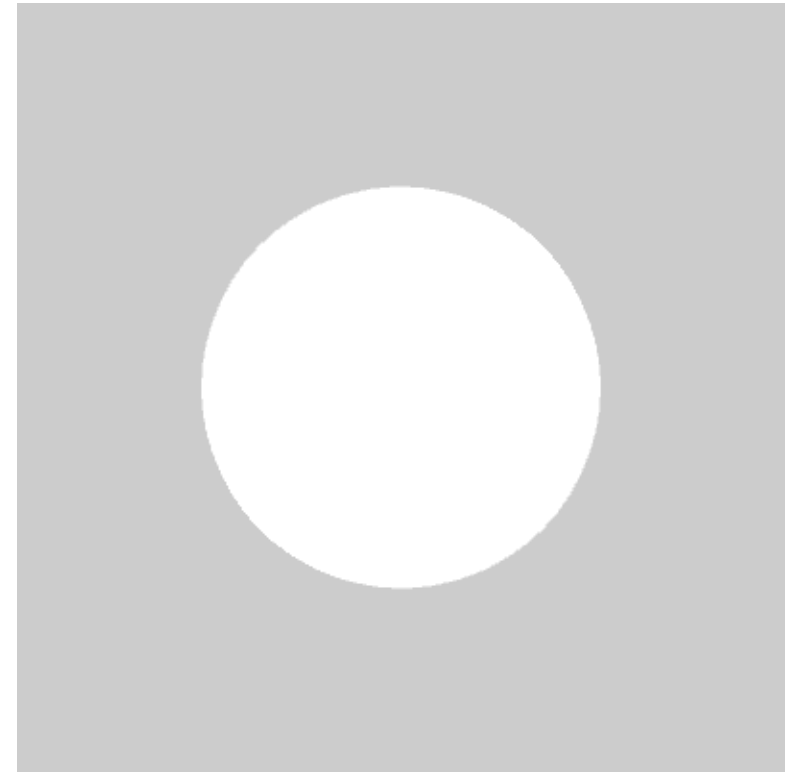
| Example: Sphere

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



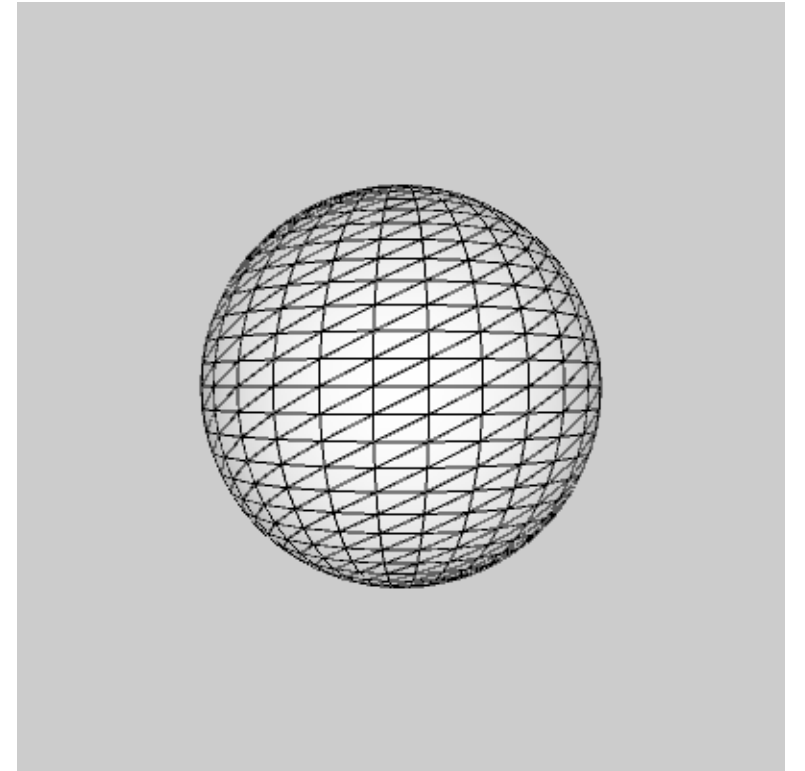
| Example: Sphere

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



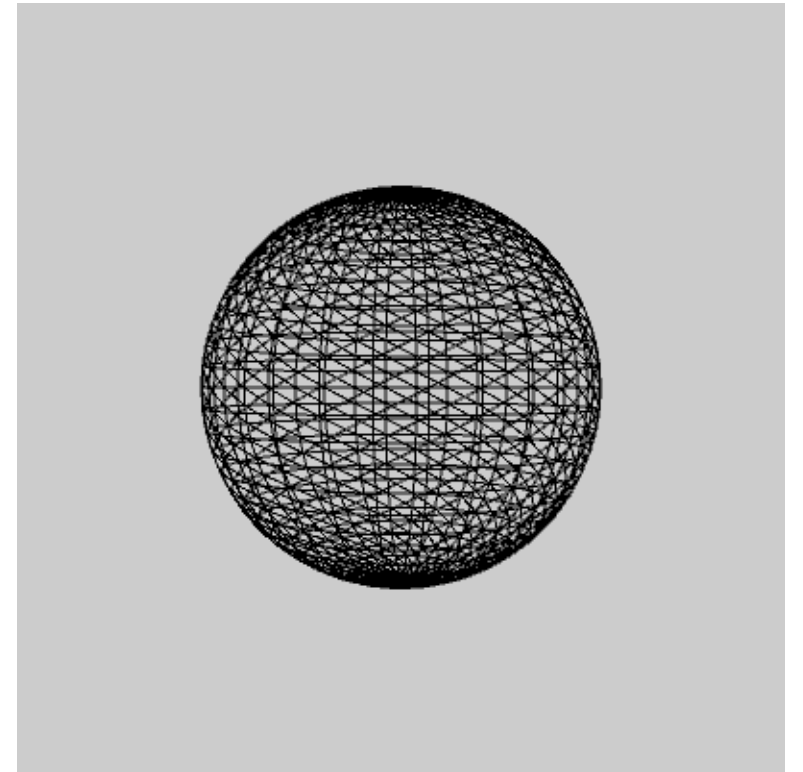
| Example: Sphere

```
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;

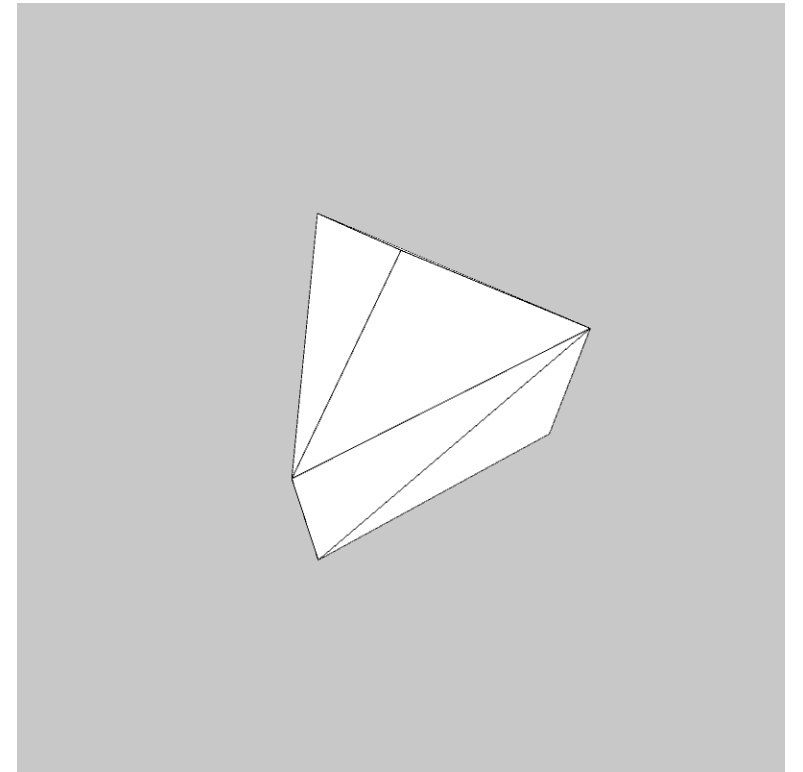
void setup() {
