# Math

### *Objective*

You will understand different math functions and how they are used in games

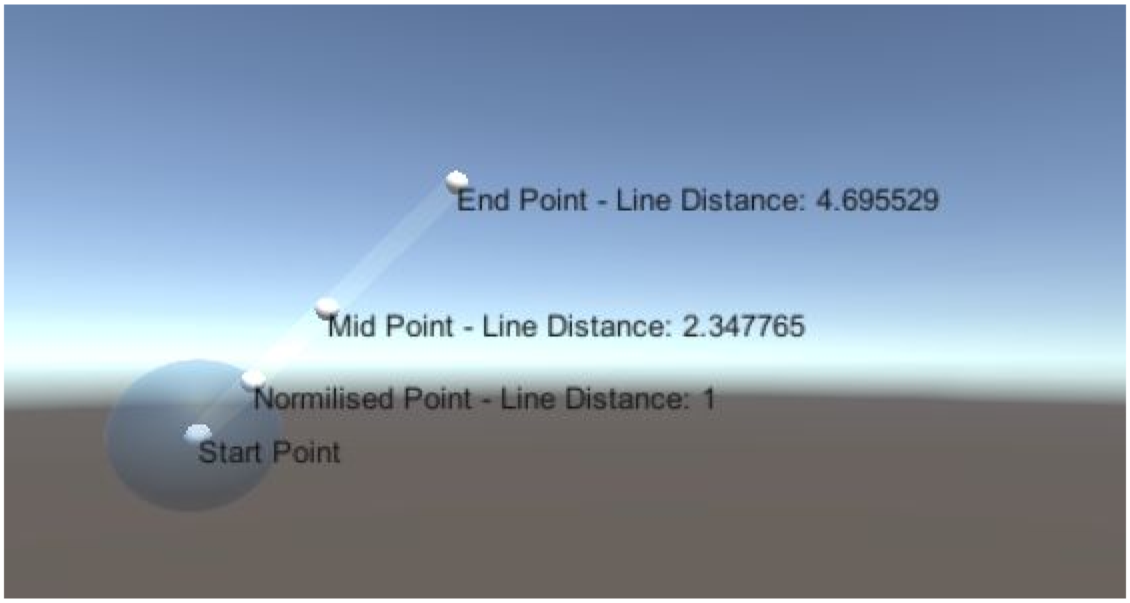
## Vectors

We are going to create a toy scene to demonstrate different math functions

### *Setup*

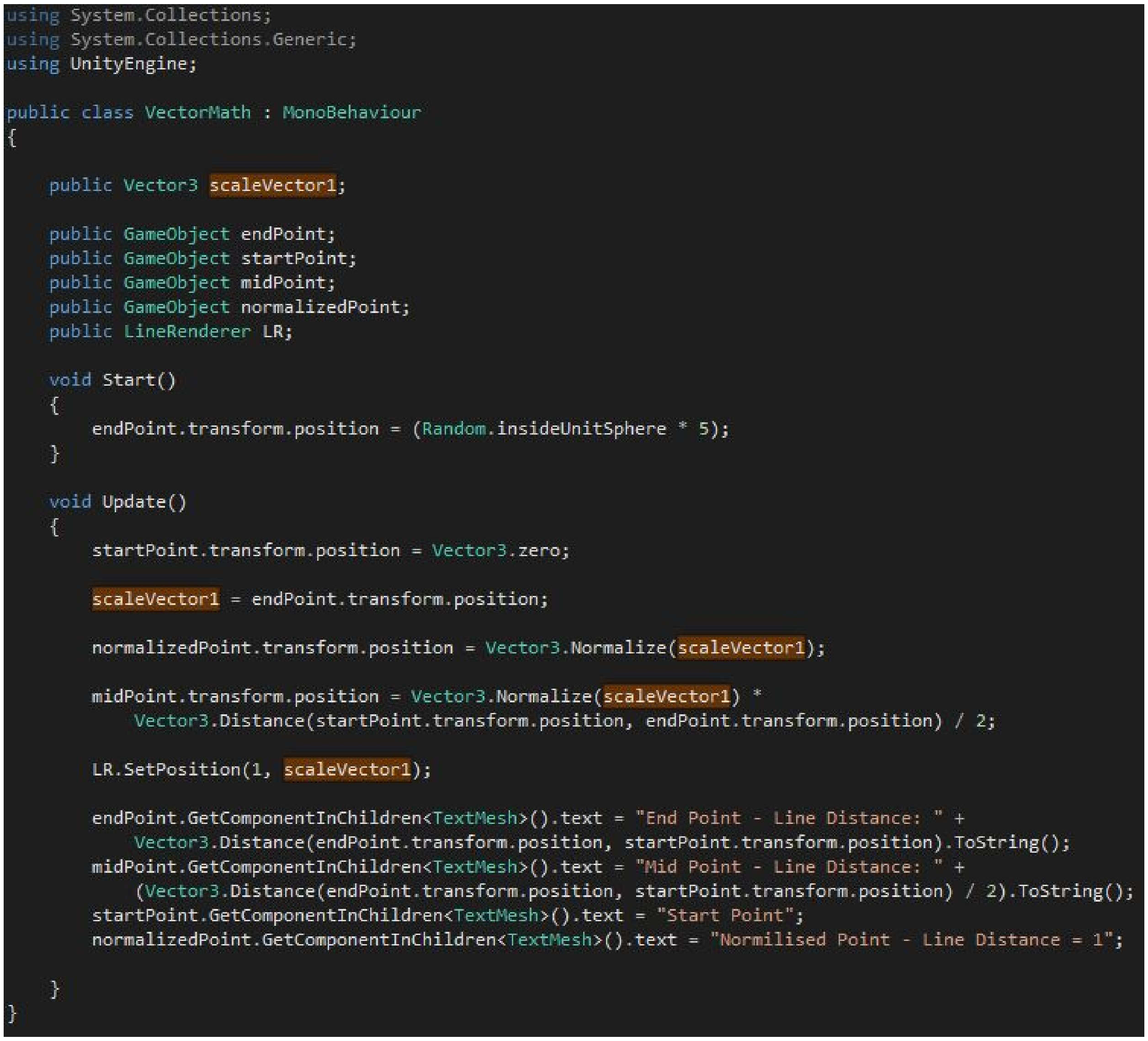
Create a new scene.

