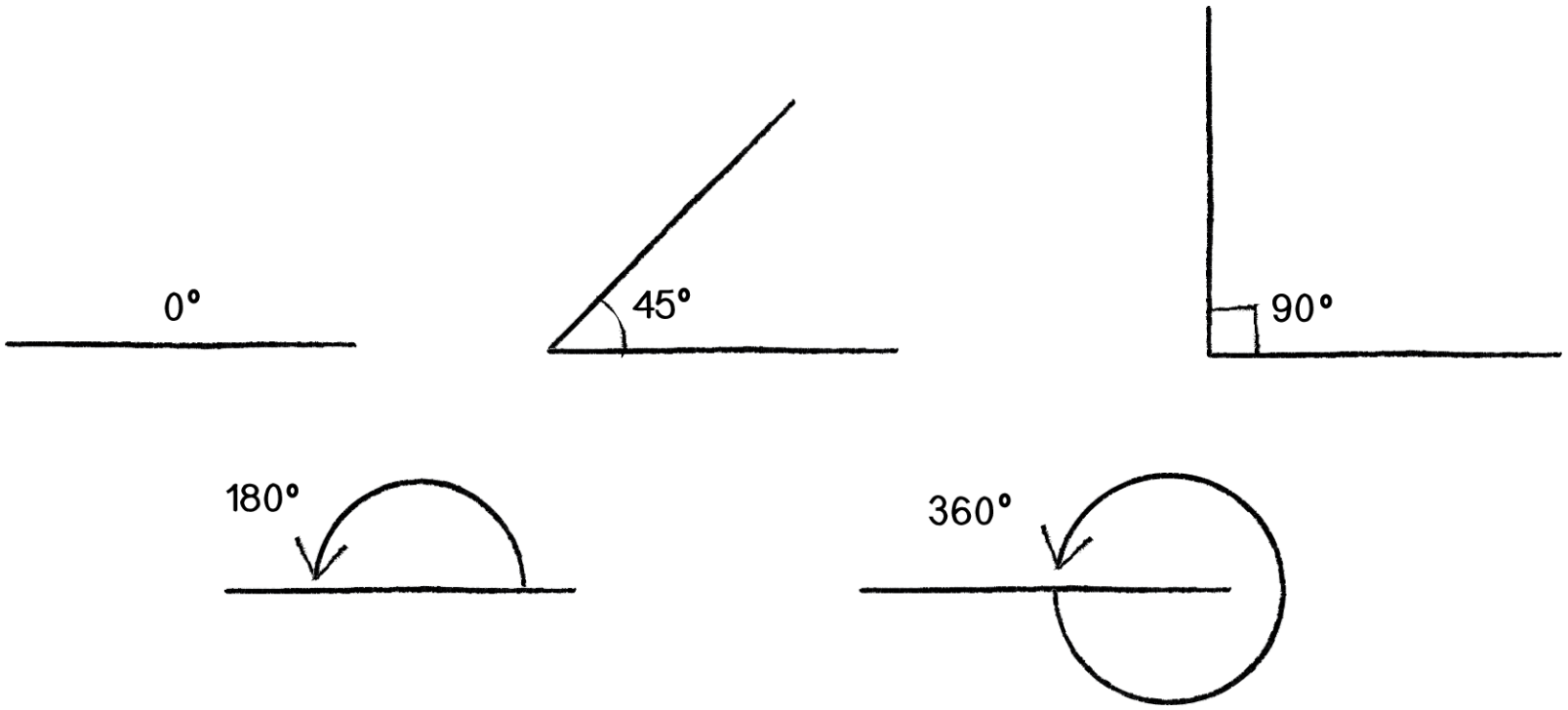
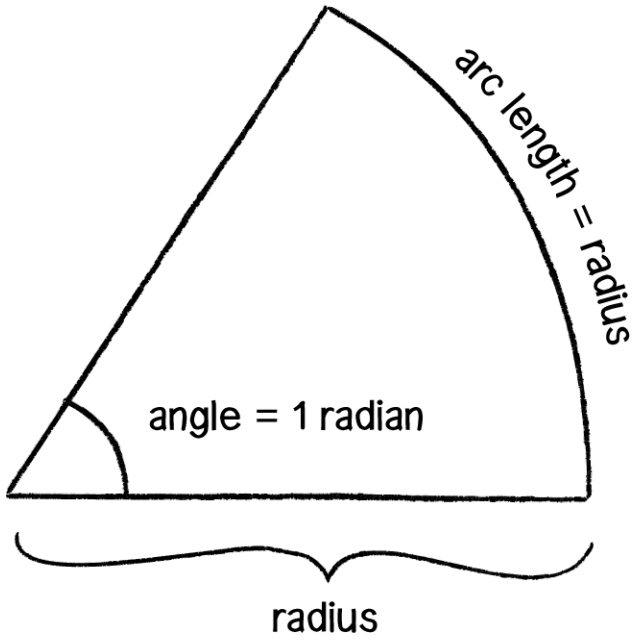
3. Oscillation

박종화
suakii@gmail.com

Angles



Angles



Angular Motion(회전 운동)

- location = location + velocity
  velocity = velocity + acceleration

- angle = angle + angular velocity
  angular velocity = angular velocity + angular acceleration

Angular Motion

```
float angle = 0;
Velocity
float aVelocity = 0;
Acceleration
float aAcceleration = 0.001;

void setup() {
  size(640,360);
}

void draw() {
  background(255);

  fill(175);
  stroke(0);
  rectMode(CENTER);
  translate(width/2,height/2);
  rotate(angle);
  line(-50,0,50,0);
  ellipse(50,0,8,8);
  ellipse(-50,0,8,8);

aVelocity += aAcceleration;
angle += aVelocity;
}
```

Angular Motion - Mover

```
void update() {

velocity.add(acceleration);
    location.add(velocity);

aVelocity += aAcceleration;
    angle += aVelocity;

    acceleration.mult(0);
  }
```