  size(800, 800, P3D);
}

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

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

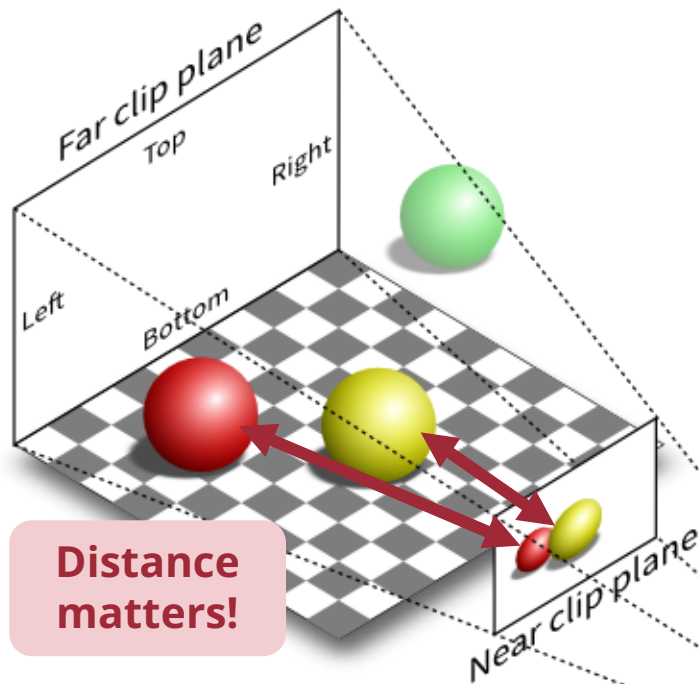
  sphereDetail(res);
  sphere(200);

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



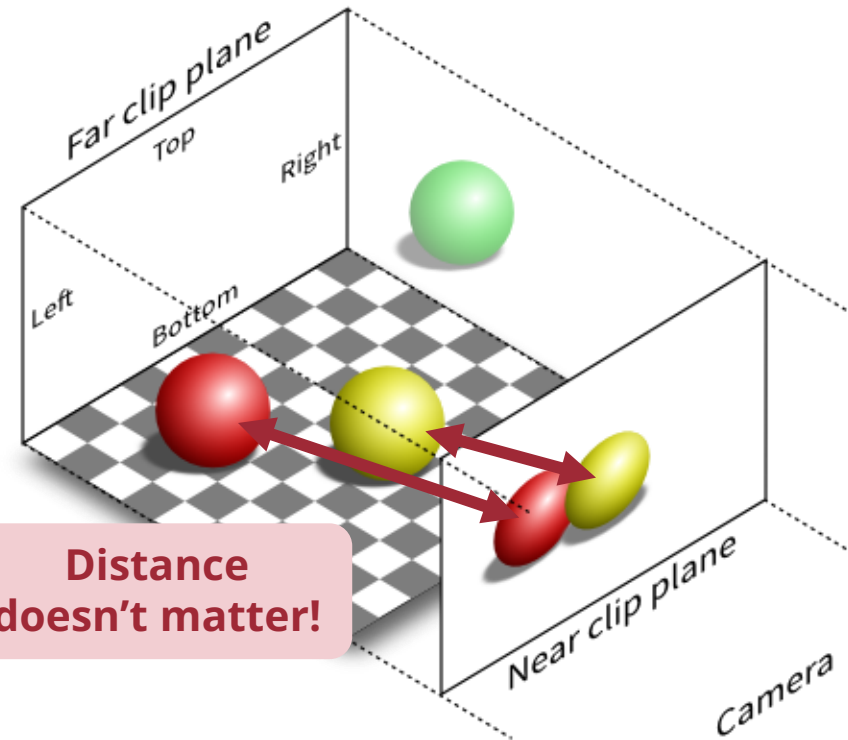
Perspective vs Orthographic Projections

perspective()



Perspective projection (P)

ortho()

