# 20170923 Ajou1

Jonghwa Park
suakii@gmail.com

GYEONGGI SCIENCE HIGH SCHOOL

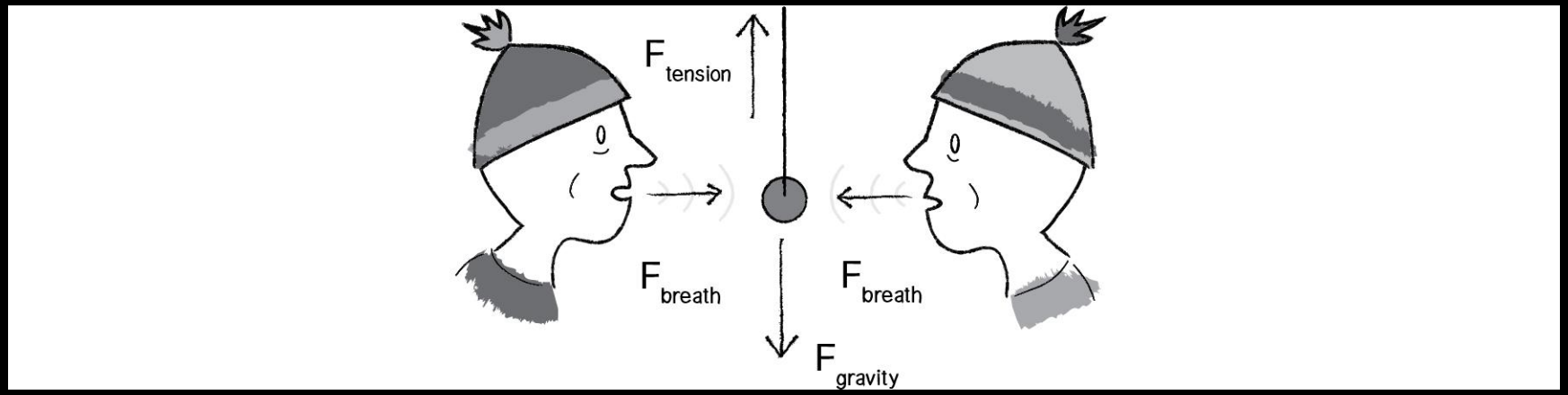
# Forces and Newton’s Laws of Motion

- A force is a vector that causes an object with mass to accelerate.
- 힘은 질량 있는 객체가 가속도를 갖게 하는 벡터.

# Newton’s First Law

- 다른 힘의 영향을 받지 않는다면 가만히 있는 물체는 계속 가만히 있고, 움직이는 물체는 같은 속도와 방향으로 계속 움직인다.

