CMP 105 Games Programming

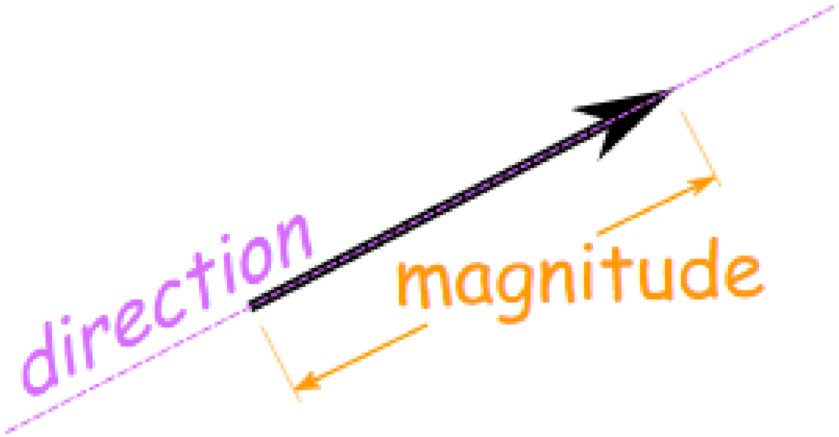
Gravity
(and other related maths, forces and movement)

This week

- Vectors
- Velocity
- Acceleration
- Gravity
- Applying forces
- Movement

Vectors

- A vector has a direction and magnitude (length)
- A vectors whose magnitude is 1
  - Is a unit or normalised vector
- A vector does not have a location
- In games vectors are used for:
  - Indicating a direction
    - Toward an enemy, light, perpendicular to a plane
    - Representing change (velocity of a moving object)



Static functions

- A non-static function can be called only after instantiating the class
- A static function can be called, even when a class is not instantiated
- Can only access
  - Static data/variables
  - Other static functions
  - Data and functions outside the class
- Static function cannot have access the this pointer of the class

Static functions

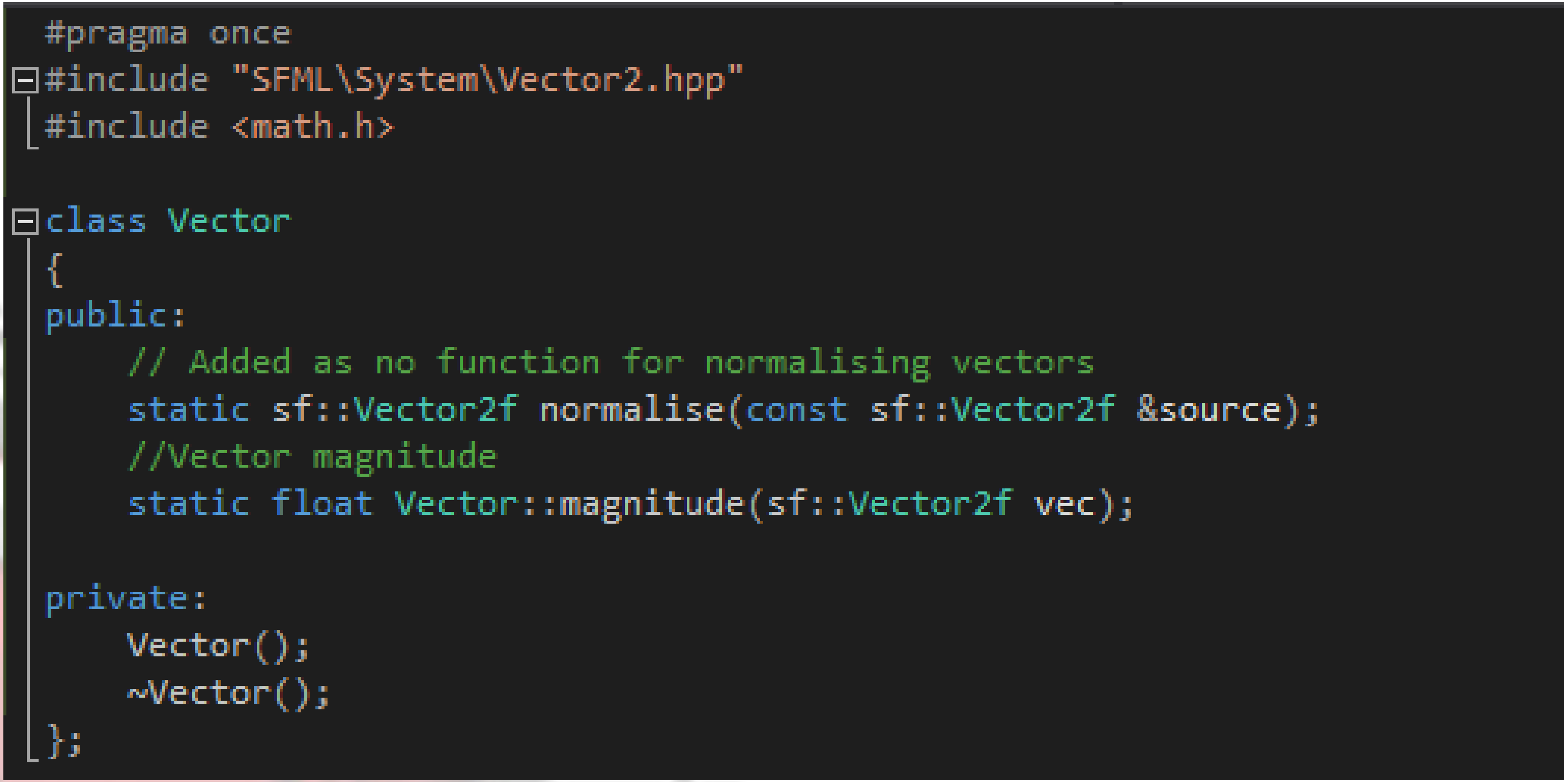
```
class myClass
{
      public:
      static void myFunction();
}

void someFunction()
{
      ...
      myClass::myFunction();
      ...
}
```