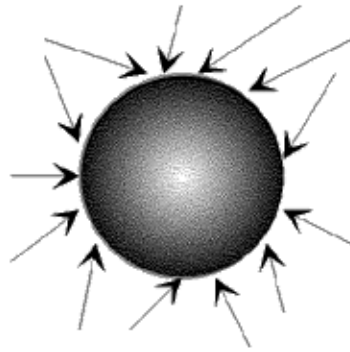


Orthographic projection (O)

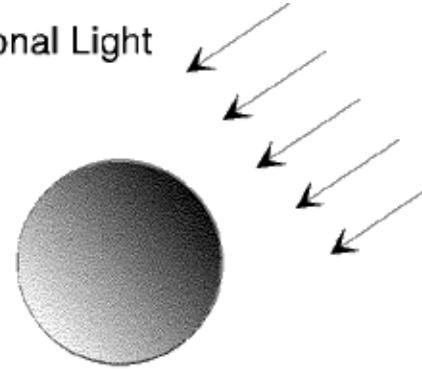
Lights

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

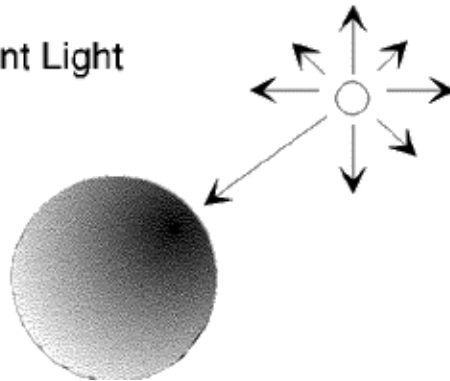
Ambient Light



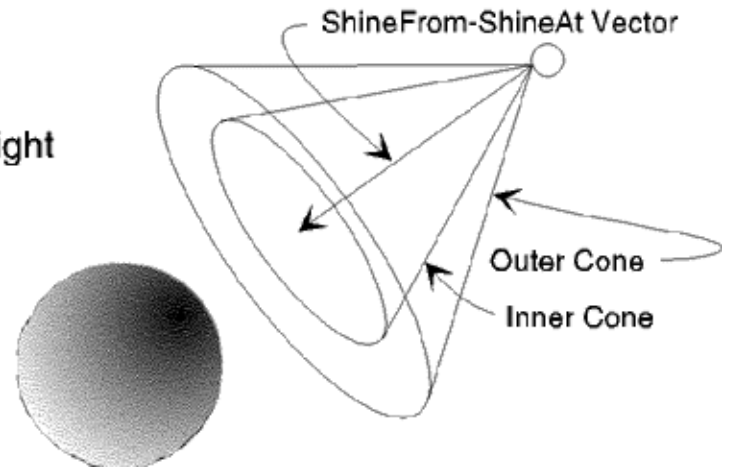
Directional Light



Point Light

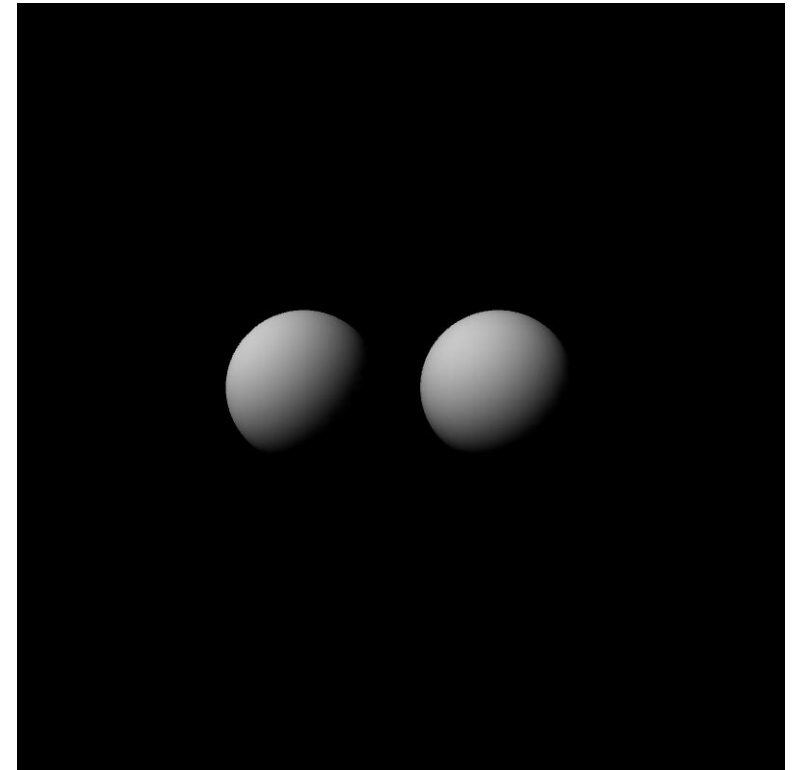


Spot Light



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);  
}
```