# Newton's First Law



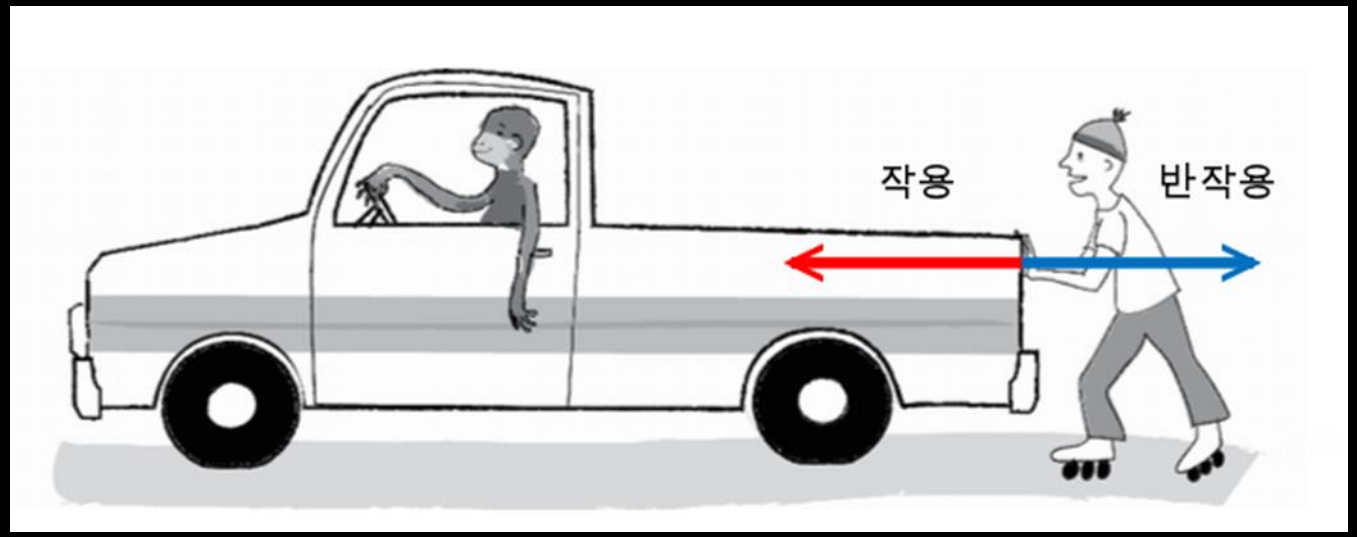
# Newton’s First Law

- An object’s PVector velocity will remain constant if it is in a state of equilibrium.

# Newton's Third Law

- 어떤 물체에 힘을 주면 반대 방향으로도 같은 힘이 일어난다.
- Forces always occur in pairs.
- **The two forces are of equal strength, but in opposite directions.**

# Newton's Third Law



# Newton’s Third Law

- **A라는 물체가 B라는 물체에 PVector f라는 힘을 준다면 B도 A에 PVector.mult(f,-1)만큼의 힘을 주어야 한다.**

# Newton's Second Law

- **Force equals mass times acceleration.**

$$F = ma$$

$$a = F/m$$

# Newton’s Second Law

- **The acceleration of an object is equal to force.**
- **Location is adjusted by velocity, and velocity by acceleration.**
- **Acceleration was where it all began.**
- **Force is truly where it all begins.**

# Newton's Second Law

```
class Mover {
  PVector location;
  PVector velocity;
  PVector acceleration;
}
```

# Newton’s Second Law

```
void applyForce(PVector force) {
 //Newton’s second law at its simplest.
//But this function has problem.
//What’s that?
  acceleration = force;
}

mover.applyForce(wind);
mover.appyForce(gravity);
```

# Force Accumulation

- **velocity.add(acceleration);**

```
void applyForce(PVector force) {
        acceleration.add(force);
 }
```

# Force Accumulation

```
if (mousePressed) {
  PVector wind = new PVector(0.5,0);
  mover.applyForce(wind);
}
```

Since we're adding all the forces together at any given moment, we have to make sure that we clear acceleration (i.e. set it to zero) before each time update() is called.

# Force Accumulation

```
void update() {
    velocity.add(acceleration);
    location.add(velocity);
    acceleration.mult(0);// 가속도를 0으로
}
```

# Dealing with Mass

```
class Mover {
  PVector location;
  PVector velocity;
  PVector acceleration;
  float mass;
}
```

# Dealing with Mass

```
Mover() {
    location = new
      PVector(random(width),random(height));
    velocity = new PVector(0,0);
    acceleration = new PVector(0,0);
    mass = 10.0;
 }
```

# Creating Forces

- Make up a force.
- Model a force

# Gravity on Earth and Modeling a Force

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

    PVector wind = new PVector(0.001,0);
    float m = movers[i].mass;
            //Scaling gravity by mass to be more accurate
    PVector gravity = new PVector(0,0.1*m);
    movers[i].applyForce(wind);
    movers[i].applyForce(gravity);