Static functions

- Standard SFML Vector class is lacking a few functions
- Mainly
    - Normalise
    - Magnitude
- I’ve put together a simple small class with some static functions to provide these
    - Easy to add to current projects
    - Provides functionality
    - Simple functionality you could implement yourself (if you want)

Vector.h

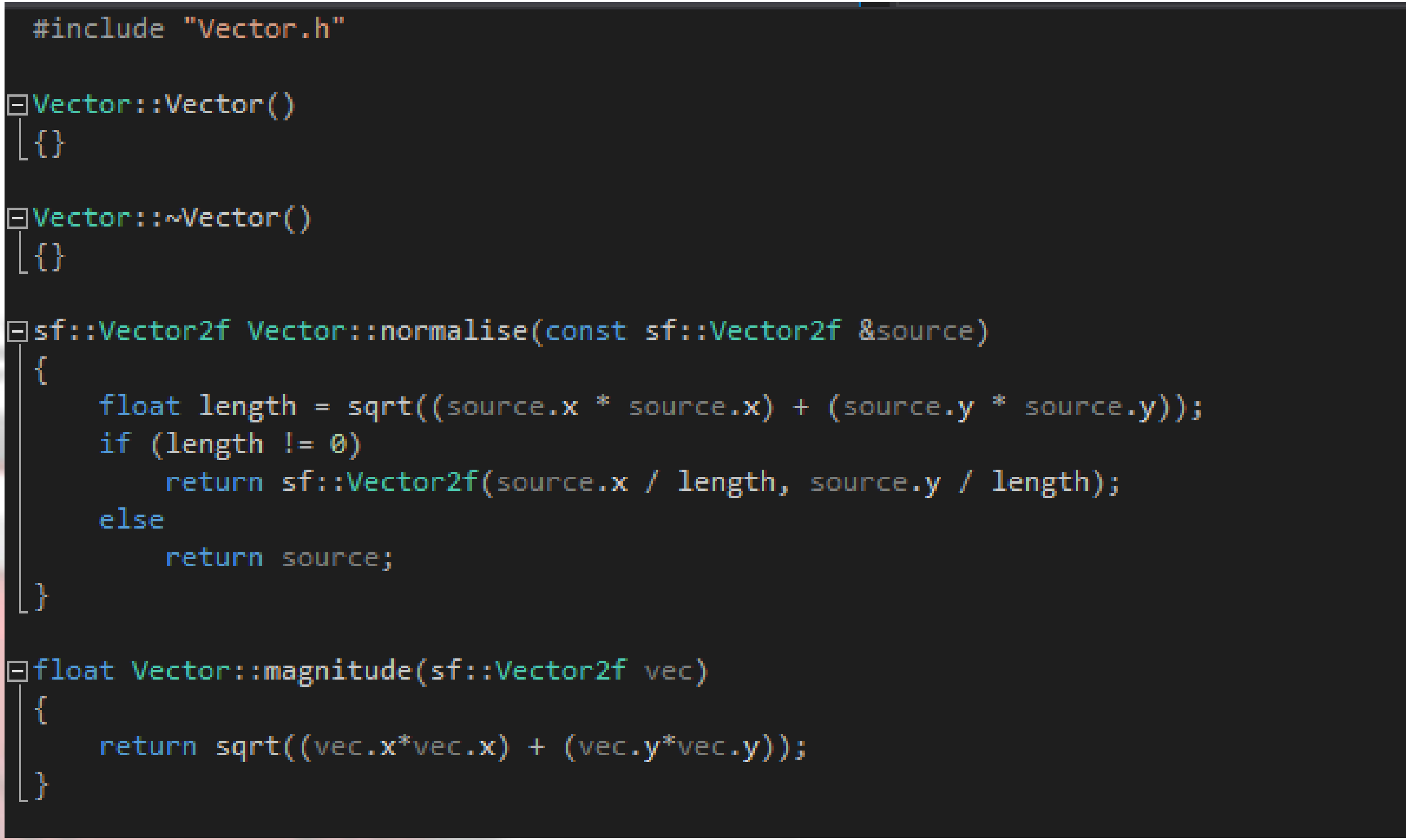


```
#pragma once
#include "SFML\System\Vector2.hpp"
#include <math.h>

class Vector
{
public:
    // Added as no function for normalising vectors
    static sf::Vector2f normalise(const sf::Vector2f &source);
    //Vector magnitude
    static float Vector::magnitude(sf::Vector2f vec);

private:
    Vector();
    ~Vector();
};
```