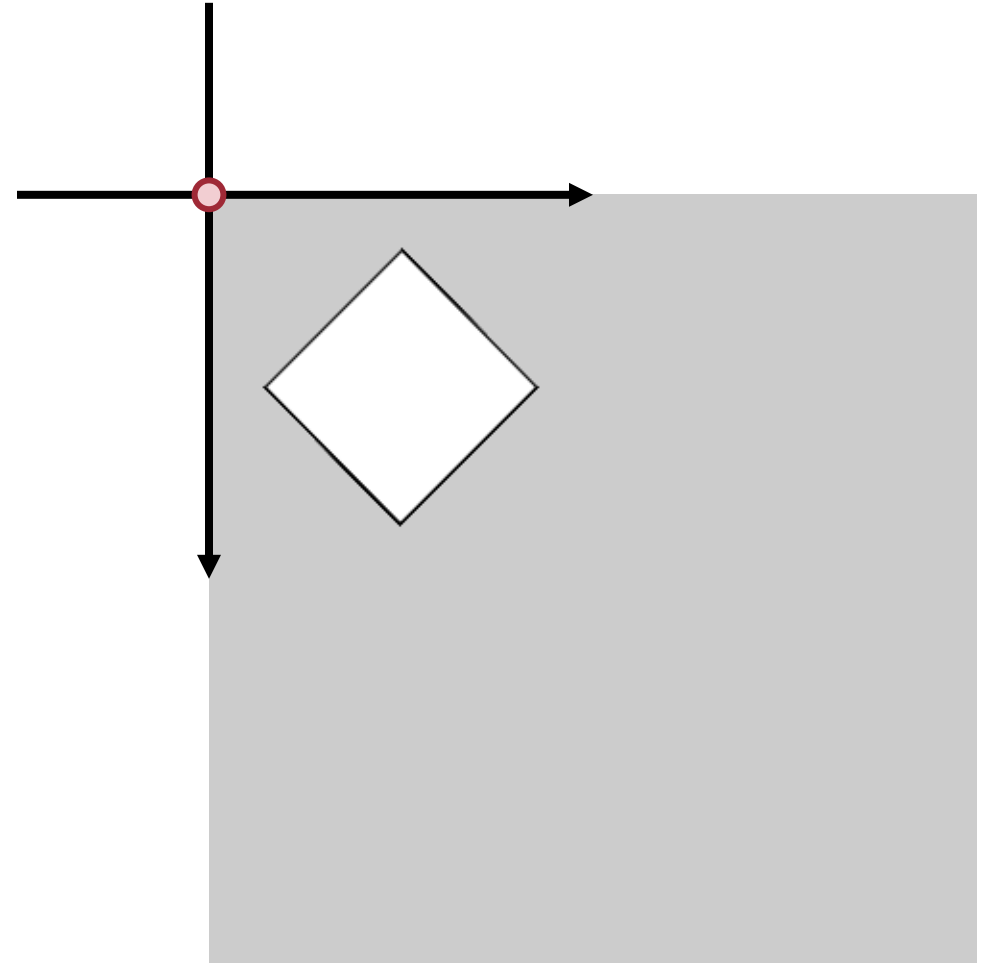
| Material

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

Recap

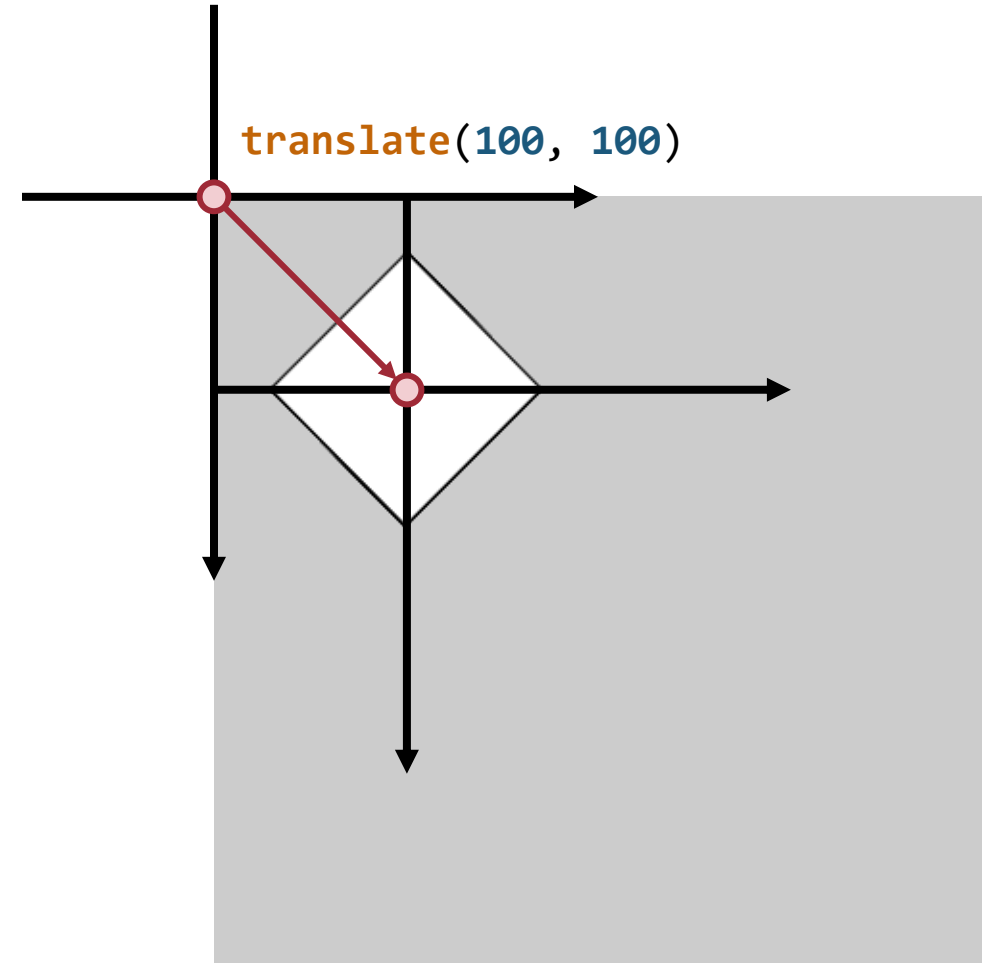
Example: Rotated Square

```
void setup() {  
  size(400, 400);  
}  
  
void draw() {  
  rectMode(CENTER);  
  
  translate(100, 100);  
  rotate(PI / 4);  
  square(0, 0, 100);  
}
```