    movers[i].update();
    movers[i].display();
    movers[i].checkEdges();
  }
```

# Reference

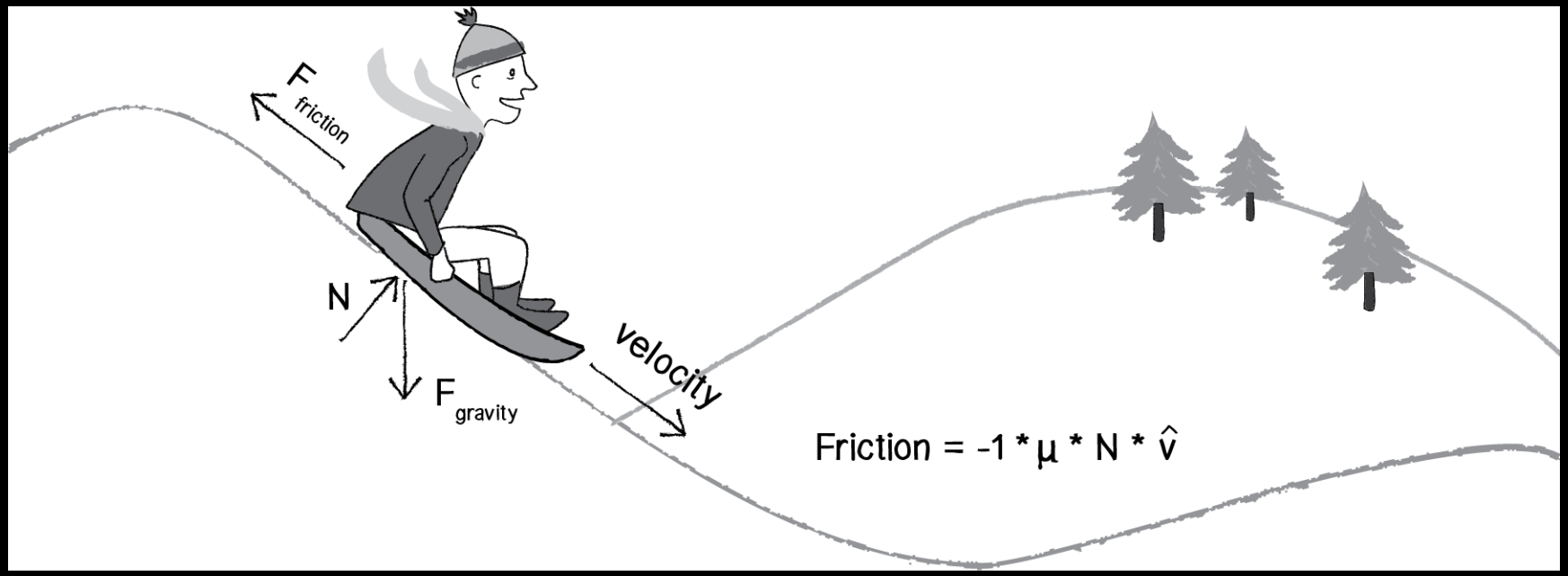
```
void applyForce(PVector force) {
//Making a copy of the PVector before //using it!
  PVector f = force.get();
  f.div(mass);
  acceleration.add(f);
}
```

# Test

# Friction

- 마찰력
  - 흩어지는 힘으로 물체가 움직일 때 전체 에너지를 감소 시키는 힘
- static friction
- kinetic friction

# Friction



# Friction

```
PVector friction = velocity.get();
friction.normalize();
friction.mult(-1);
```

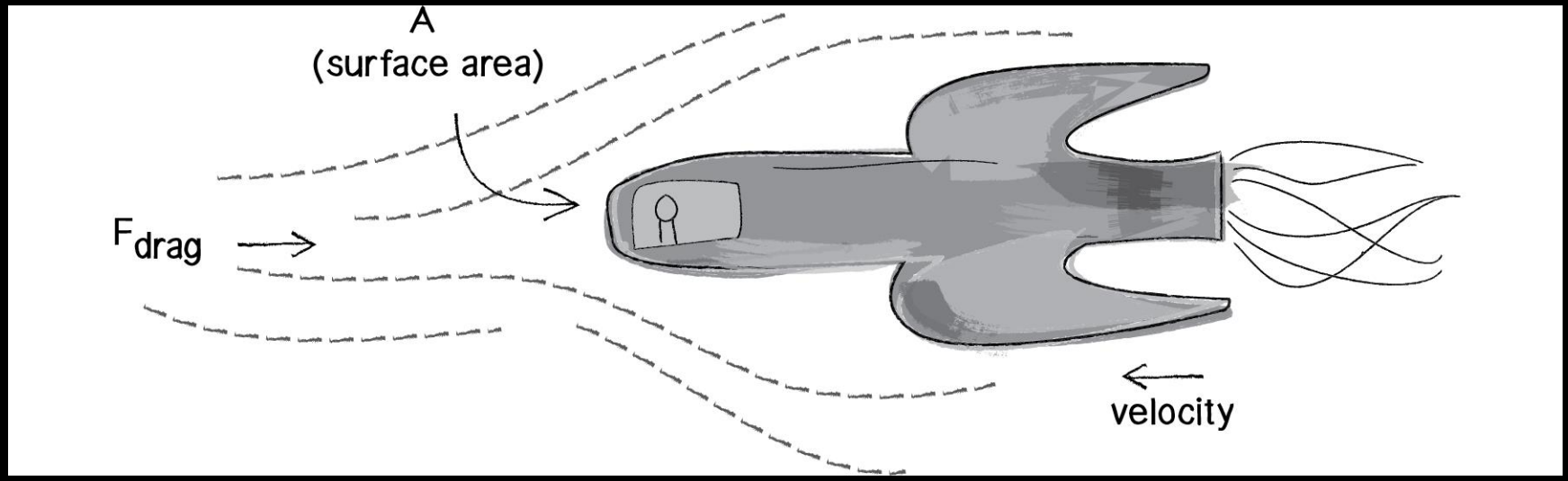
# Friction

- μ * N
- μ : coefficient of friction
- N : Normal force