Vector.cpp



```
#include "Vector.h"

Vector::Vector()
{}

Vector::~Vector()
{}

sf::Vector2f Vector::normalise(const sf::Vector2f &source)
{
    float length = sqrt((source.x * source.x) + (source.y * source.y));
    if (length != 0)
        return sf::Vector2f(source.x / length, source.y / length);
    else
        return source;
}

float Vector::magnitude(sf::Vector2f vec)
{
    return sqrt((vec.x*vec.x) + (vec.y*vec.y));
}
```

Velocity

- A 2D vector representing the direction and speed (magnitude) of an object
- We've already been working with this

```
velocity.x = 5.f;
move(velocity*dt);
```

- Or

```
velocity = sf::Vector2f(4.f, 2.f);
move(velocity*dt);
```

Acceleration

- The rate of change of velocity (with respect to time)
- Basic concept
  - Object moving along at a certain velocity an acceleration amount is continuously added to the velocity
  - Friction is the opposite of acceleration, subtracting an mount

```
acceleration = 2.f;
velocity.x = velocity.x + acceleration;
move(velocity);
```

Gravity

- Constant acceleration in a downward direction
- Important
  - You only want to apply gravity to objects in the air
  - Need to detect if object is on the ground/moving/falling etc and apply gravity as required
- E.g.