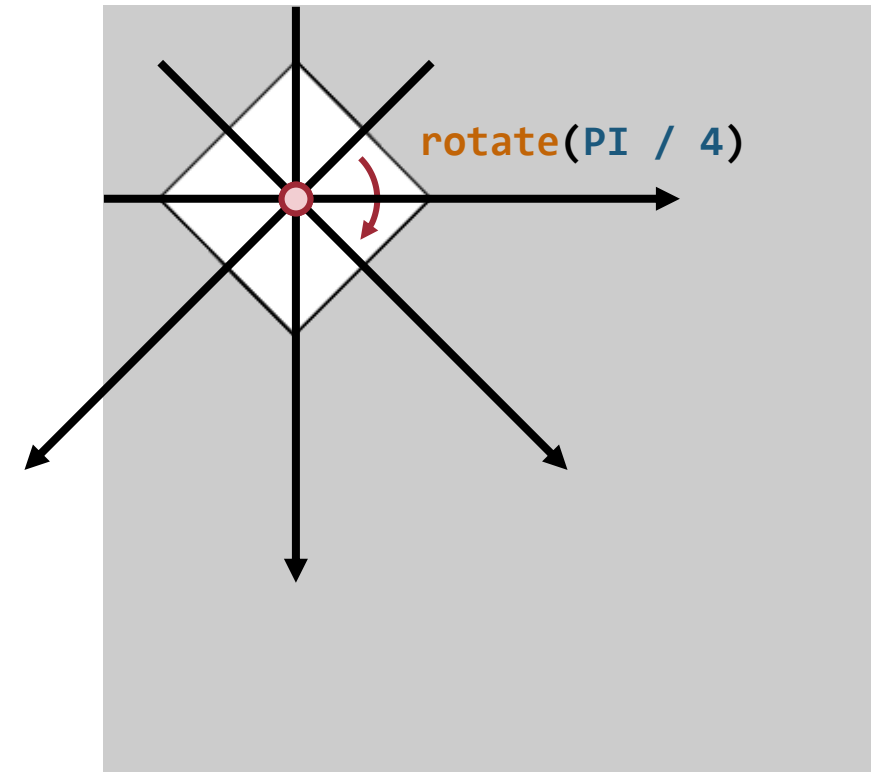
Example: Rotated Square

```
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);  
}
```

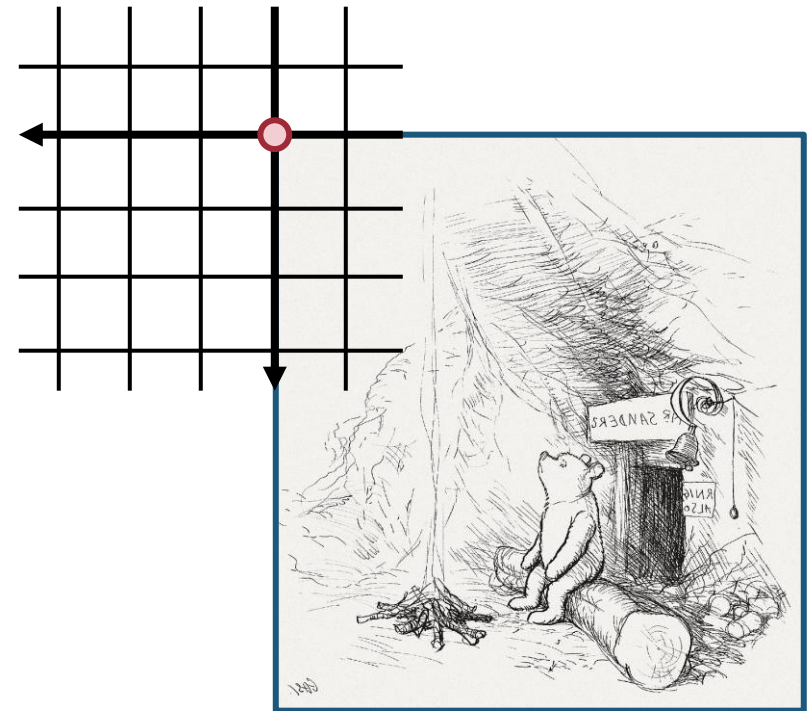
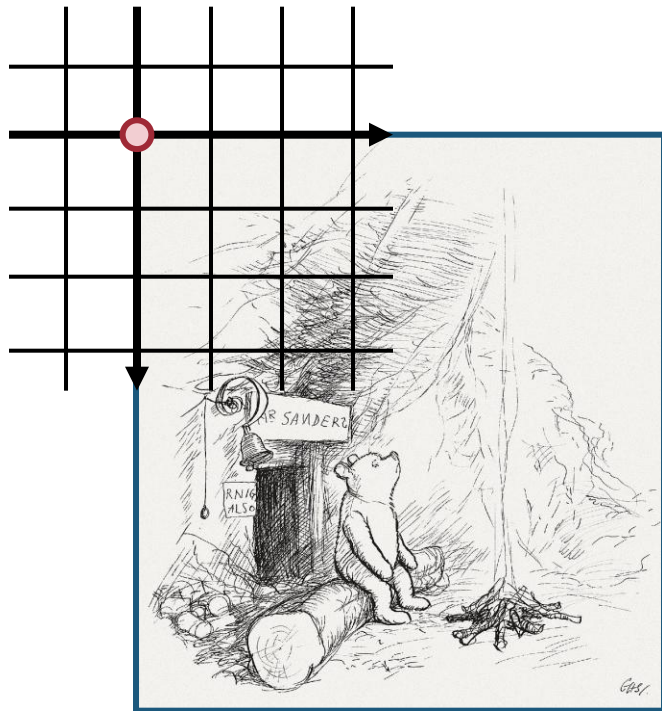


