

PAT 204/504 (Fall 2026)

Creative Coding

Lecture 11: Motion & Physics

Instructor: Hao-Wen (Herman) Dong

Acceleration

Example: Gravity

```
// Apply gravity to the ball
```

```
void applyGravity() {  
    speedY += gravity;  
}
```

Apply gravity as y-acceleration

```
// Check if the ball hit the walls
```

```
void checkWalls() {  
    ...
```

```
// Check if the ball hit the top and bottom walls
```

```
if (y > height - radius) {
```

```
    speedY = -abs(speedY) * decay;
```

```
    y = height - radius;
```

```
} else if (y < radius) {
```

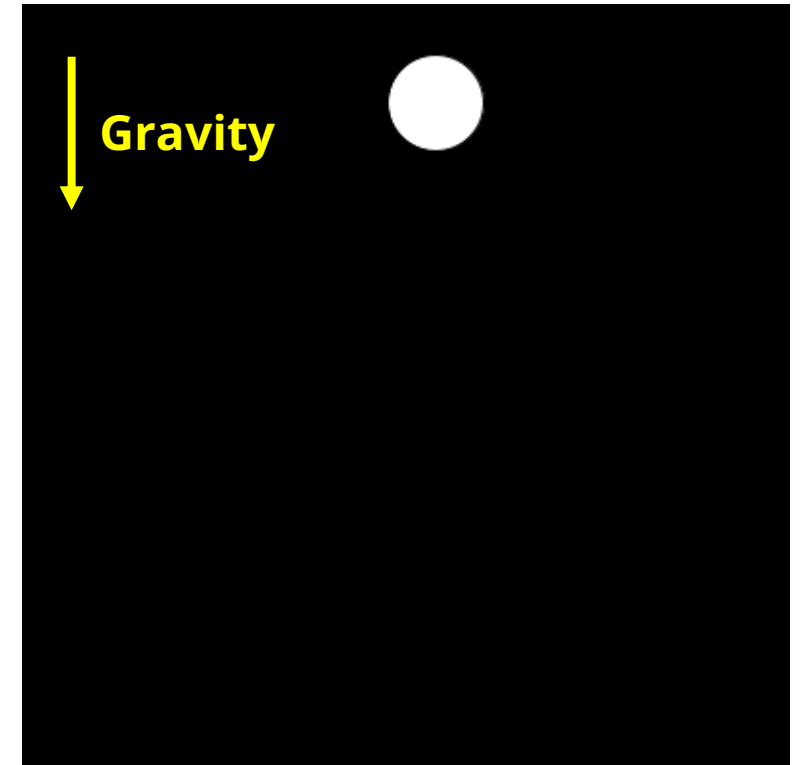
```
    speedY = abs(speedY);
```

```
    y = radius;
```

```
}
```

```
}
```

Reduce the speed a little bit
when it hits the bottom wall



Example: Upside-down Gravity

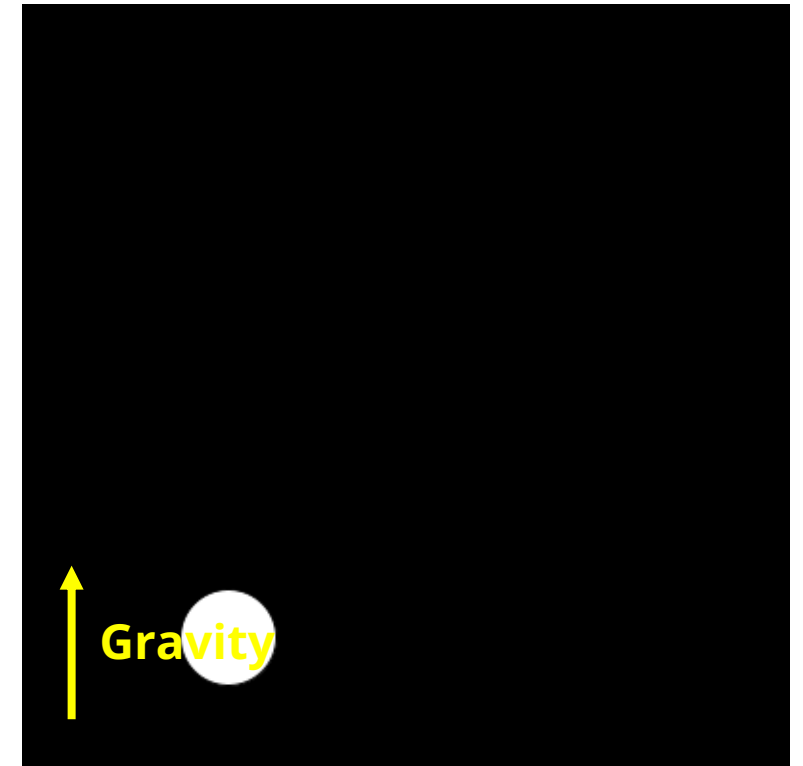
```
// Apply gravity to the ball
void applyGravity() {
    speedY -= gravity;
}

// Check if the ball hit the walls
void checkWalls() {
    ...

    // Check if the ball hit the top and bottom walls
    if (y > height - radius) {
        speedY = -abs(speedY);
        y = height - radius;
    } else if (y < radius) {
        speedY = abs(speedY) * decay;
        y = radius;
    }
}
```

Apply gravity as y-acceleration

Reduce the speed a little bit when it hits the top wall



Example: Acceleration

```
class Mover {
```

```
  PVector location;  
  PVector velocity;  
  PVector acceleration;
```

Declare the location, velocity
and acceleration vectors

```
  float topspeed = 5;
```

```
  Mover() {
```

```
    location = new PVector(width/2, height/2);  
    velocity = new PVector(0, 0);
```

Initialize the ball to be at the center
with zero initial speed

```
  void display() {
```

```
    stroke(255);
```

```
    strokeWeight(2);
```

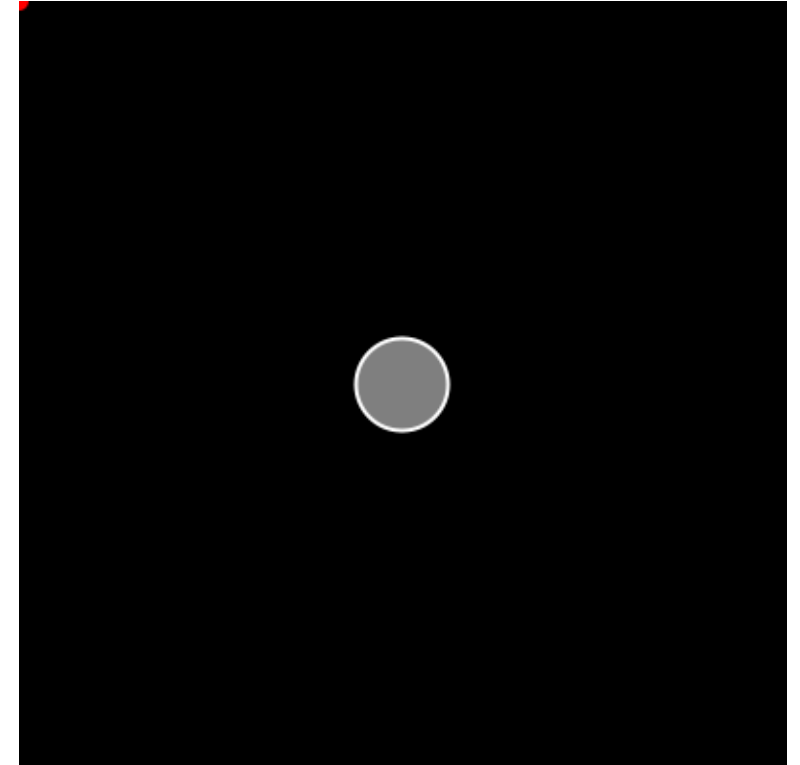
```
    fill(127);
```

```
    ellipse(location.x, location.y, 48, 48);
```

```
  }
```

```
  ...
```