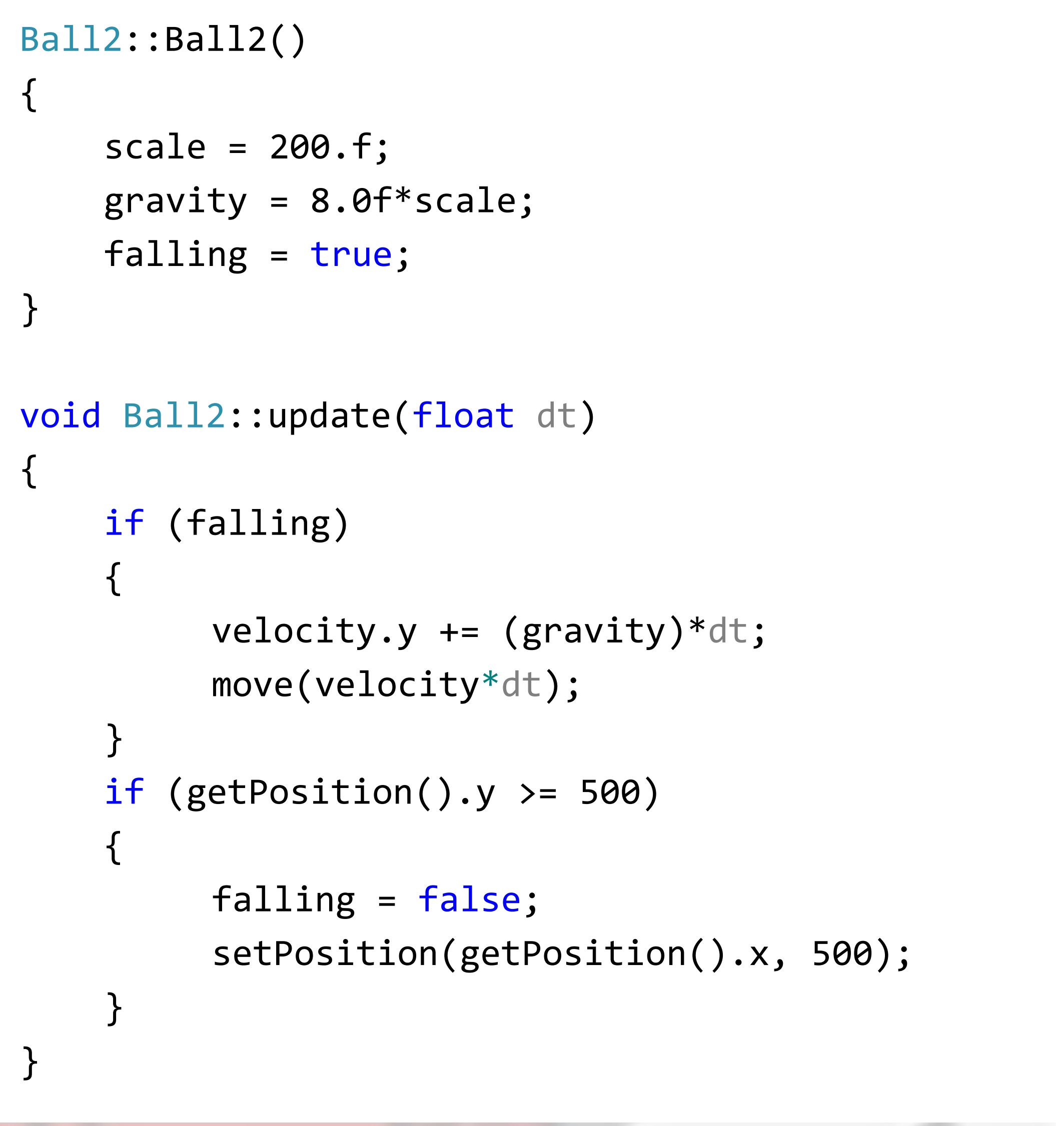
Gravity

```
if(object.falling == true)
{
    velocity.y velocity.y + gravity;
    move(velocity);
    if(object.position > 500)
    {
        // object has hit or passed floor
        velocity.y = 0;
        object.falling = false;
        object.position.y = 500;
    }
}
```

Working with delta time

- Previous example is a little misleading
  - Doesn’t use delta time
  - Requires fixed framerate
  - Values are unrealistic
- When using delta time we deal with pixels per second
  - Meaning velocity and gravity values are going to be quite large
  - We must multiply acceleration AND velocity by delta time

Working with delta time



```
Ball2::Ball2()
{
    scale = 200.f;
    gravity = 8.0f*scale;
    falling = true;
}

void Ball2::update(float dt)
{
    if (falling)
    {
        velocity.y += (gravity)*dt;
        move(velocity*dt);
    }
    if (getPosition().y >= 500)
    {
        falling = false;
        setPosition(getPosition().x, 500);
    }
}
```

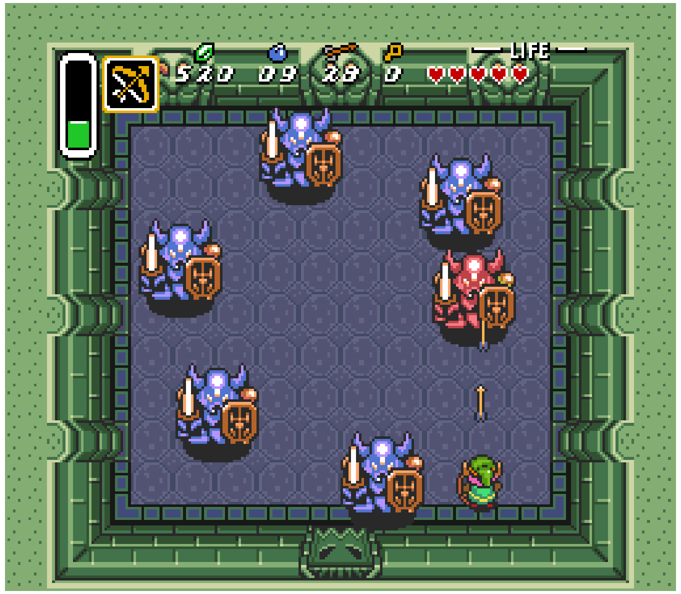
Applying forces

- For example Mario jumping, launching an angry bird, bouncing of a spring etc etc
- On key press or in-game event provide a new velocity value (direction and magnitude)
  - You may want to limit when these events happen
  - For example a player can only jump when on the ground

```
if (input->isKeyDown(sf::Keyboard::Space))
{
   velocity.y = -2.f*scale;
   falling = true;
}
```