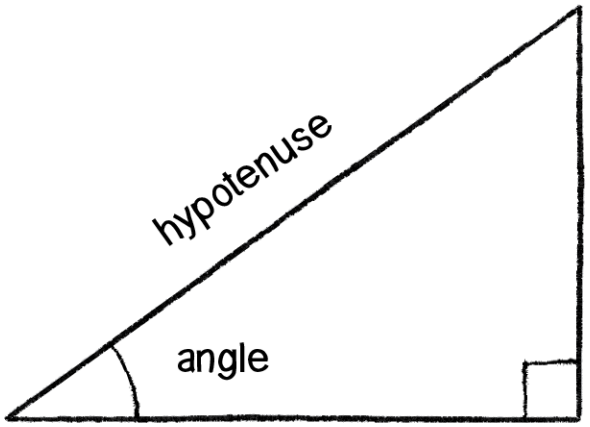
Angular Motion - Mover

```
void display() {
   stroke(0);
   fill(175,200);
   rectMode(CENTER);
   pushMatrix();
      translate(location.x,location.y);
      rotate(angle);
      rect(0,0,mass*16,mass*16);
   popMatrix();
 }
```

Cannon Example

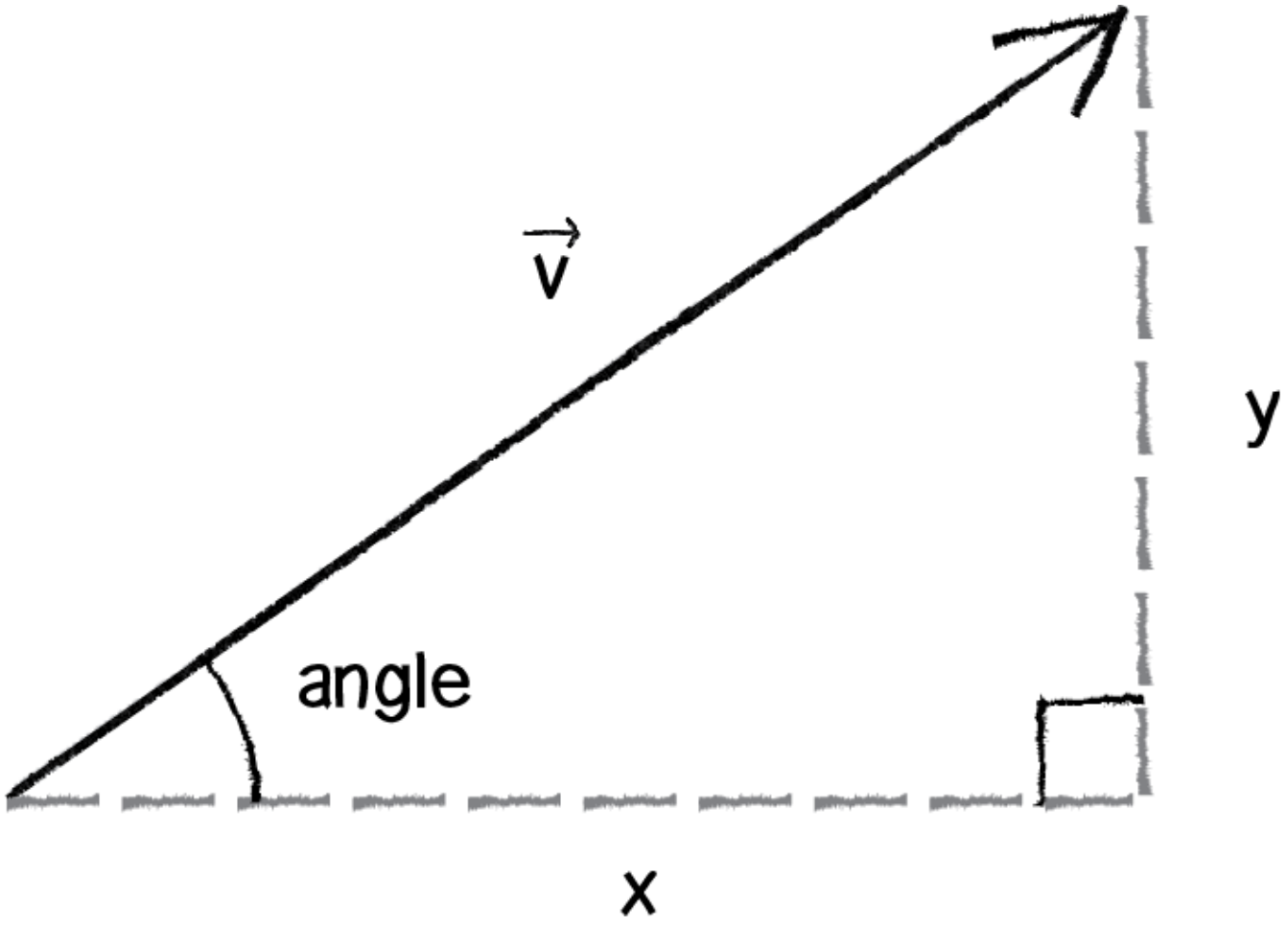
- sketch_4_Exercise_3_02_cannon

Trigonometry

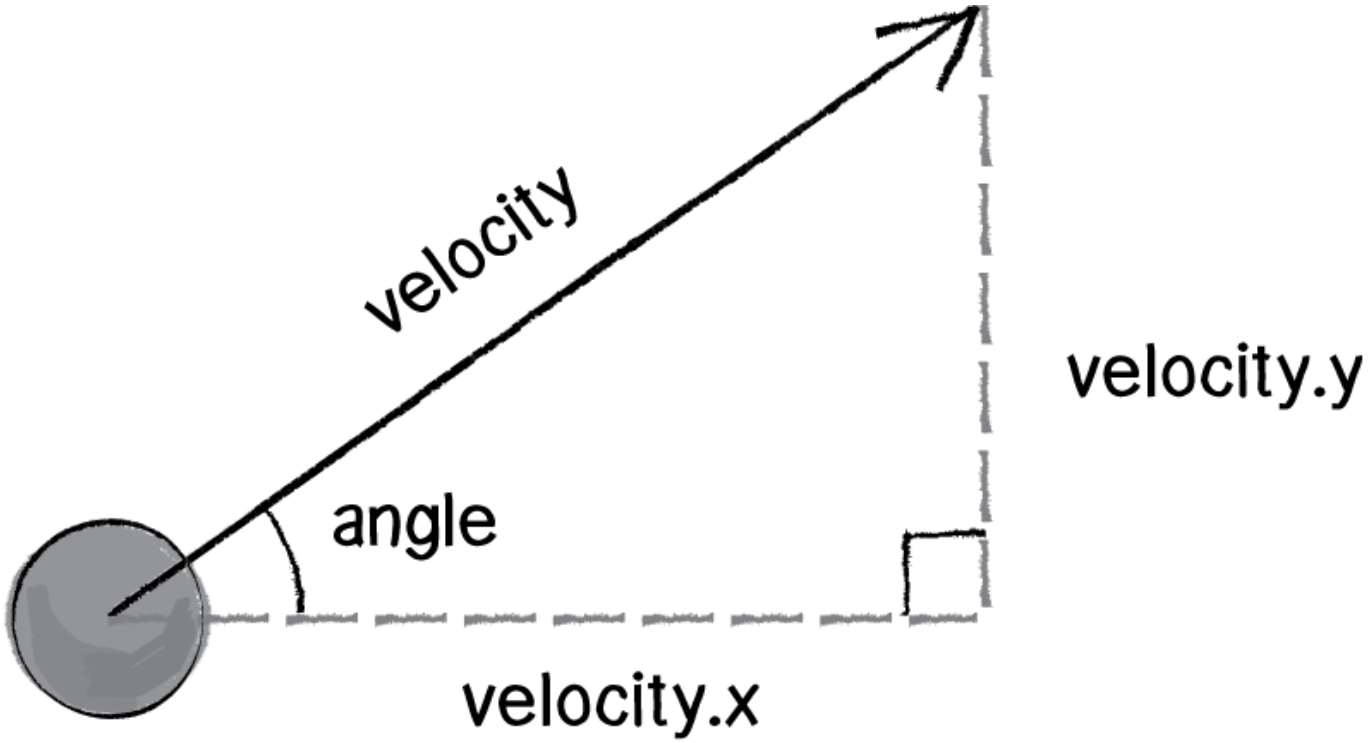


right angle = 90° = $\pi/2$ radians

Trigonometry



Pointing in the Direction of Movement