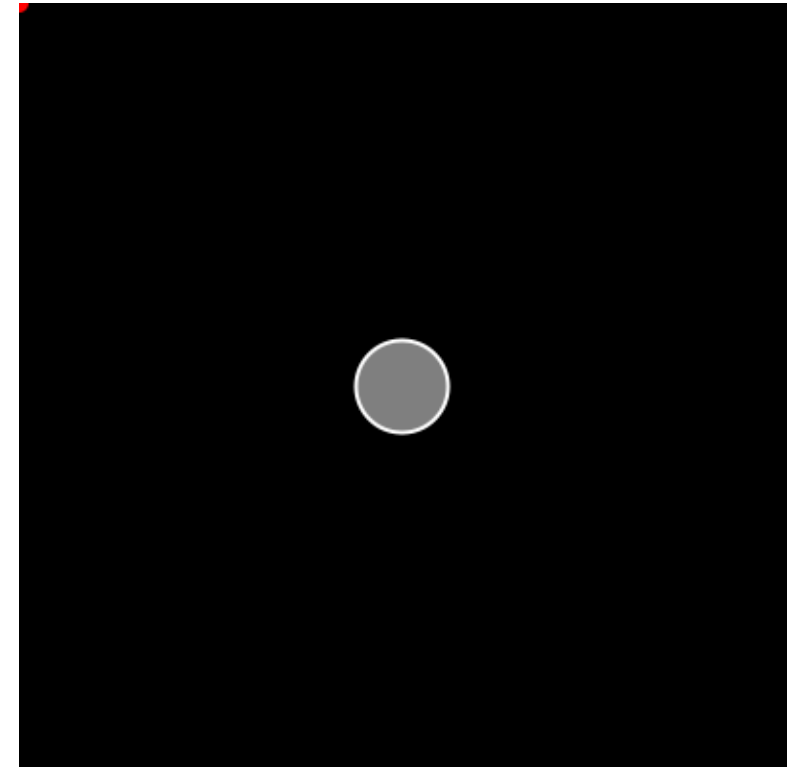
```
}
```



(Source: processing.org)

Example: Acceleration

```
class Mover {  
  PVector location;  
  PVector velocity;  
  PVector acceleration;  
  float topspeed = 5;  
  
  ...  
  
  void update() {  
    Calculate acceleration  
    PVector mouse = new PVector(mouseX, mouseY);  
    PVector acceleration = PVector.sub(mouse, location);  
    acceleration.setMag(0.2);  
  
    Apply the acceleration  
    velocity.add(acceleration);  
    velocity.limit(topspeed);  
  
    Move the ball  
    location.add(velocity);  
  }  
}
```



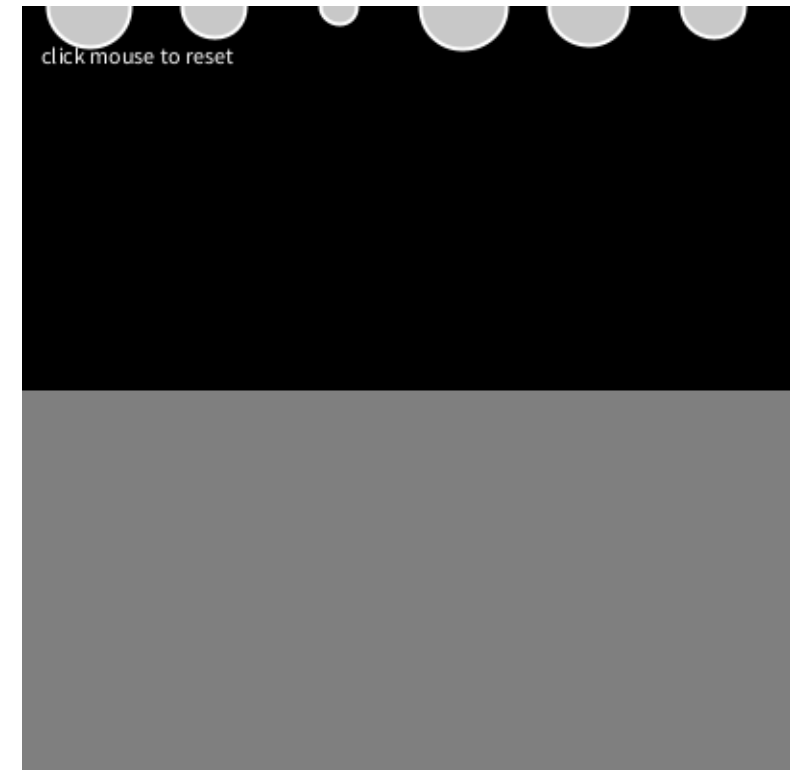
(Source: processing.org)

Example: Gravity & Fluid Resistance

- Simulating multiple forces
 - Gravity
 - Fluid resistance (drag force)

Gravity ↓

Drag force
($f \propto v^2$) ↑



(Source: processing.org)

Example: Purple Rain

- Maintaining an array of **Drop** objects
- Each **Drop** object
 - Falls with gravity
 - Random initial velocity
 - Random stroke weight (illusion of distance)



(Source: thecodingtrain.com)

Collision

Example: Bouncing Balls

`Ball[] balls = new Ball[20];` An array of objects

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

```
  for (int i = 0; i < balls.length; i++) {  
    balls[i] = new Ball();  
  }
```

Initialization

```
void draw() {  
  background(0);
```

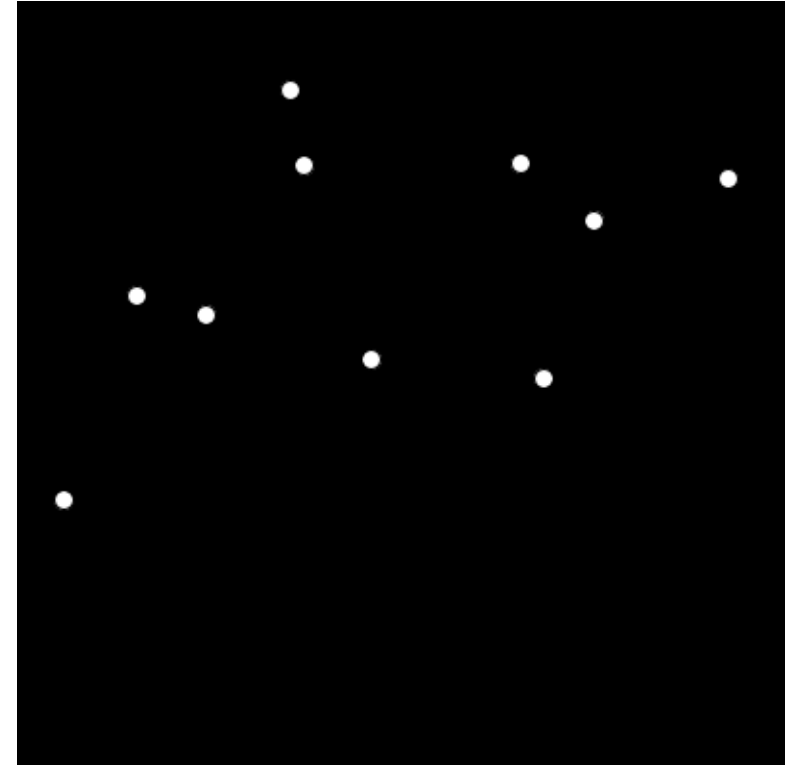
```
  for (int i = 0; i < balls.length; i++) {
```

```
    balls[i].move();
```

```
    balls[i].checkWalls();
```

```
    balls[i].show();
```

Call the methods!



Example: Bouncing Balls with Collision Detection

- What else do we need?
- How to check if the ball collides with another?
- What kind of function do we need?