Example: Mirroring an Image

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



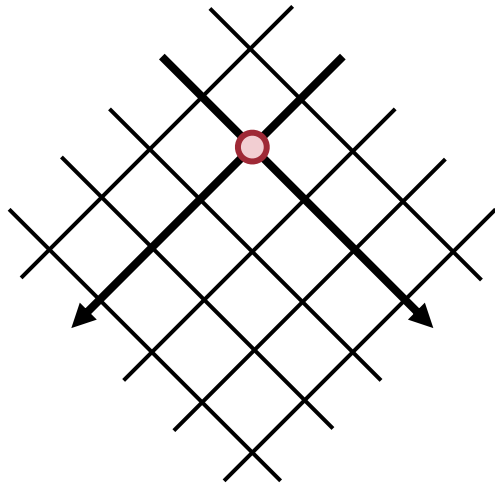
```
void draw() {  
  scale(-1, 1);  
  image(video, -video.width, 0);  
}
```



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

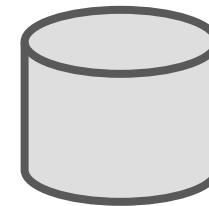
Transformation matrix



pushMatrix()

popMatrix()

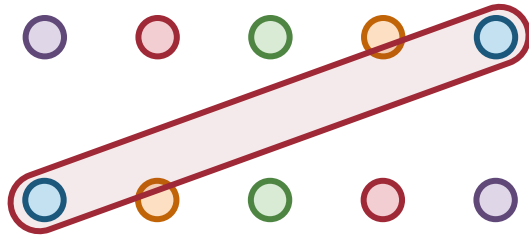
Matrix stack



First in, last out!