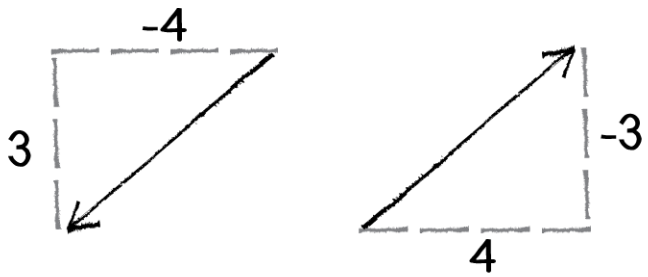


tangent(angle) = velocity.y/velocity.x

Pointing in the Direction of Movement

```
void display() {
   //Solve for angle by using atan().
   float angle = atan(velocity.y/velocity.x);

   stroke(0);
   fill(175);
   pushMatrix();
   rectMode(CENTER);
   translate(location.x,location.y);
   //Rotate according to that angle.
   rotate(angle);
   rect(0,0,30,10);
   popMatrix();
}
```

Actan Problem

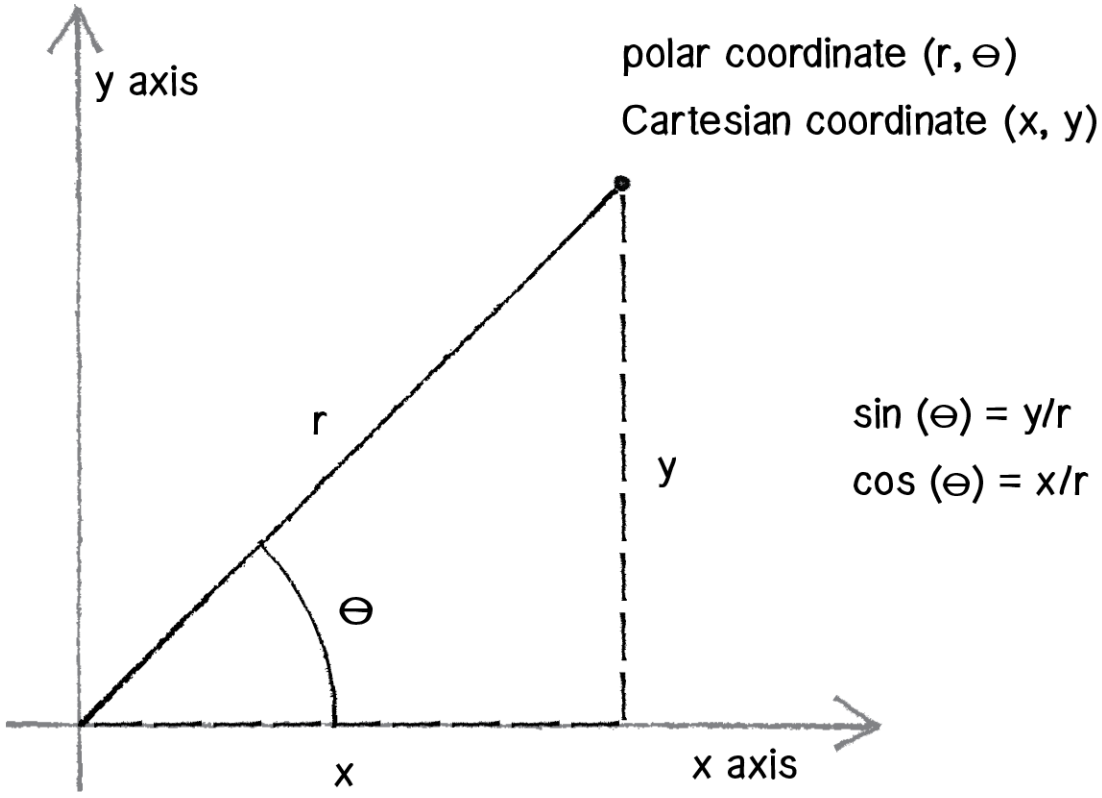


$\vec{v1}$= (-4, 3) $\vec{v2}$= (4, -3)