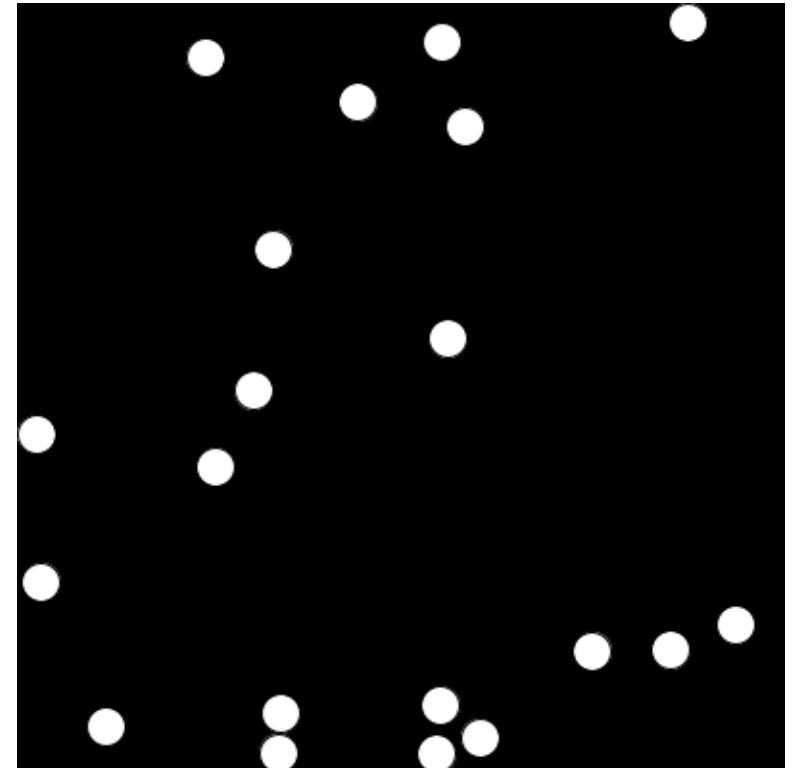
`Ball[] others;` An array to store other balls

```
void checkCollisions() {  
    for (Ball other: others) {  
        collide(other);  
    }  
}
```

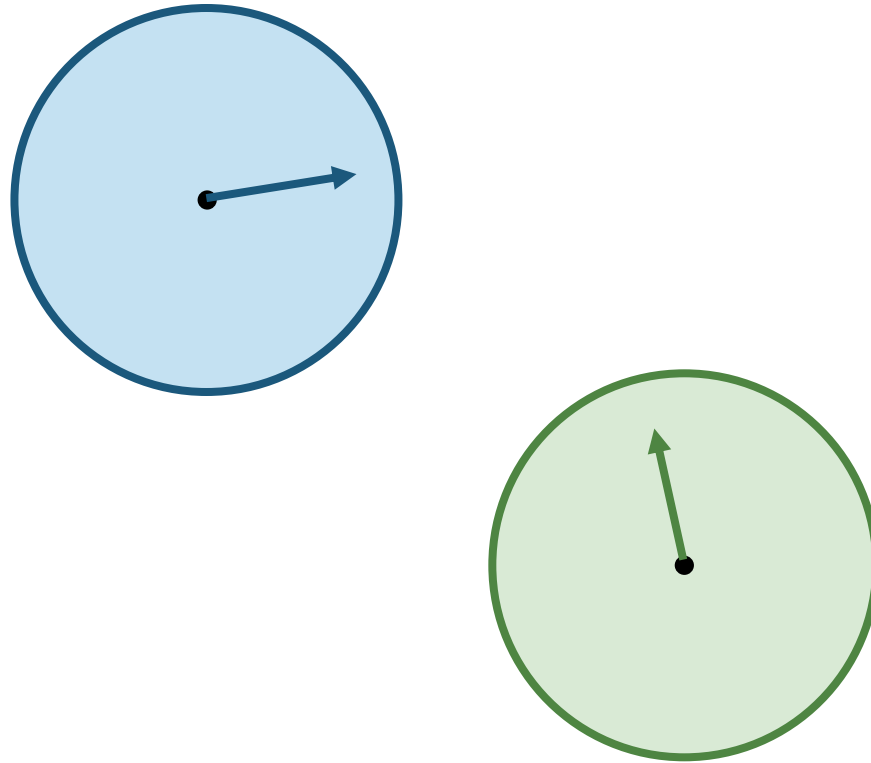
Iterate over other balls to
check if there's a collision

```
void collide(Ball other) {  
    ???  
}
```