| Stack vs Queue

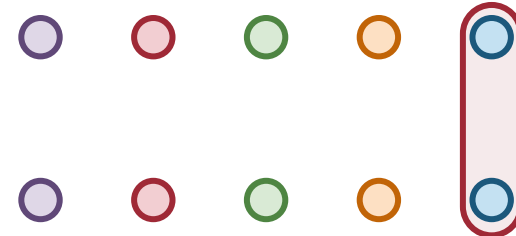
Stack



First in last out



Queue

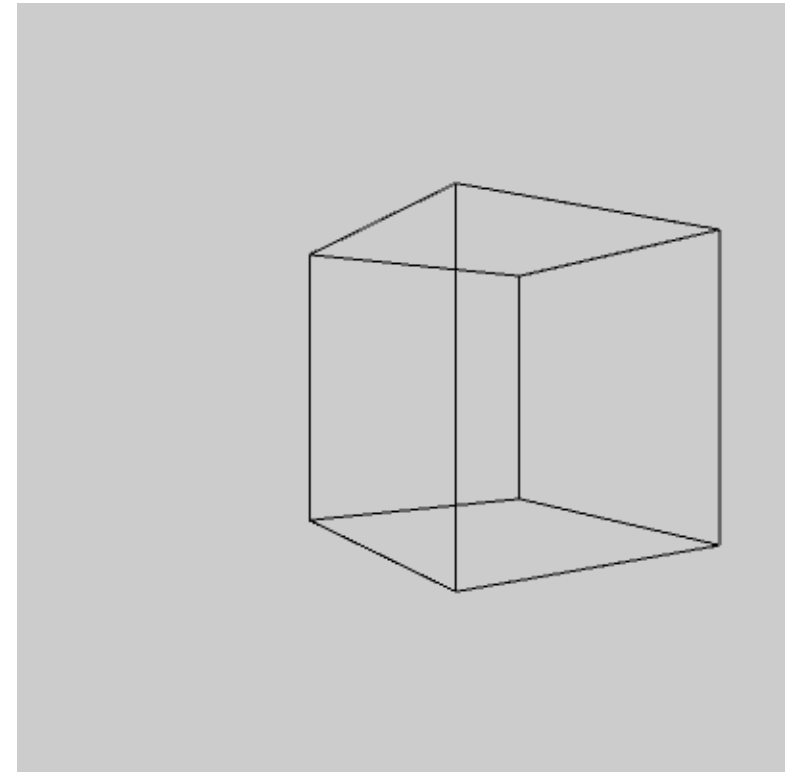


First in first out



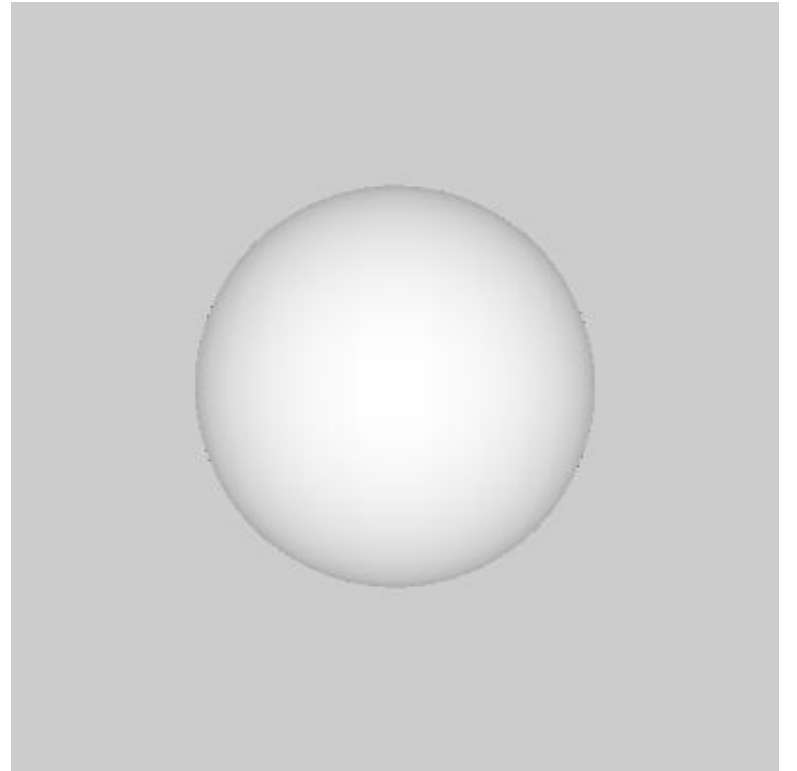
| Example: Box

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



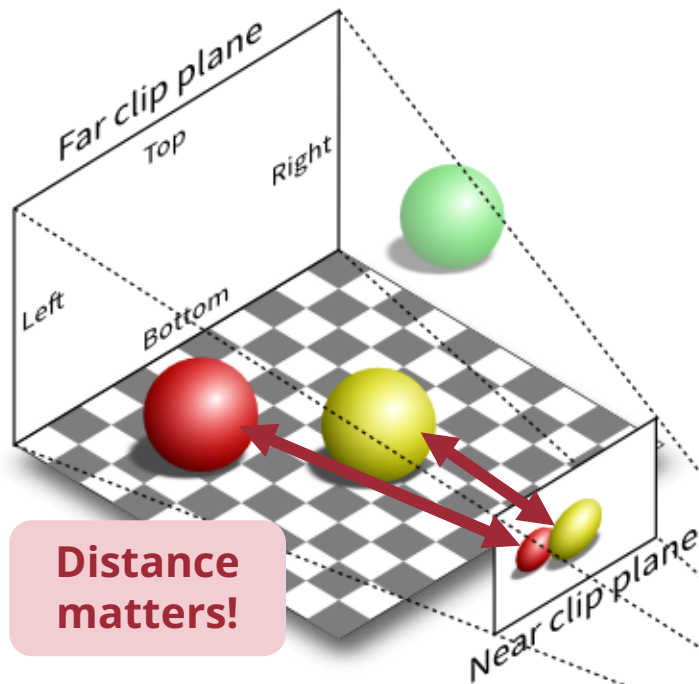
| Example: Sphere

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



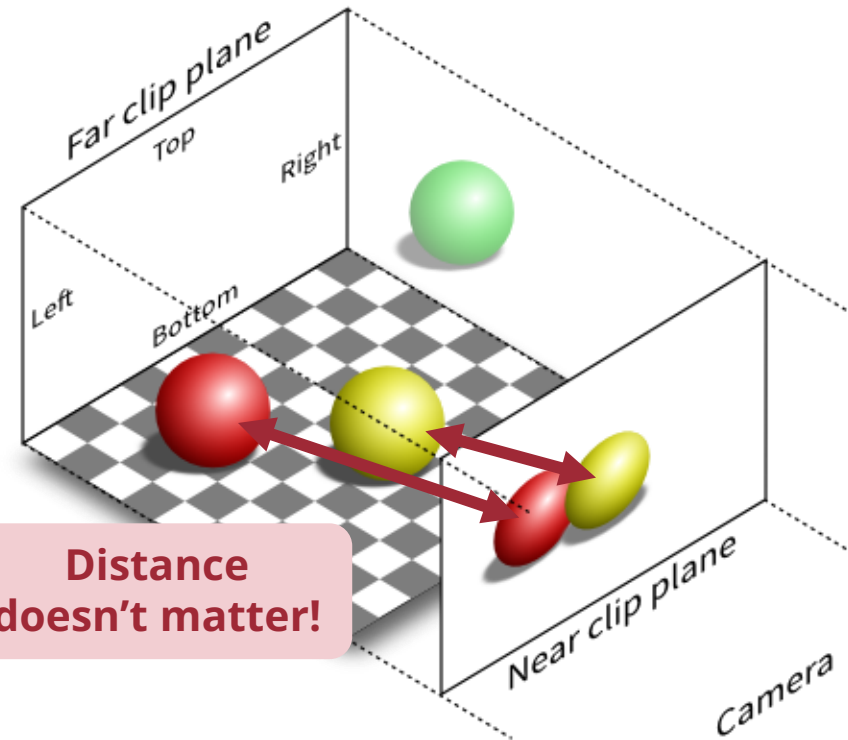
Perspective vs Orthographic Projections

perspective()