Actan2 or Heading()

```
void display() {
   //Using atan2() to account for all possible directions
   float angle = atan2(velocity.y,velocity.x);
   //angle = velocity.heading();
   stroke(0);
   fill(175);
   pushMatrix();
   rectMode(CENTER);
   translate(location.x,location.y);
   //Rotate according to that angle.
   rotate(angle);
   rect(0,0,30,10);
   popMatrix();
 }
```

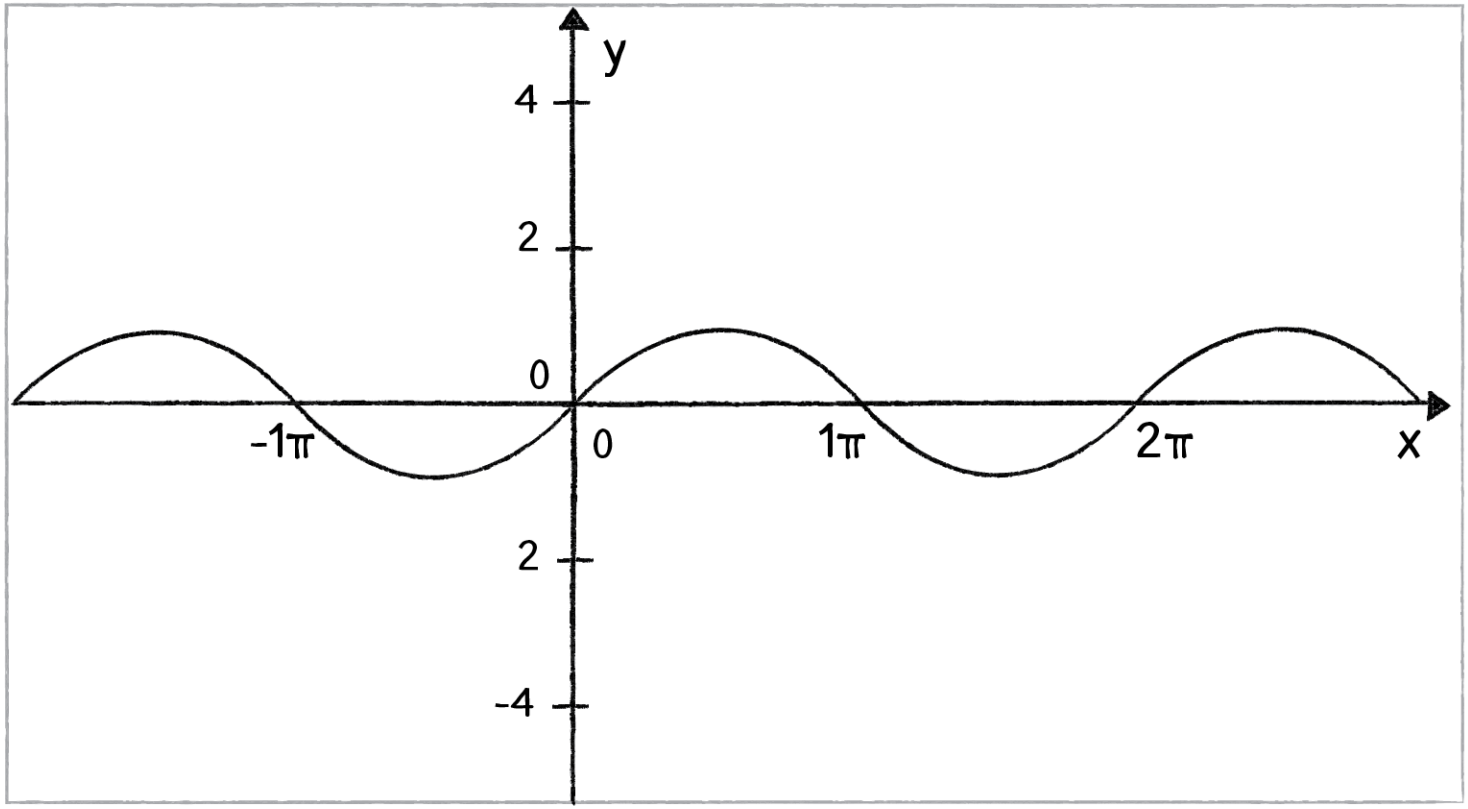
Polar vs Cartesian Coordinates



PolarToCartesian

```
float r = 75;
float theta = PI / 4;
//Converting from polar (r,theta) to
//Cartesian (x,y)
float x = r * cos(theta);
float y = r * sin(theta);
```

Oscillation Amplitude and Period



Simple Harmonic Motion

```
void setup() {
  size(640,360);
}

void draw() {
  background(255);

  float period = 120;
  float amplitude = 100;
 //Calculating horizontal location according to the formula for simple harmonic motion
  float x = amplitude * cos(TWO_PI * frameCount / period);
  stroke(0);
  fill(175);
  translate(width/2,height/2);
  line(0,0,x,0);
  ellipse(x,0,20,20);
}
```

Oscillation with Angular Velocity

```
float angle = 0;
float aVelocity = 0.05;

void setup() {
  size(640,360);
}

void draw() {
  background(255);

  float amplitude = 100;
  float x = amplitude * cos(angle);
  //Using the concept of angular velocity to increment an angle variable
  angle += aVelocity;

  ellipseMode(CENTER);
  stroke(0);
  fill(175);
  translate(width/2,height/2);
  line(0,0,x,0);
  ellipse(x,0,20,20);
}
```

Waves

Waves

- float angle = 0;
- float angleVel = 0.2;
- float amplitude = 100;
- 너비를 기준으로 주기를 만들어 보자.