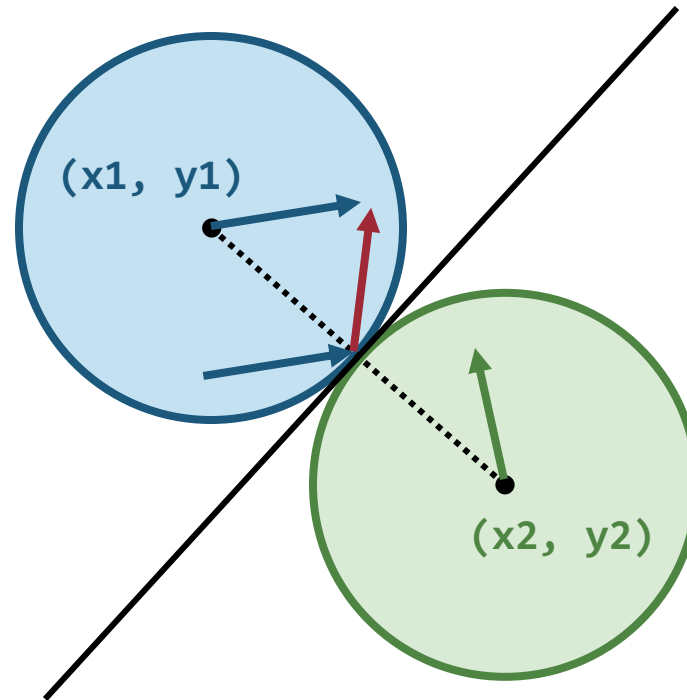
Check if there's a collision



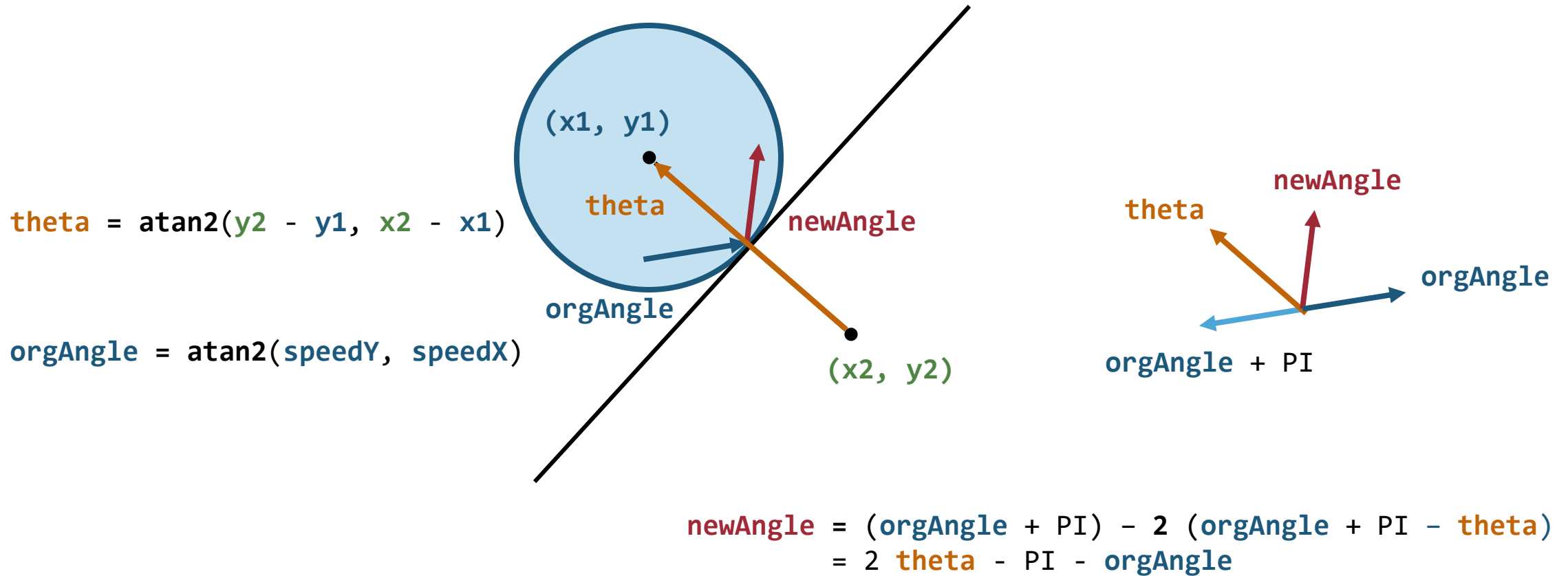
| Handling Collisions



| Handling Collisions



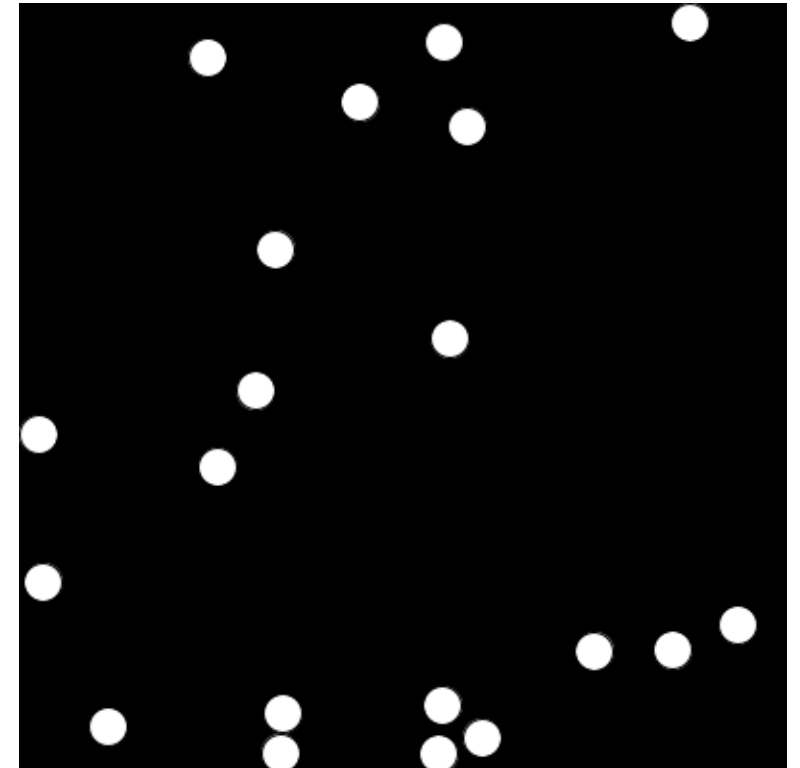
Handling Collisions



Example: Bouncing Balls with Collision Detection

```
void collide(Ball other) {  
    if (other == this) return; Do nothing if it's the same ball  
  
    float dist = dist(x, y, other.x, other.y);  
  
    if (dist >= size) return; Do nothing if they do not collide  
  
    x -= speedX; Revert the ball back to where  
    y -= speedY; it was before the collision  
  
    float theta = atan2(other.y - y, other.x - x);  
    float orgAngle = atan2(speedY, speedX);  
    float newAngle = (theta - PI + theta - orgAngle);  
    speedX = speed * cos(newAngle);  
    speedY = speed * sin(newAngle);  
}
```

Find the velocity after the collision



Elastic Collisions

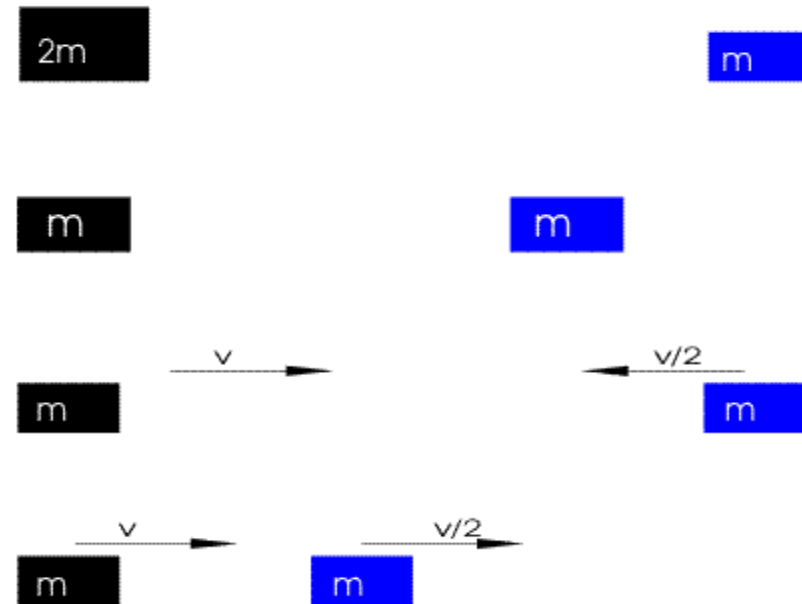
Conservation of momentum

$$m_1 v_1 + m_2 v_2 = m_1 v'_1 + m_2 v'_2$$

Conservation of kinetic energy

$$m_1 v_1 + m_2 v_2 = \frac{1}{2} m_1 v'^2_1 + \frac{1}{2} m_2 v'^2_2$$

(Does not hold for inelastic collision)



(Source: Roque)

2D Elastic Collisions

Conservation of momentum

$$m_1 \vec{v}_1 + m_2 \vec{v}_2 = m_1 \vec{v}'_1 + m_2 \vec{v}'_2$$

Conservation of kinetic energy

$$m_1 \vec{v}_1 + m_2 \vec{v}_2 = \frac{1}{2} m_1 \vec{v}'_1{}^2 + \frac{1}{2} m_2 \vec{v}'_2{}^2$$

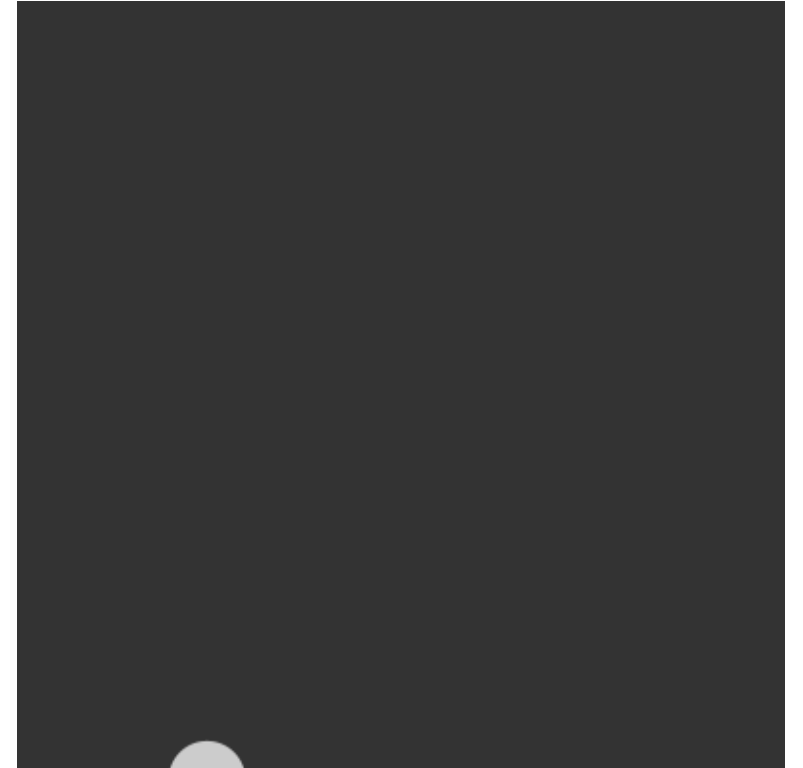
Conservation of angular momentum



(Source: Steinmann)

Example: Collision with **Weights**

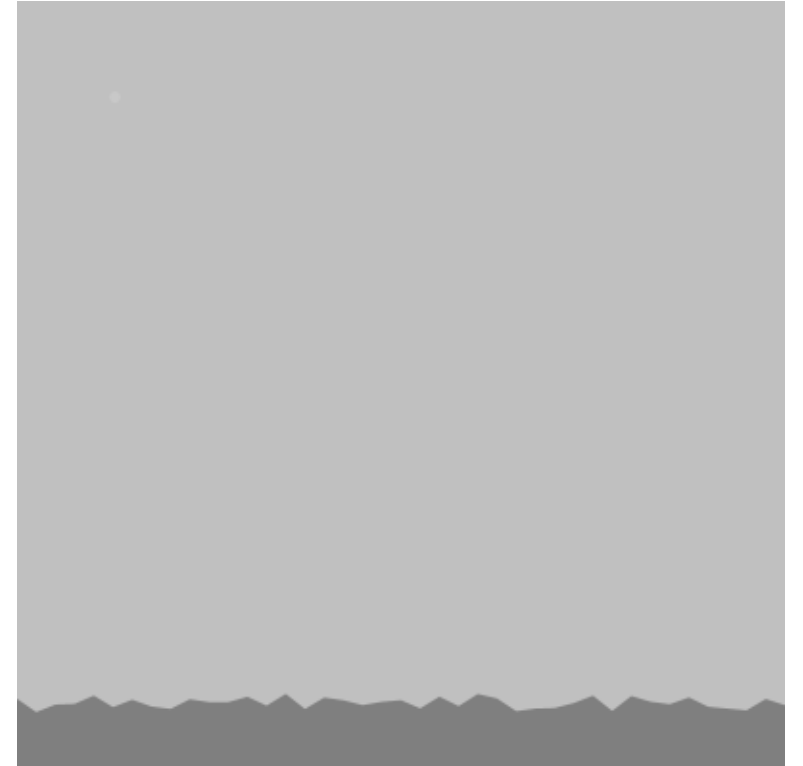
- Different **weights** and different **speeds**
- Solve the equations to find the new velocities
 - Conservation of momentum
 - Conservation of kinetic energy
 - Conservation of angular momentum