- Make a spheres with a scale of 0.25,0.25,0.25
  - Create an empty object as a child of the sphere and rename it Text
  - Add a '**Text mesh**' component to the empty object
- Duplicate the sphere 3 times and name each
  - Start Point
  - End Point
  - Mid Point
  - Normal Point
- With the Start Point sphere, Add a line render component to it and give it a material and a width of 0.25


End Point - Line Distance: 4.695529
Mid Point - Line Distance: 2.347765
Normilised Point - Line Distance: 1
Start Point

### *Lesson*

Create a new script called ‘**`VectorMath`**’


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class VectorMath : MonoBehaviour
{

    public Vector3 scaleVector1;

    public GameObject endPoint;
    public GameObject startPoint;
    public GameObject midPoint;
    public GameObject normalizedPoint;
    public LineRenderer LR;

    void Start()
    {
        endPoint.transform.position = (Random.insideUnitSphere * 5);
    }

    void Update()
    {
        startPoint.transform.position = Vector3.zero;

        scaleVector1 = endPoint.transform.position;

        normalizedPoint.transform.position = Vector3.Normalize(scaleVector1);

        midPoint.transform.position = Vector3.Normalize(scaleVector1) *
            Vector3.Distance(startPoint.transform.position, endPoint.transform.position) / 2;

        LR.SetPosition(1, scaleVector1);

        endPoint.GetComponentInChildren<TextMesh>().text = "End Point - Line Distance: " +
            Vector3.Distance(endPoint.transform.position, startPoint.transform.position).ToString();
        midPoint.GetComponentInChildren<TextMesh>().text = "Mid Point - Line Distance: " +
            (Vector3.Distance(endPoint.transform.position, startPoint.transform.position) / 2).ToString();
        startPoint.GetComponentInChildren<TextMesh>().text = "Start Point";
        normalizedPoint.GetComponentInChildren<TextMesh>().text = "Normilised Point - Line Distance = 1";