Waves

```
for (int x = 0; x <= width; x += 24) {
```

1) Calculate the y location according to amplitude and sine of the angle.

```
    float y = amplitude*sin(angle);
```

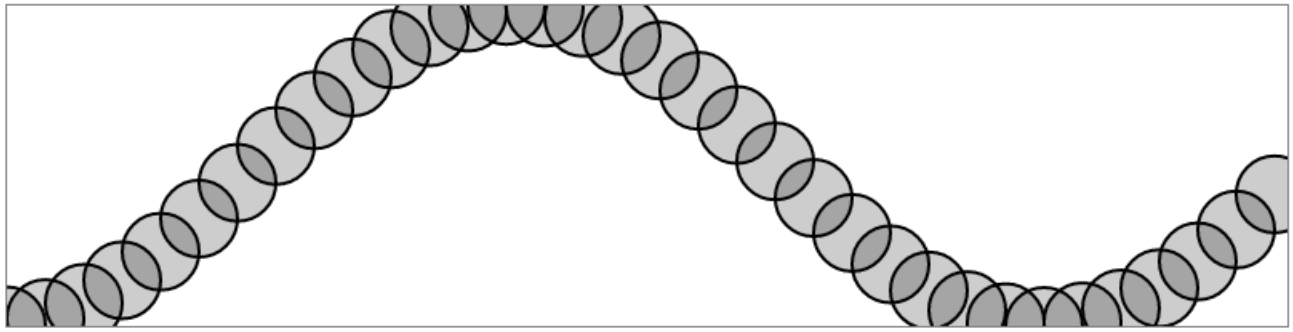
2) Draw a circle at the (x,y) location.

```
    ellipse(x,y+height/2,48,48);
```

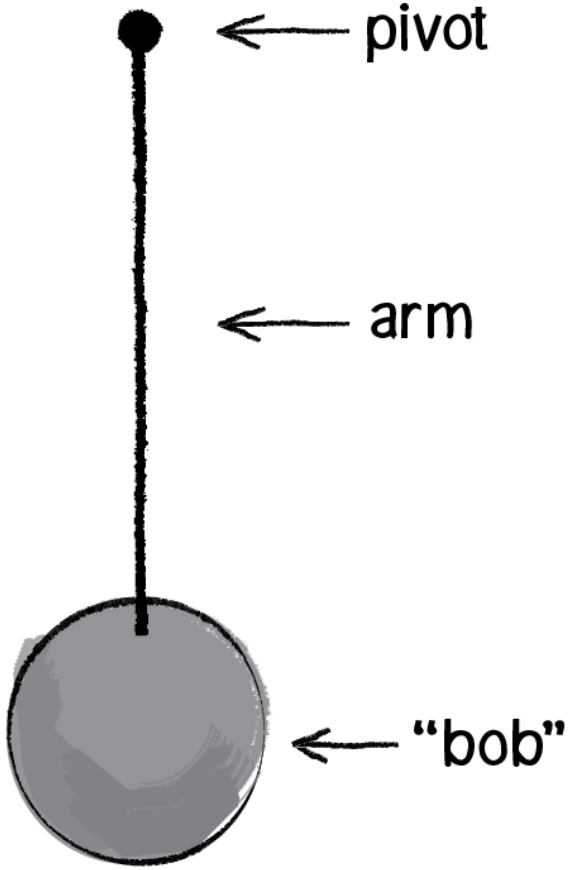
3) Increment the angle according to angular velocity.

```
    angle += angleVel;
  }
```