(Source: processing.org)

Example: Reflection

- Several **Ground** objects
 - Initialized with random widths and slopes
 - Together compose the ground surface
- An **Orb** object
 - Initialized with a small x-velocity
 - Influenced by gravity
 - May collide with the walls and the grounds

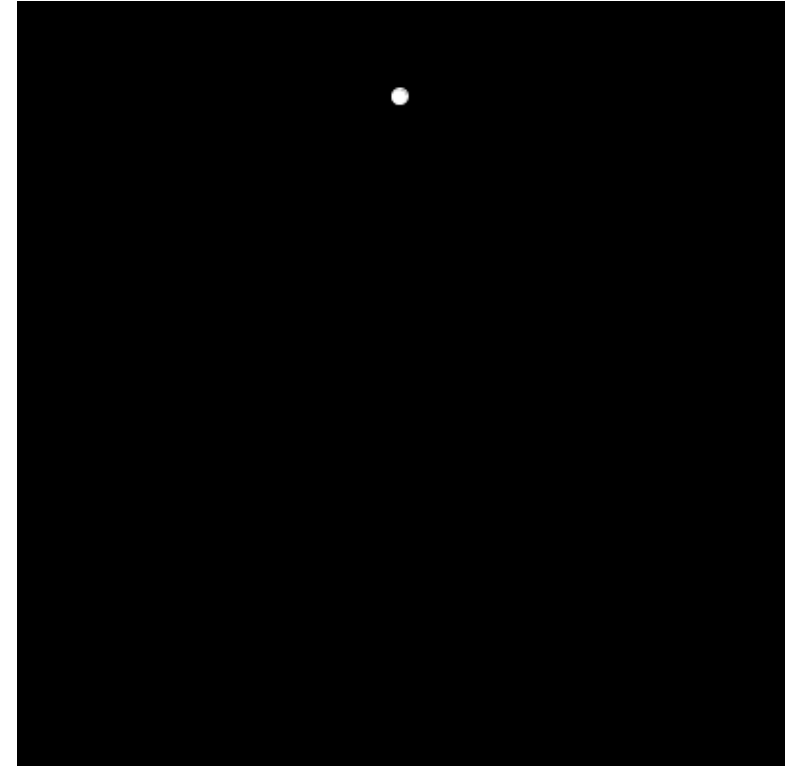


(Source: processing.org)

Particle Systems

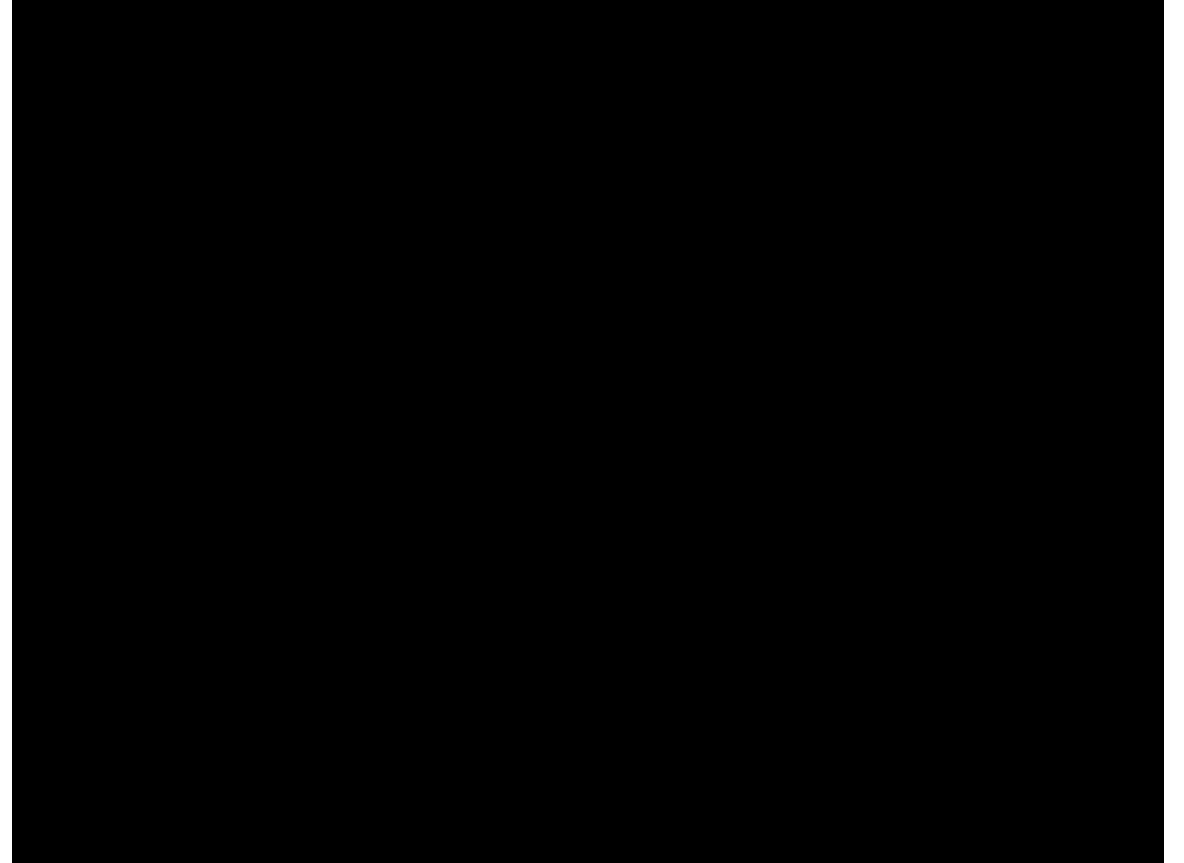
Example: Simple Particle System

- A **Particle** object
 - Falls with gravity
 - Fades out over time (die after a predefined lifespan)
- A **ParticleSystem** object is an ArrayList of **Particle** objects (i.e., **ArrayList<Particle>**)
 - Allows storing a variable number of particles
- Show all the **Particle** objects at each call of the `draw()` function



Example: Fireworks

- A **Firework** object
 - Starts as one single **Particle** object
 - Initialized with random force up
 - Flies up with gravity slowing it down
 - Explodes when speed reaches zero
 - Becomes many **Particle** objects after explosion
 - Initialized with random forces towards random directions
 - Fall with gravity
 - Die after invisible on the canvas



Recap

Example: Gravity

```
// Apply gravity to the ball
```

```
void applyGravity() {  
    speedY += gravity;  
}
```

Apply gravity as y-acceleration

```
// Check if the ball hit the walls
```

```
void checkWalls() {  
    ...
```

```
// Check if the ball hit the top and bottom walls
```

```
if (y > height - radius) {
```

```
    speedY = -abs(speedY) * decay;
```

```
    y = height - radius;
```

```
} else if (y < radius) {
```