Perspective projection (P)

ortho()

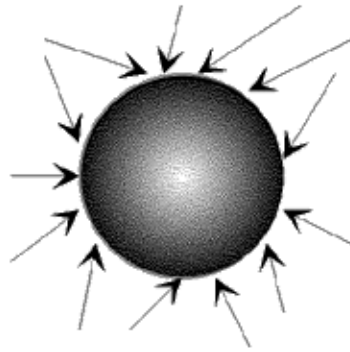


Orthographic projection (O)

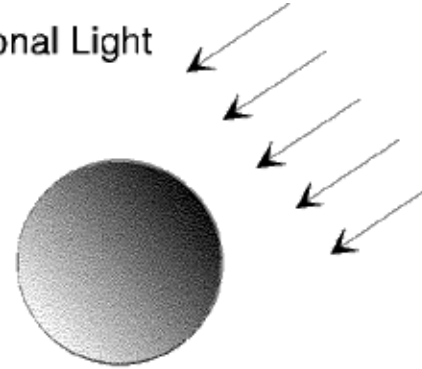
Lights

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

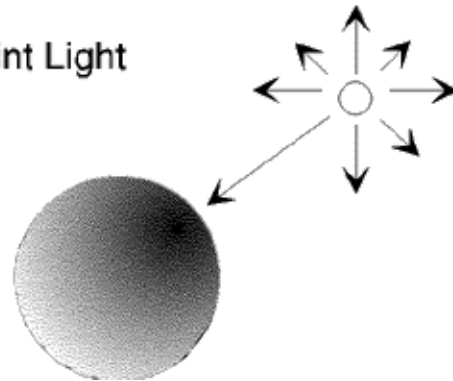
Ambient Light



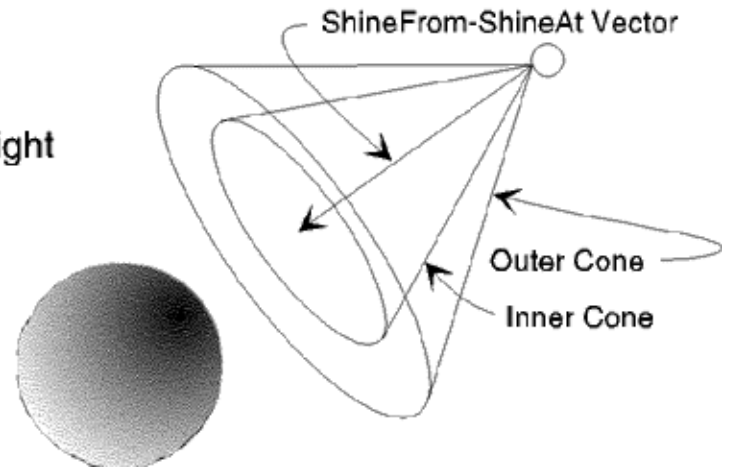
Directional Light



Point Light

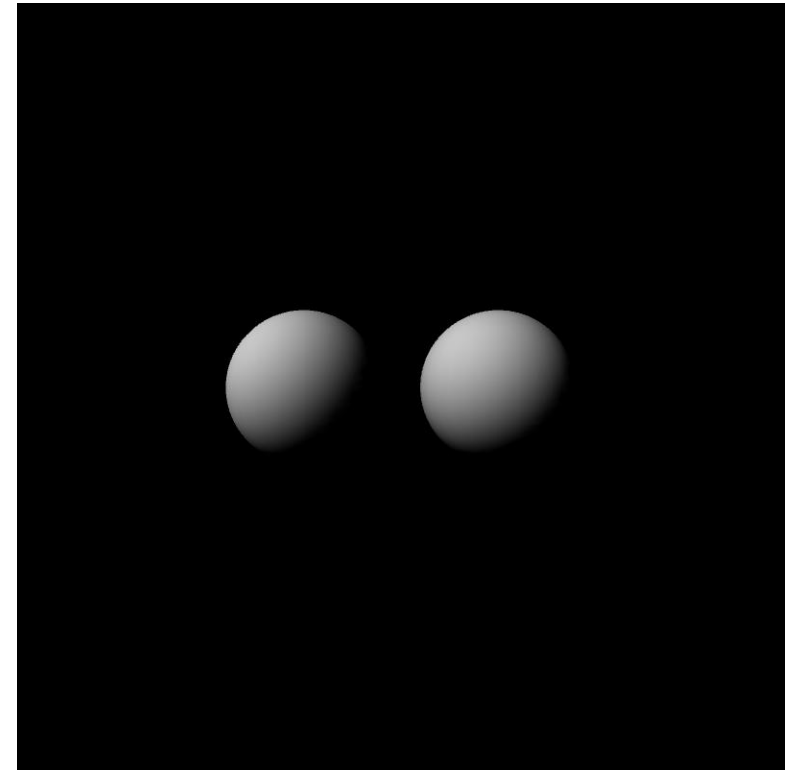


Spot Light



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);  
}
```



Next Lecture

Motion & Physics