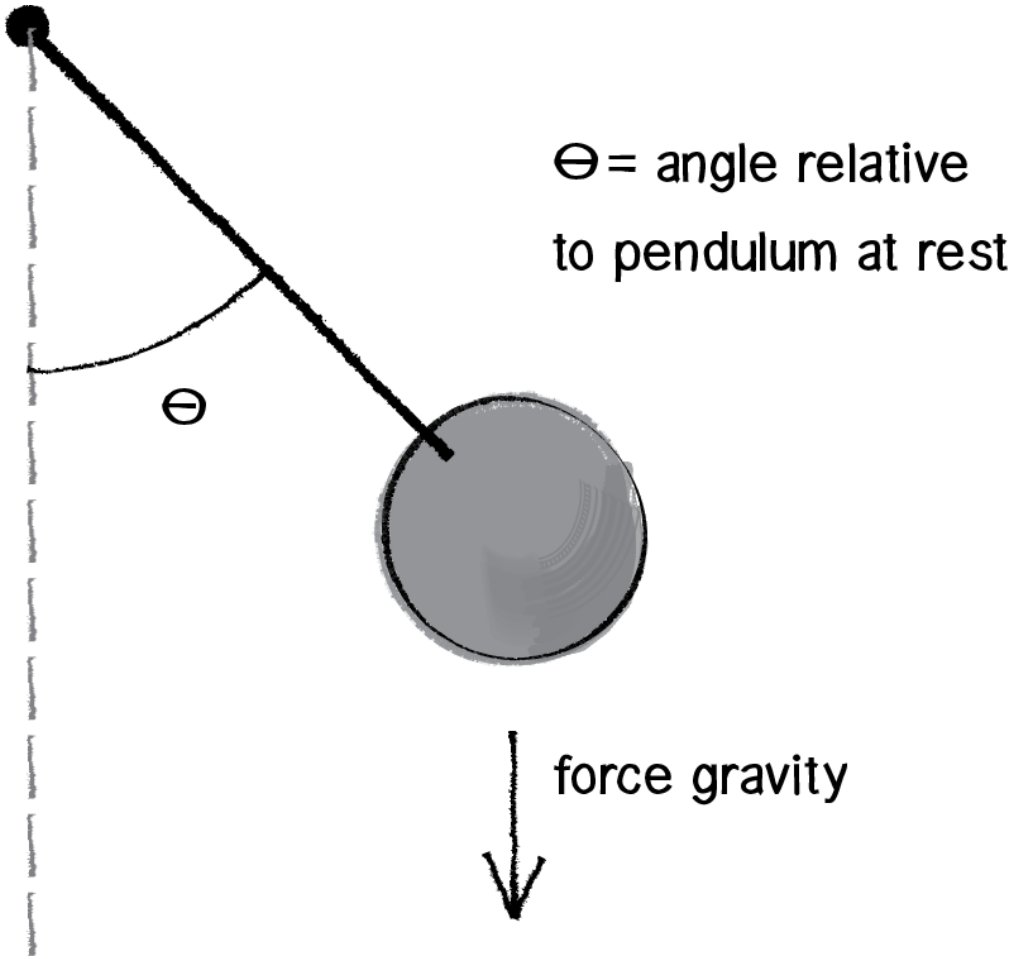
Moving Waves



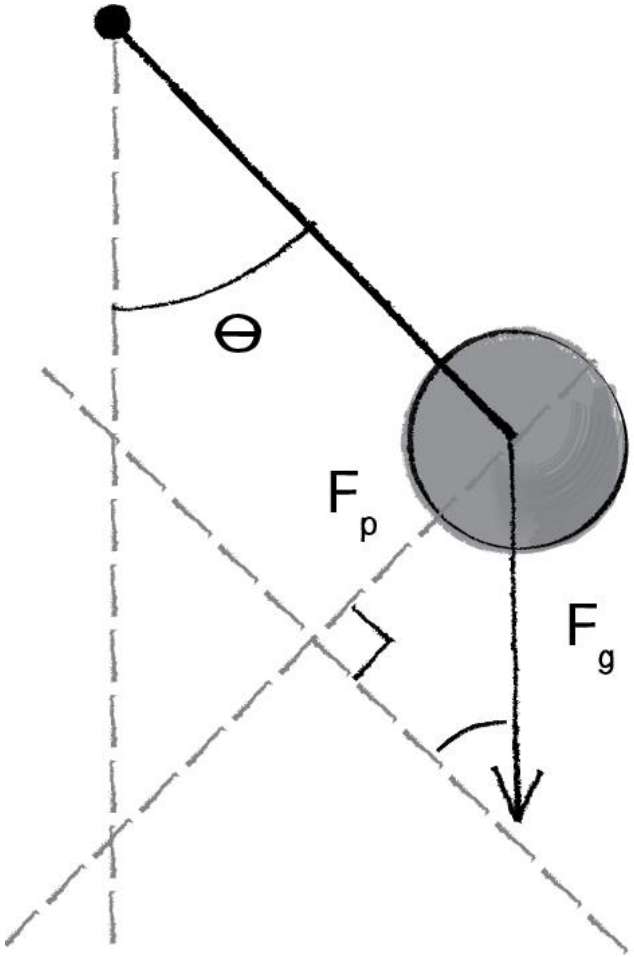
Trigonometry and Forces: The Pendulum

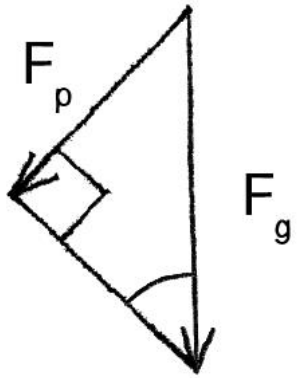


Trigonometry and Forces: The Pendulum



Trigonometry and Forces: The Pendulum





$\sin(\Theta) = F_g / F_p$

Trigonometry and Forces: The Pendulum

- $\sin(\theta) = Fp / Fg$
- $F_p = F_g * \sin(\theta)$

- pendulum angular acceleration = acceleration due to gravity * sin
- (θ)
- angular acceleration = gravity * $\sin(\theta)$

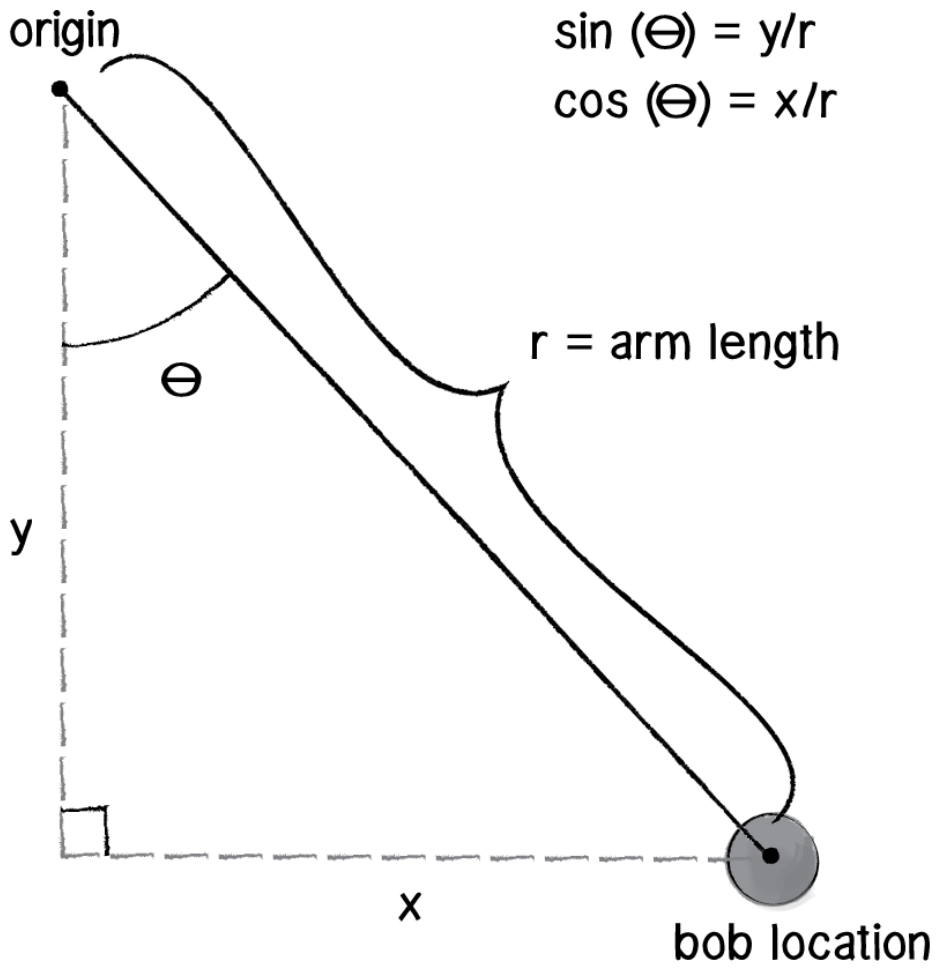
Pendulum Class

```
class Pendulum  {

float r;//Length of arm

float angle;//Pendulum arm angle

float aVelocity;//Angular velocity

float aAcceleration;//Angular acceleration
```

Pendulum Class

```
void update() {
  float gravity = 0.4;
  aAcceleration = -1 * gravity * sin(angle);
  aVelocity += aAcceleration;
  angle += aVelocity;
 }
```