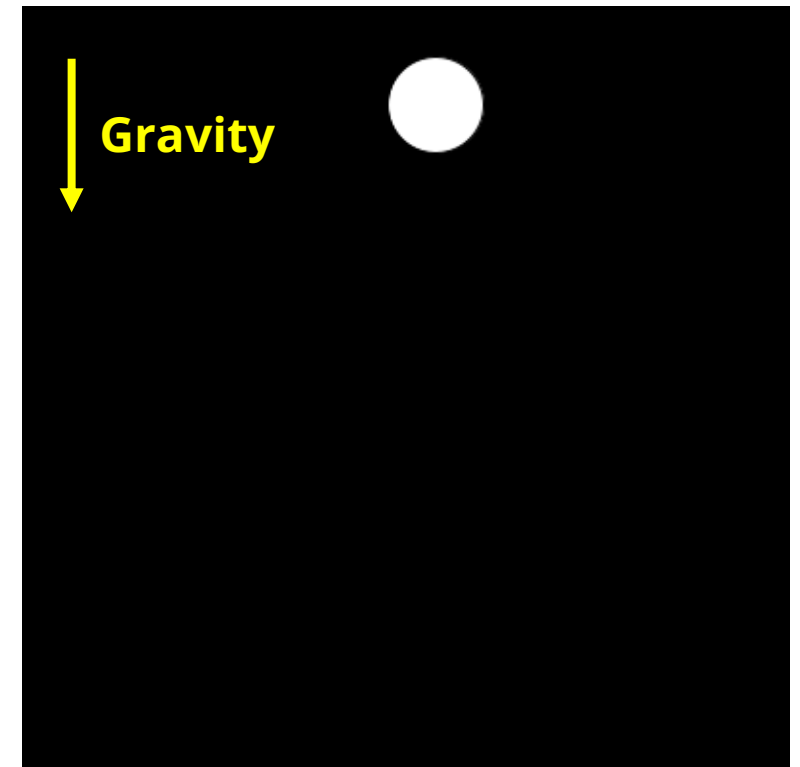
```
    speedY = abs(speedY);
```

```
    y = radius;
```

```
}
```

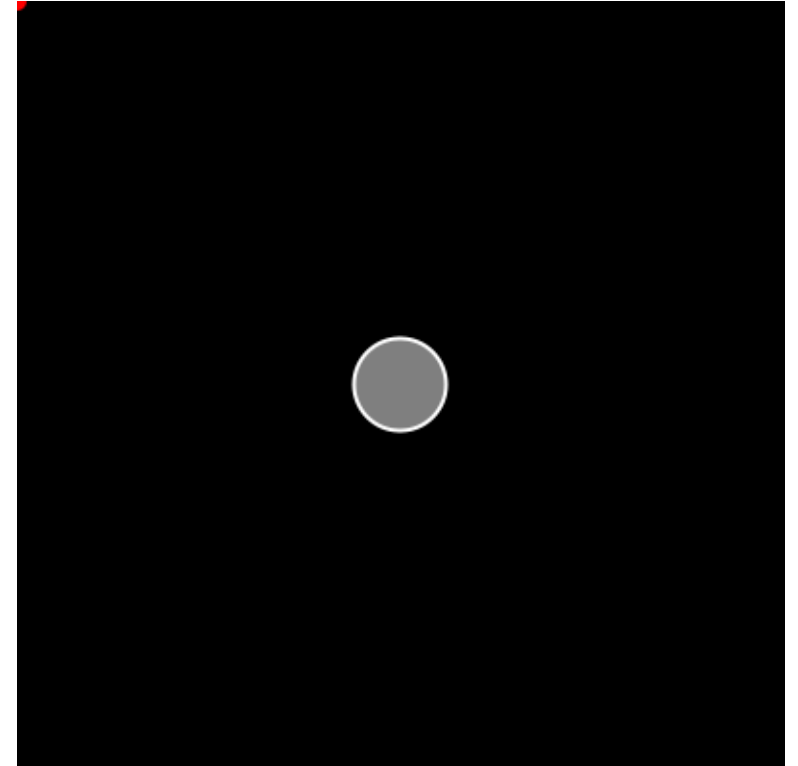
```
}
```

Reduce the speed a little bit
when it hits the bottom wall



Example: Acceleration

```
class Mover {  
  PVector location;  
  PVector velocity;  
  PVector acceleration;  
  float topspeed = 5;  
  
  ...  
  
  void update() {  
    Calculate acceleration  
    PVector mouse = new PVector(mouseX, mouseY);  
    PVector acceleration = PVector.sub(mouse, location);  
    acceleration.setMag(0.2);  
  
    Apply the acceleration  
    velocity.add(acceleration);  
    velocity.limit(topspeed);  
  
    Move the ball  
    location.add(velocity);  
  }  
}
```



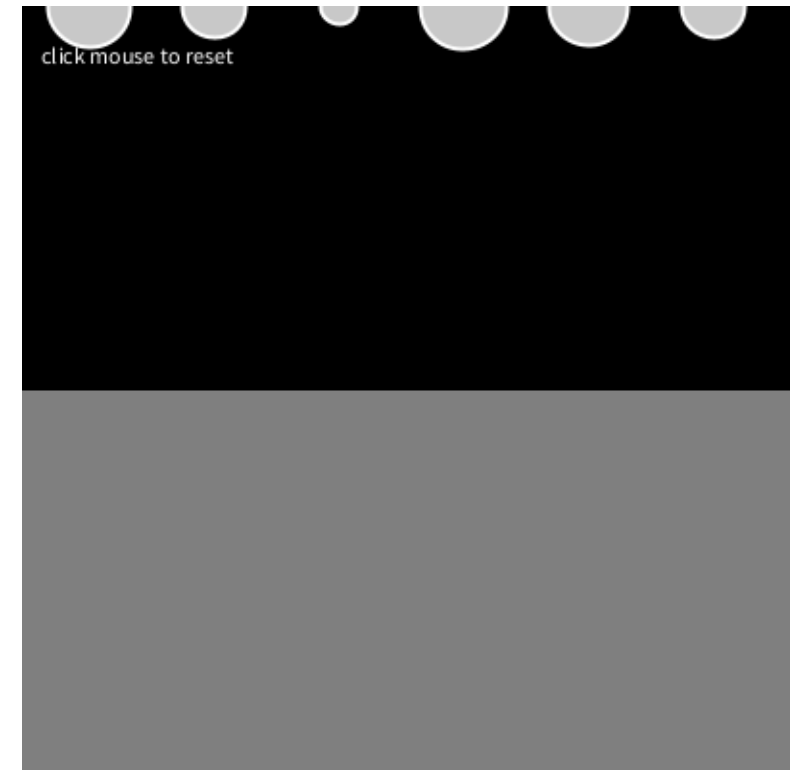
(Source: processing.org)

Example: Gravity & Fluid Resistance

- Simulating multiple forces
 - Gravity
 - Fluid resistance (drag force)

Gravity ↓

Drag force
 $(f \propto v^2)$ ↑



(Source: processing.org)

Example: Bouncing Balls with Collision Detection

- What else do we need?
- How to check if the ball collides with another?
- What kind of function do we need?

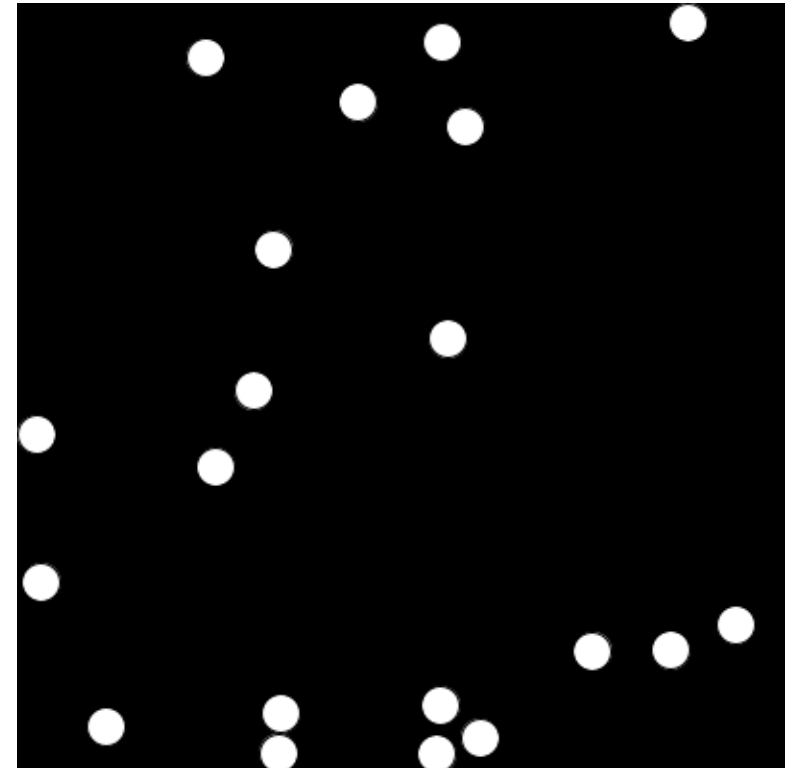
`Ball[] others;` An array to store other balls

```
void checkCollisions() {  
    for (Ball other: others) {  
        collide(other);  
    }  
}
```