    }
}
```

Once done, you should be able to press play and move the endpoint around (in the Scene view) and watch all the calculations change.

Think about how you could use these if you were calculating things like points between the player and the enemy.

## Rotations

Because rotations are so hard and so complex, here are a couple of toys that you can make to understand them a little better and reference back to when they need to.

## *Setup*

Create a new scene.
Add the provided SpacePackage into your project

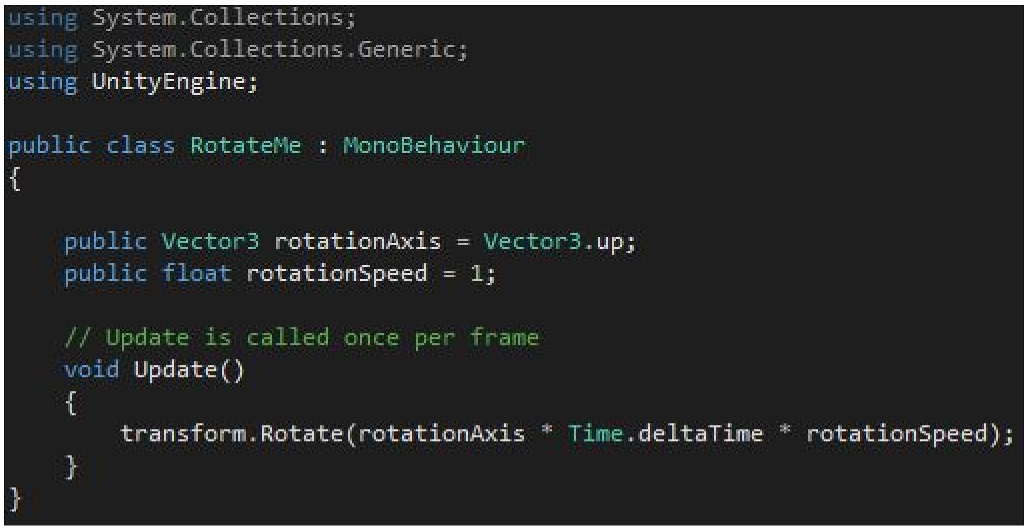
Make a sphere and name it ‘**Earth**’

## *Lesson*

We are going to make a scene that uses several rotation scripts.

Use the earth texture provided to make your earth sphere look good.
To make the earth spin, we change the X,Y and Z of the rotation it its transform.

let’s write a script to do this for us called **‘RotateMe’**



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class RotateMe : MonoBehaviour
{

    public Vector3 rotationAxis = Vector3.up;
    public float rotationSpeed = 1;

    // Update is called once per frame
    void Update()
    {
        transform.Rotate(rotationAxis * Time.deltaTime * rotationSpeed);
    }
}
```

This is a very simple script that is very useful

Place it on the earth sphere and press Play.

Next make a smaller sphere (canned ‘Moon’ at 0.2 scale and texture it accordingly). Place it alongside the Earth sphere.

Now how do you think you will make the moon rotate around the earth?

Make a gameobject called MoonPivot and parent it to the Earth.
Then Parent the Moon to the MoonPivot
Then place the RotateMe script on the moon pivot.

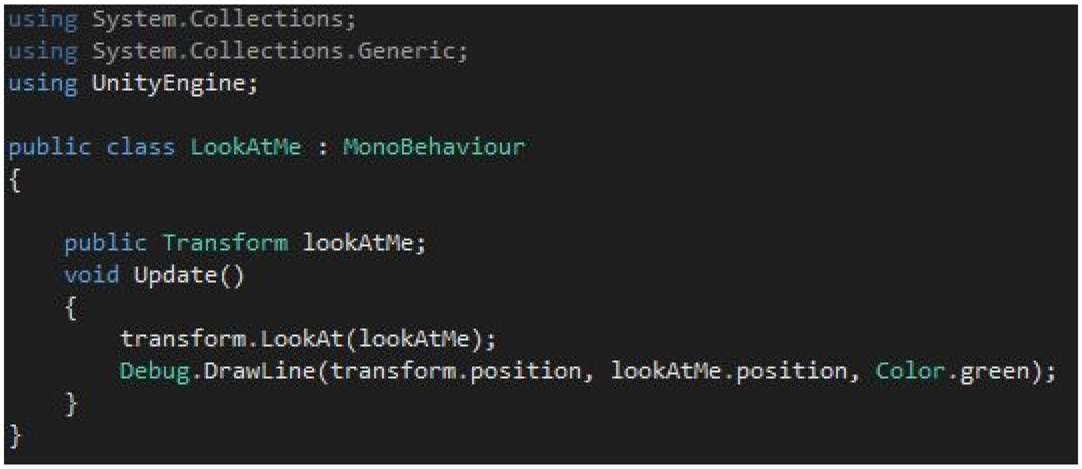
Press Play

With this knowledge of nesting rotations, Make a Sun for the Earth to go around and then using the provided Rocket prefab, make a rocket orbit the moon.

To make the scene better you can remove all the light from the scene, Add a large point  light in the Sun and use the Star skybox and Sun flair.
Set the sun gameobject to the ‘Ignore Raycast’ layer to make the sun flare work.

Once done you should have a little orbiting toy.

Make another script called ‘LookAtMe’