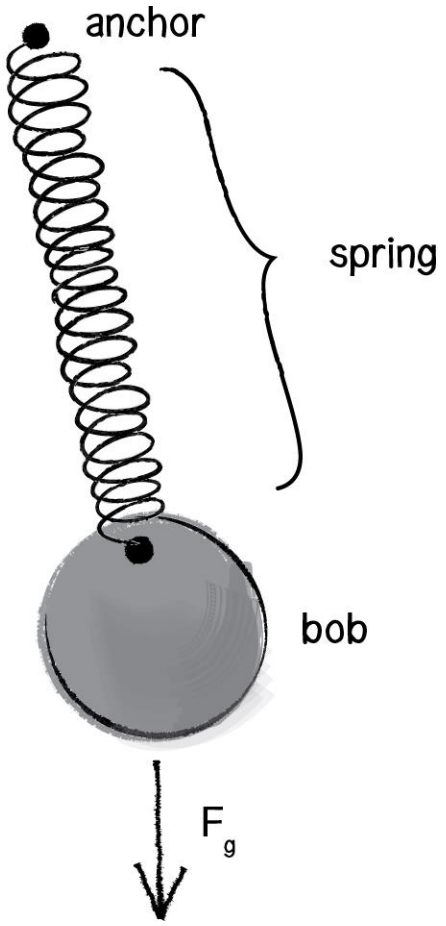
Pendulum Class



Pendulum Class

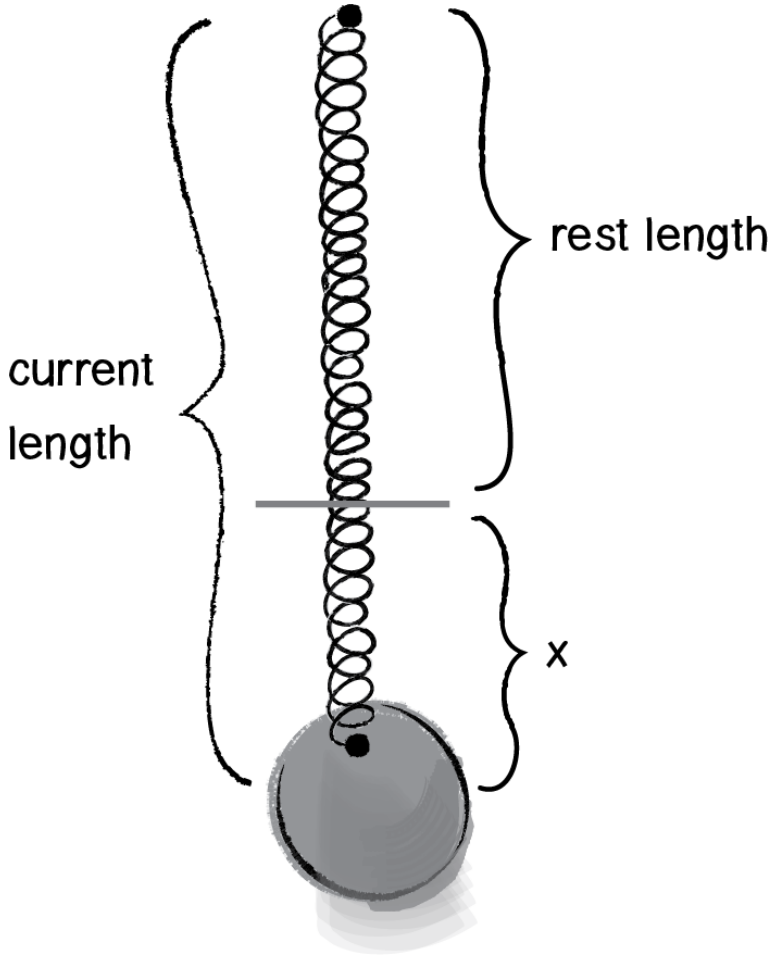
- PVector origin = new PVector(100,10);
- float r = 125;
- PVector location = new PVector(r*sin(angle),r*cos(angle));
- location.add(origin);
- aAcceleration = (-1 * G * sin(angle)) / r;