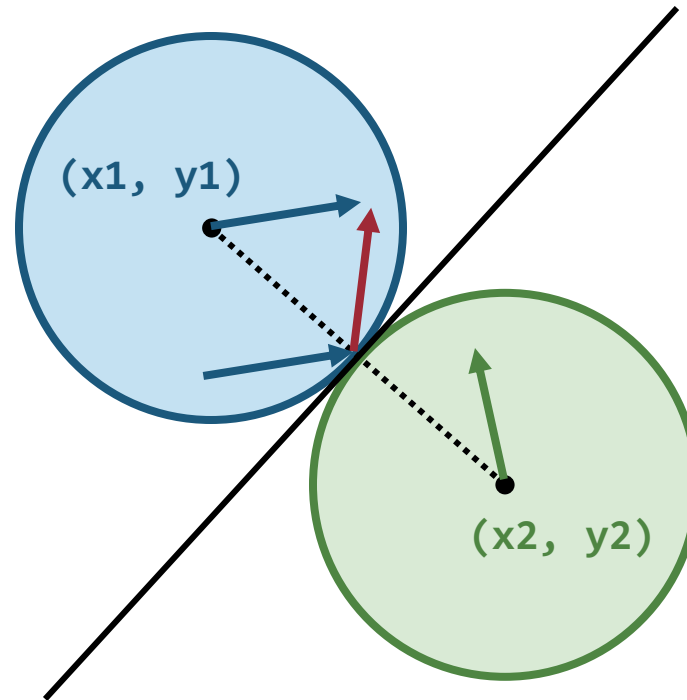
Iterate over other balls to check if there's a collision

```
void collide(Ball other) {  
    ???  
}
```

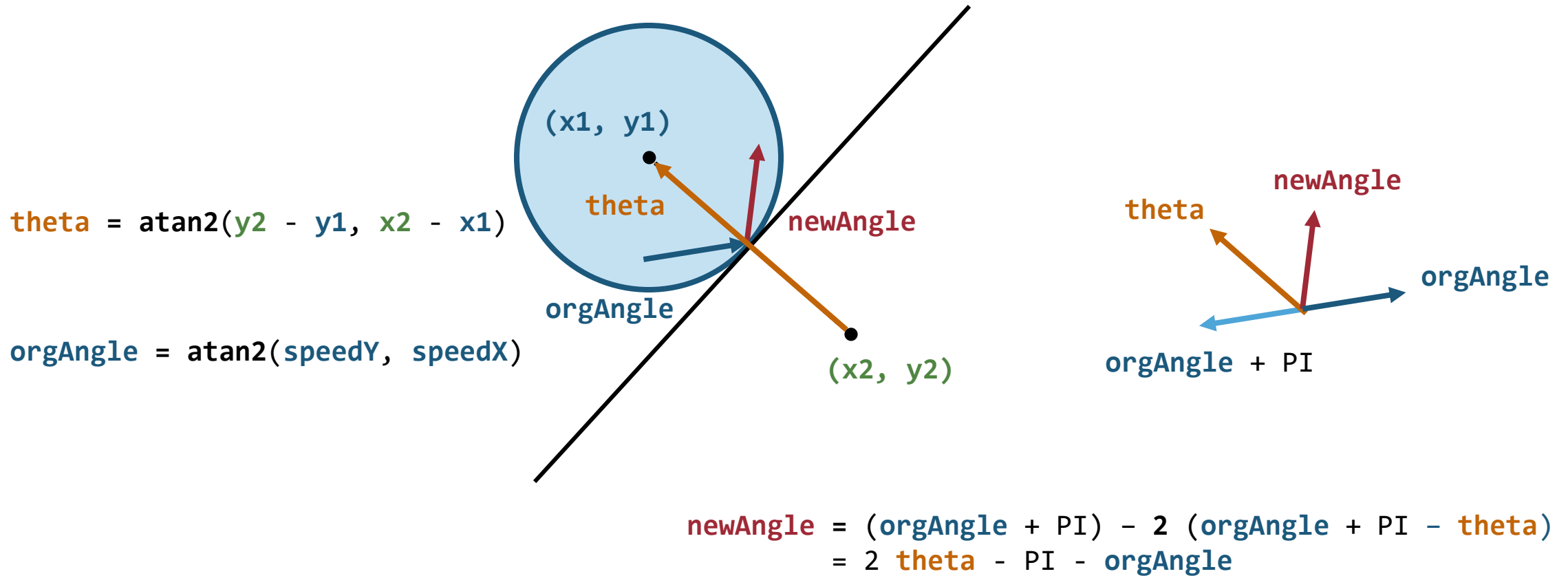
Check if there's a collision



| Handling Collisions



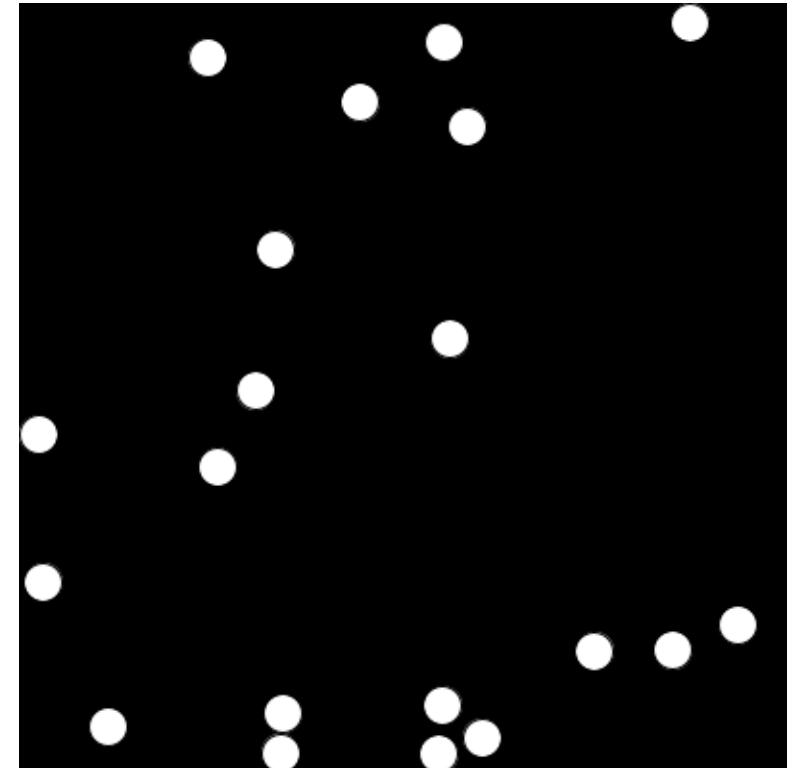
Handling Collisions



Example: Bouncing Balls with Collision Detection

```
void collide(Ball other) {  
    if (other == this) return; Do nothing if it's the same ball  
  
    float dist = dist(x, y, other.x, other.y);  
  
    if (dist >= size) return; Do nothing if they do not collide  
  
    x -= speedX; Revert the ball back to where  
    y -= speedY; it was before the collision  
  
    float theta = atan2(other.y - y, other.x - x);  
    float orgAngle = atan2(speedY, speedX);  
    float newAngle = (theta - PI + theta - orgAngle);  
    speedX = speed * cos(newAngle);  
    speedY = speed * sin(newAngle);  
}
```

Find the velocity after the collision



2D Elastic Collisions

Conservation of momentum

$$m_1 \vec{v}_1 + m_2 \vec{v}_2 = m_1 \vec{v}'_1 + m_2 \vec{v}'_2$$

Conservation of kinetic energy

$$m_1 \vec{v}_1 + m_2 \vec{v}_2 = \frac{1}{2} m_1 \vec{v}'_1{}^2 + \frac{1}{2} m_2 \vec{v}'_2{}^2$$