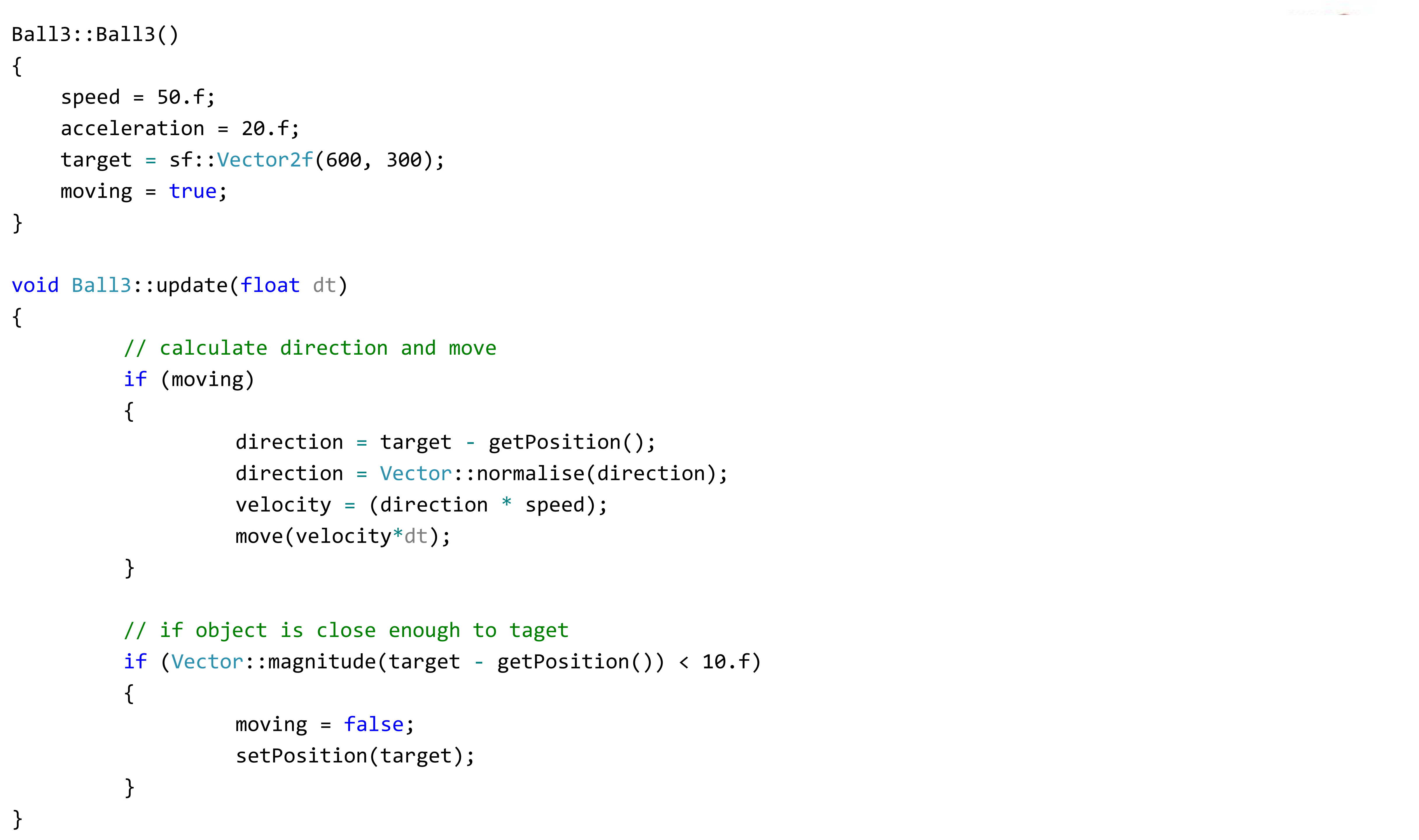
Movement

- Could be as simple as setting a velocity value on key press
  - If “right” is pressed
  - Velocity.x = 5.f
  - Move(velocity*dt);
- Could be more complex by applying a friction
  - While character is on the ground