Spring Forces

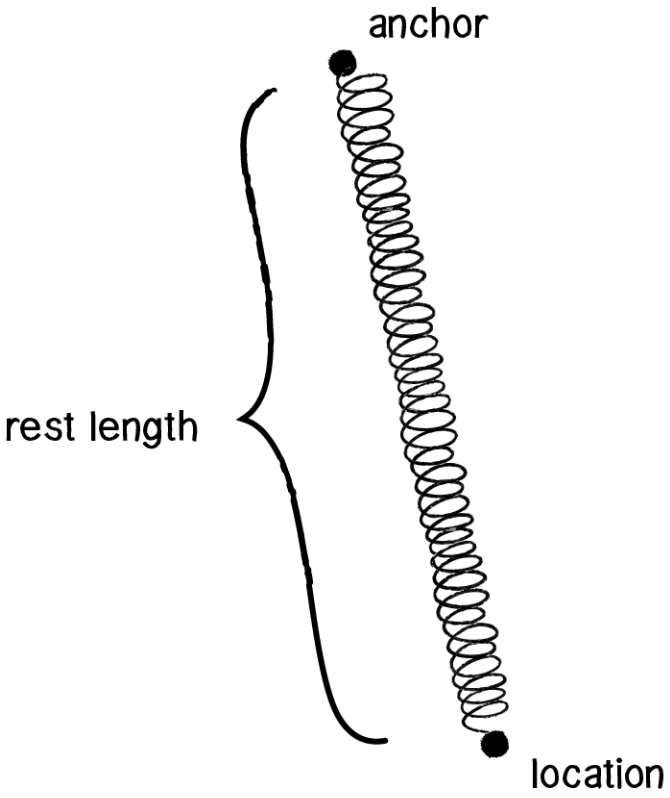


Hooke's law

- $F_{spring} = - k * x$



Spring Forces



Spring Forces

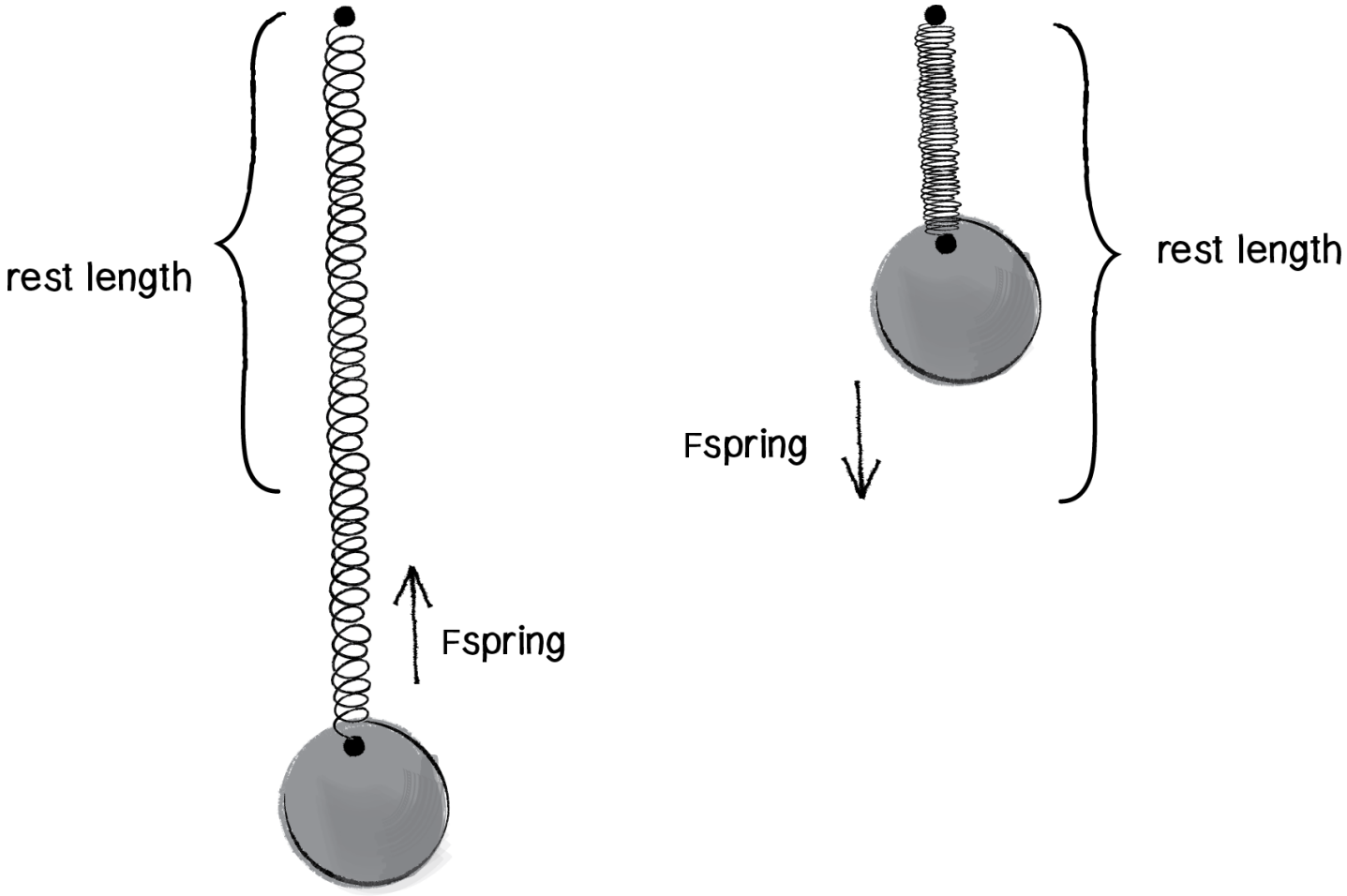
- PVector anchor;
- PVector location;
- float restLength;

- float k = 0.1

Spring Forces

```
PVector dir = PVector.sub(bob,anchor);
float currentLength = dir.mag();
float x = restLength - currentLength;
```

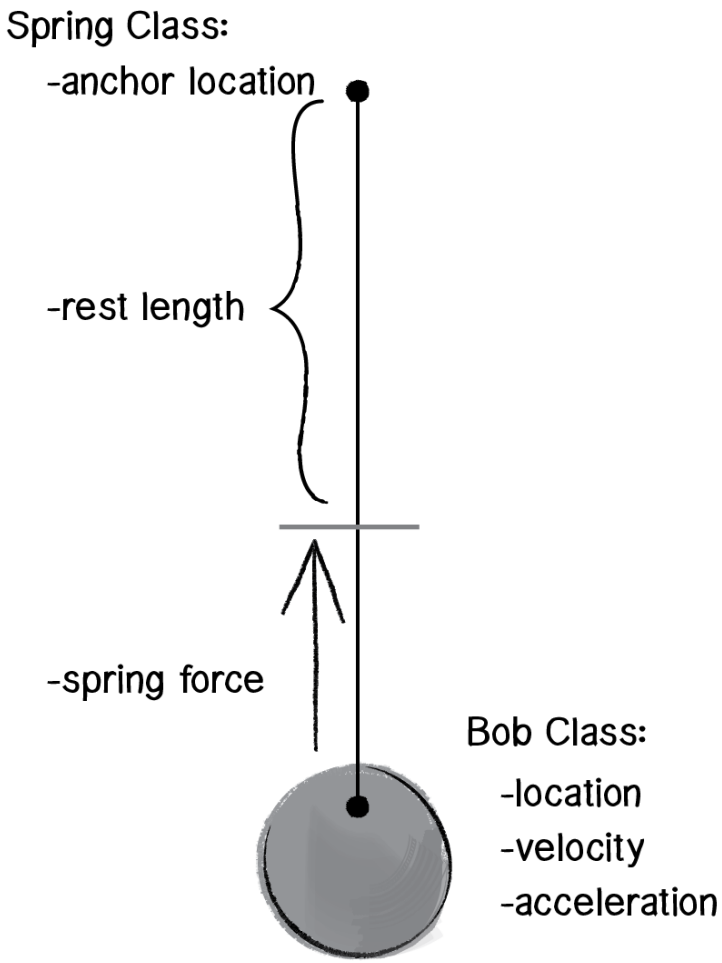
Spring Forces



Spring Forces

```
float k = 0.1;
PVector force = PVector.sub(bob,anchor);
float currentLength = dir.mag();
float x = restLength - currentLength;

//Direction of spring force (unit vector)
force.normalize();

//Putting it together: direction and magnitude!
force.mult(-1 * k * x);
```

Spring Class



Q&A