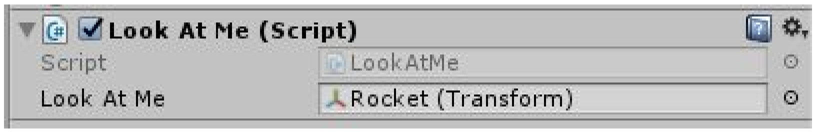
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LookAtMe : MonoBehaviour
{

    public Transform lookAtMe;
    void Update()
    {
        transform.LookAt(lookAtMe);
        Debug.DrawLine(transform.position, lookAtMe.position, Color.green);
    }
}
```

Place this script on the Main camera also and have it look at the little rocket whizzing around the moon.

Make a second rocket and have it placed near the camera,
Make the camera’s ‘LookAtMe’ script look at this new rocket.

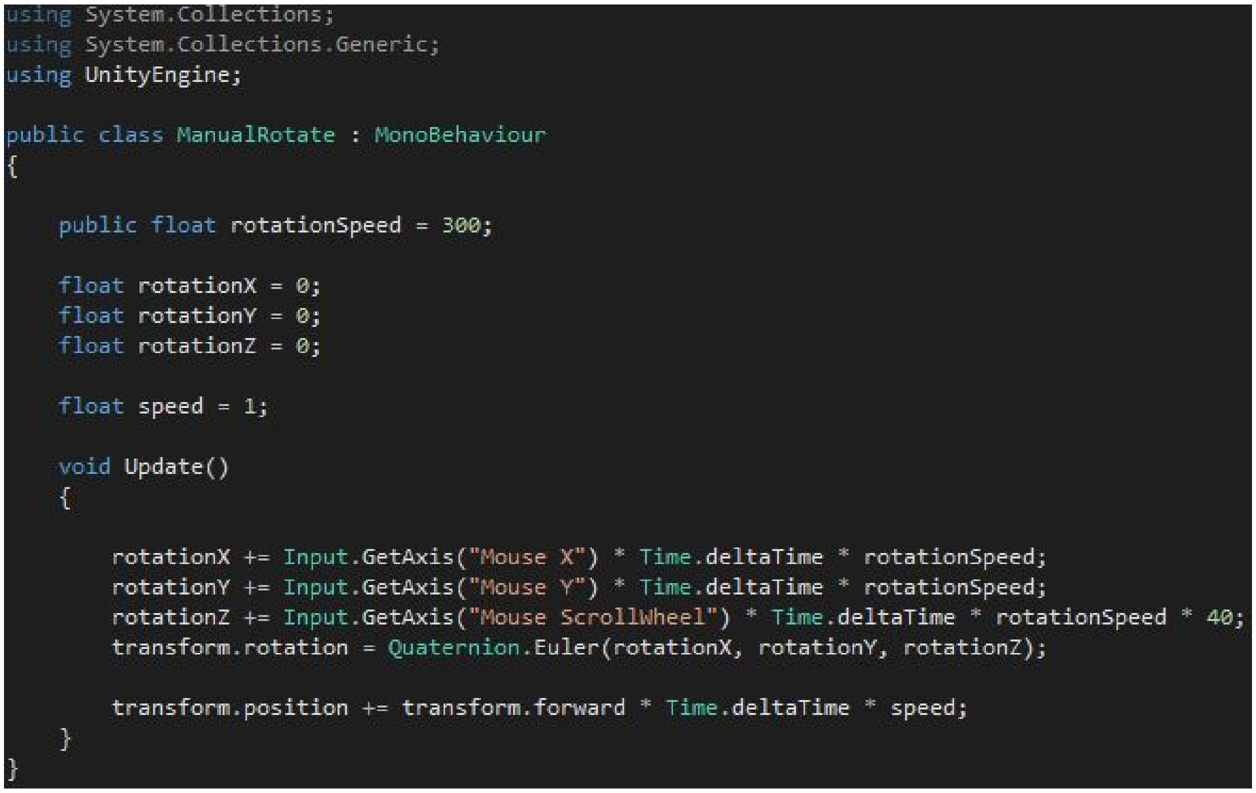


Look At Me (Script)

| | |
|---|---|
| Script | LookAtMe |
| Look At Me | Rocket (Transform) |

Press Play

Make a new script called ‘ManualRotate’ and place it on the rocket



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ManualRotate : MonoBehaviour
{

    public float rotationSpeed = 300;

    float rotationX = 0;
    float rotationY = 0;
    float rotationZ = 0;

    float speed = 1;

    void Update()
    {

        rotationX += Input.GetAxis("Mouse X") * Time.deltaTime * rotationSpeed;
        rotationY += Input.GetAxis("Mouse Y") * Time.deltaTime * rotationSpeed;
        rotationZ += Input.GetAxis("Mouse ScrollWheel") * Time.deltaTime * rotationSpeed * 40;
        transform.rotation = Quaternion.Euler(rotationX, rotationY, rotationZ);

        transform.position += transform.forward * Time.deltaTime * speed;
    }
}
```

Create a new rocket and place this script on it and press play.

Note: Depending how you want to control your rocket, you might need to swap Mouse Axis with rotation axis. Try these changes, and observe the results in-game:

**rotationX += Input.GetAxis(“Mouse Y”) * Time.deltaTime * rotationSpeed;**
**rotationY += Input.GetAxis(“Mouse X”) * Time.deltaTime * rotationSpeed;**

Save this entire scene.