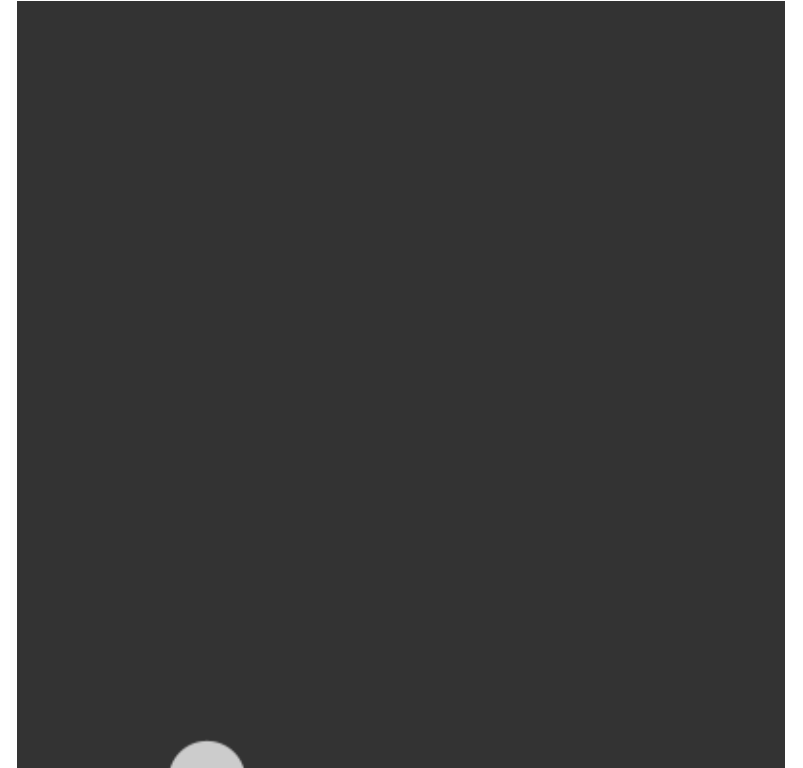
Conservation of angular momentum



(Source: Steinmann)

Example: Collision with **Weights**

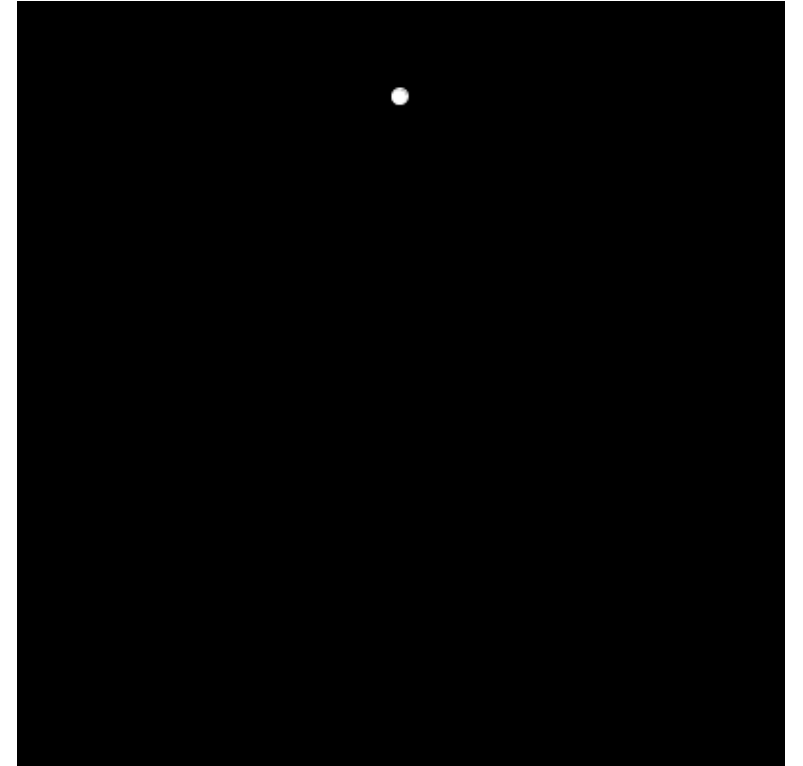
- Different **weights** and different **speeds**
- Solve the equations to find the new velocities
 - Conservation of momentum
 - Conservation of kinetic energy
 - Conservation of angular momentum



(Source: processing.org)

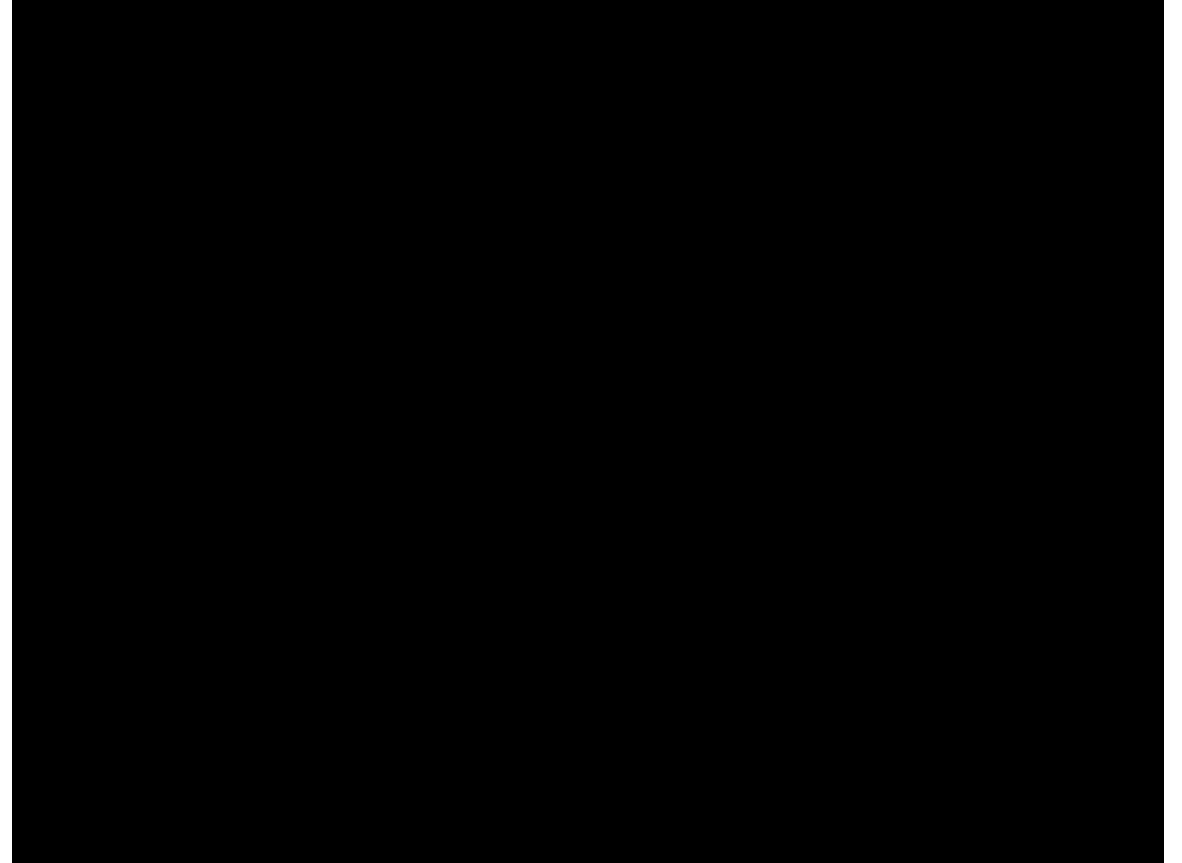
Example: Simple Particle System

- A **Particle** object
 - Falls with gravity
 - Fades out over time (die after a predefined lifespan)
- A **ParticleSystem** object is an ArrayList of **Particle** objects (i.e., **ArrayList<Particle>**)
 - Allows storing a variable number of particles
- Show all the **Particle** objects at each call of the `draw()` function



Example: Fireworks

- A **Firework** object
 - Starts as one single **Particle** object
 - Initialized with random force up
 - Flies up with gravity slowing it down
 - Explodes when speed reaches zero
 - Becomes many **Particle** objects after explosion
 - Initialized with random forces towards random directions
 - Fall with gravity
 - Die after invisible on the canvas



Next Lecture

Midterm Assignment Showcase