```
float c = 0.01;
float normal = 1;
float frictionMag = c*normal;

PVector friction = velocity.get();
friction.mult(-1);
friction.normalize();
friction.mult(frictionMag);
```

# Air and Fluid Resistance



# Air and Fluid Resistance

$$F_d = -\frac{1}{2}\rho v^2 A C_d \hat{v}$$

Fd = 저항력
P = 액체밀도
V = 객체의 속력
A = 액체와 닿는 앞쪽 면적
Cd = 저항 계수
V^ = 속도의 방향 단위 벡터

# Air and Fluid Resistance

magnitude is speed squared * coefficent of drag

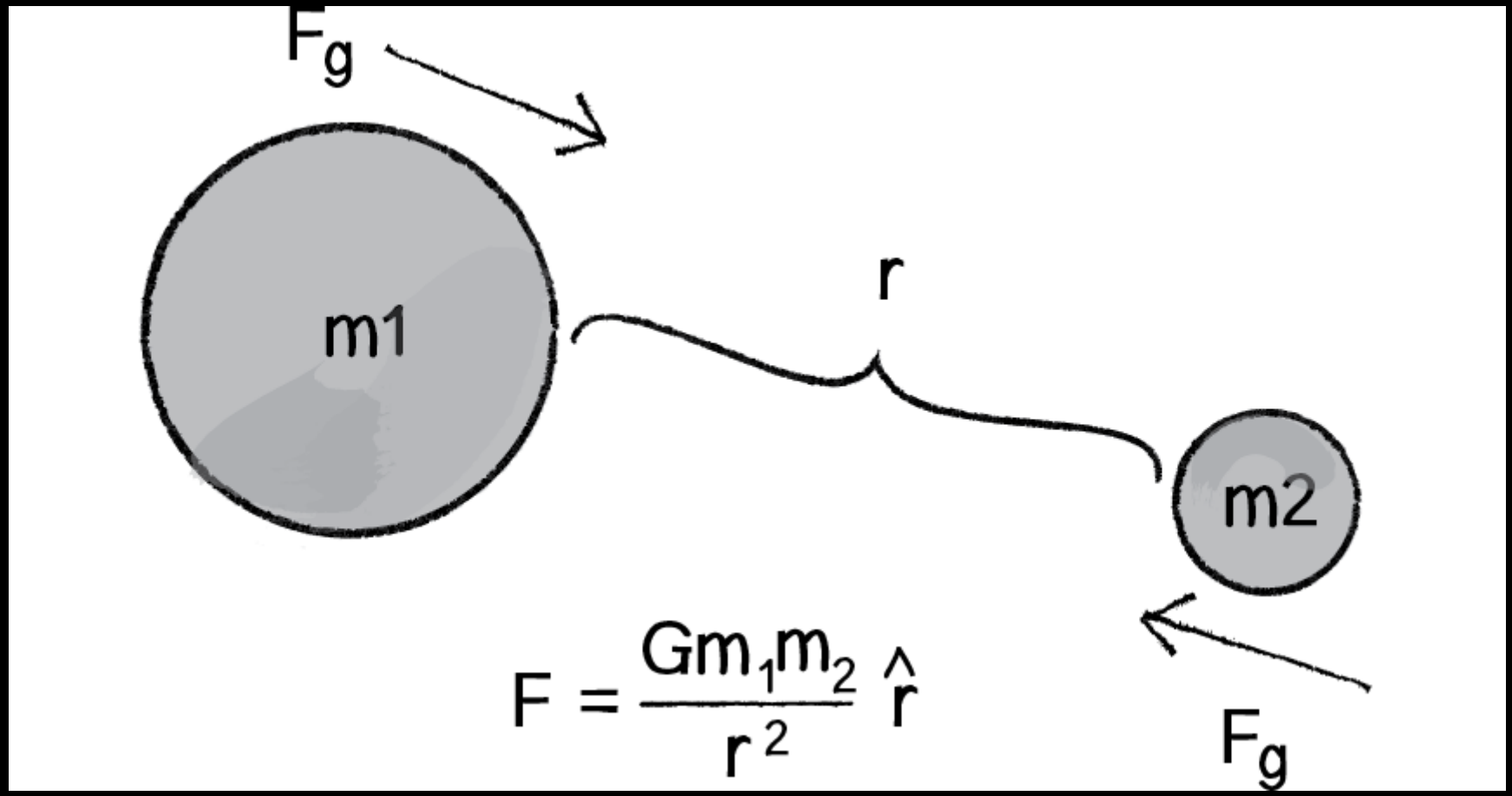
$$F_{drag} = \overbrace{\|v\|^2 * c_d}^{} * \underbrace{\hat{v} * -1}$$

direction is opposite of v (velocity)

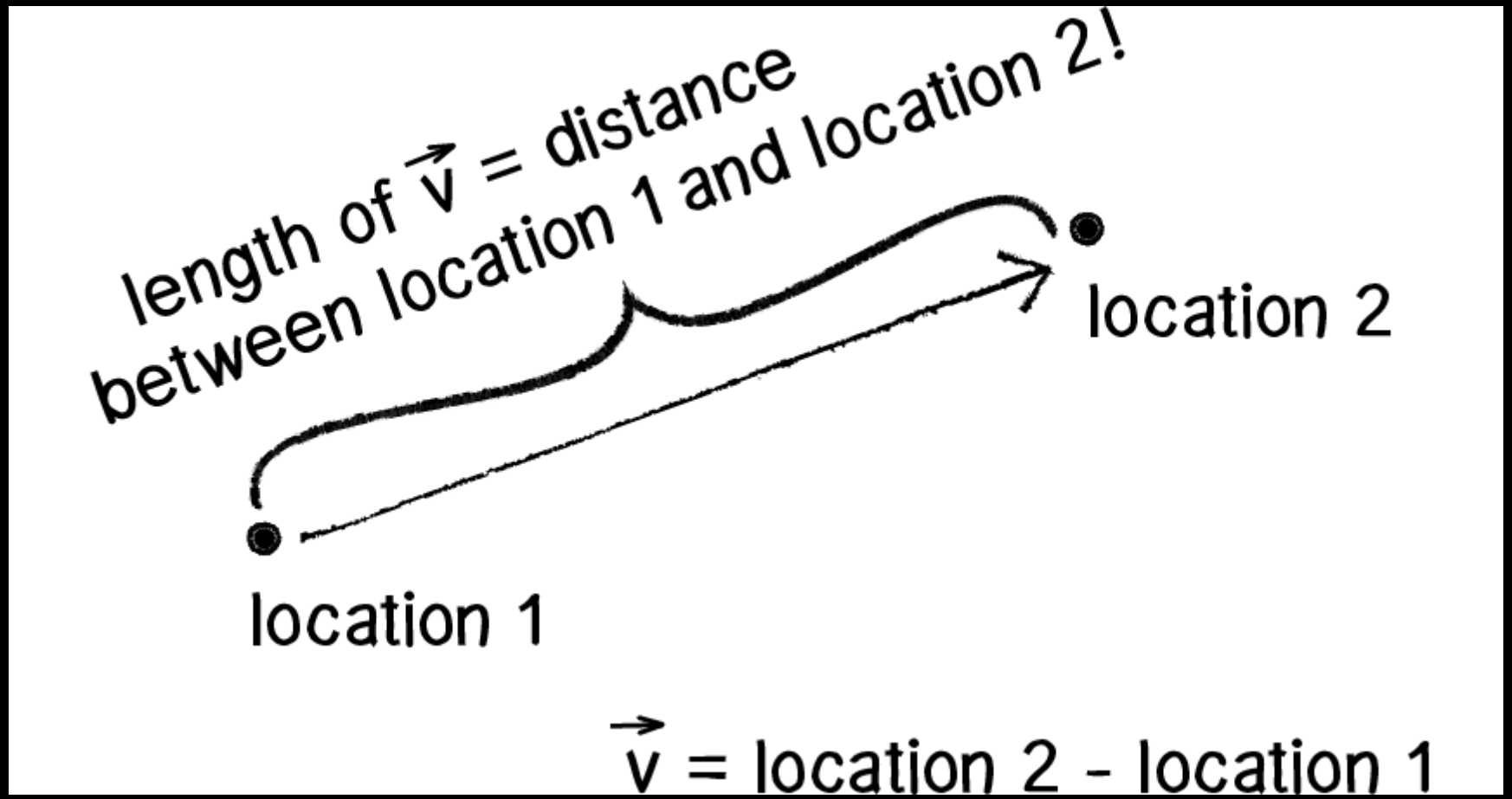
# Air and Fluid Resistance