Movement

- What about more complex or automatic movement?
  - Not just along the x-axis, to specific location, not controlled by the player
- Move from Point A to Point B (in a straight line, at a set speed)
  - Build vector (Point B) – (Point A)
  - Normalise the vector
  - Velocity = (vector * speed) * dt;
- Works if Point B moves or changes
- Need to detect when object reaches Point B and stop movement



```
Ball3::Ball3()
{
    speed = 50.f;
    acceleration = 20.f;
    target = sf::Vector2f(600, 300);
    moving = true;
}

void Ball3::update(float dt)
{
        // calculate direction and move
        if (moving)
        {
                direction = target - getPosition();
                direction = Vector::normalise(direction);
                velocity = (direction * speed);
                move(velocity*dt);
        }

        // if object is close enough to taget
        if (Vector::magnitude(target - getPosition()) < 10.f)
        {
                moving = false;
                setPosition(target);
        }
}
```

Live demo

- A few examples
  - Bouncing ball
  - Jumping ball
  - Moving ball

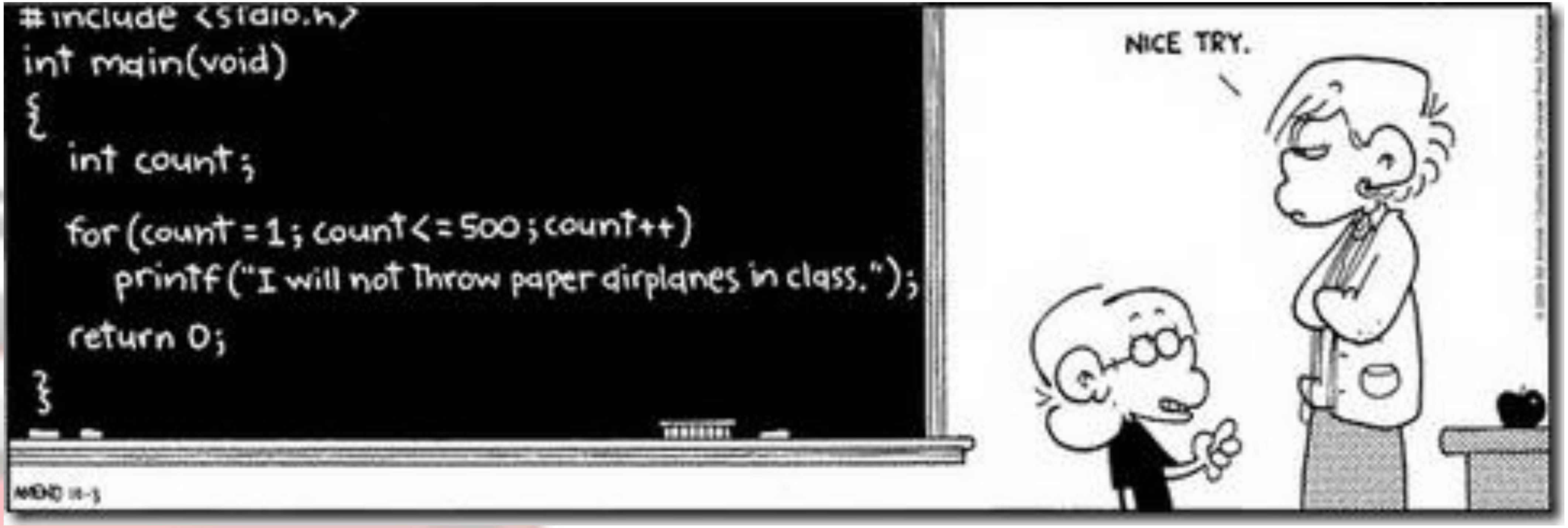
Important notice!

- Next week
  - Week 7 no lecture or labs
  - Special task like last semester
  - Normal class resume week 8 (week starting 27th Feb)
- Mid-term surveys
  - I will provide time in lab to complete them
  - These are important and very helpful. Please

In the labs

- Building some objects with forces and gravity and stuff
- Maths is fun (vectors reading)
  - https://www.mathsisfun.com/algebra/vectors.html

- Get into the habit of bringing pen and paper
  - Draw/think out the problems
  - I can leave doddles to help
  - Diagrams help!
  - Good practice



```
#include <stdio.h>
int main(void)
{
  int count;

  for(count=1; count<=500; count++)
     printf("I will not Throw paper airplanes in class.");
  return 0;
}
```