```
float c = 0.1;
float speed = v.mag();
float dragMagnitude = c * speed * speed;
PVector drag = velocity.get();
drag.mult(-1);
drag.normalize();
drag.mult(dragMagnitude);
```

# Gravitational Attraction



# Gravitational Attraction



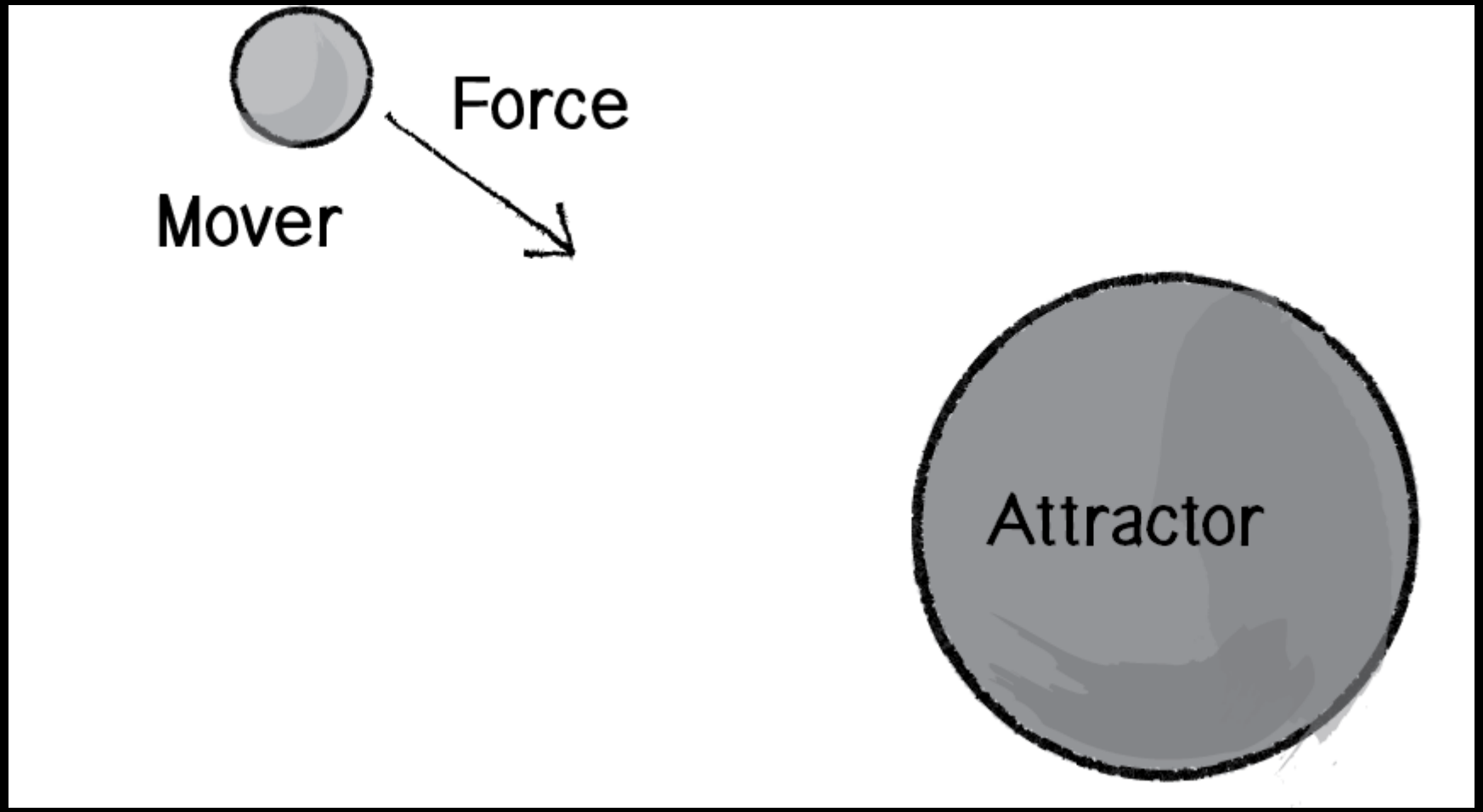
# Gravitational Attraction

```
PVector force = PVector.sub(location1,location2);

float distance = force.magnitude();
float m = (G * mass1 * mass2) / (distance * distance);

force.normalize();
force.mult(m);
```

# Gravitational Attraction



# Attractor

- Attractor is simple object that doesn’t move.
- We just need a mass and a location.

# Break…

- CU..