Let's get our Hands Dirty

Objective

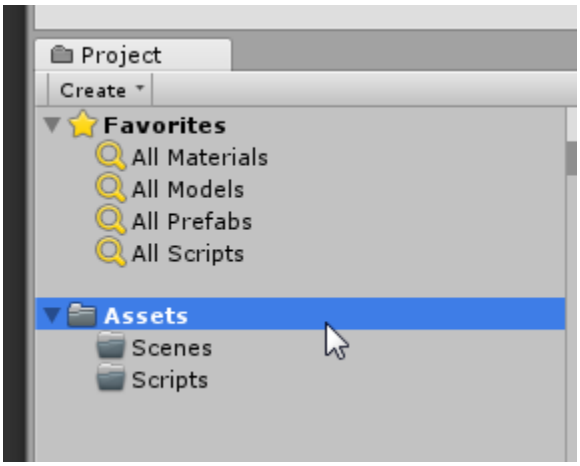
At the end of this document you will have your elbows deep in code. This is not designed for you to fully understand all the code, but instead you should be able to have a working program with a vague understanding what is going on within the code.

First Scene

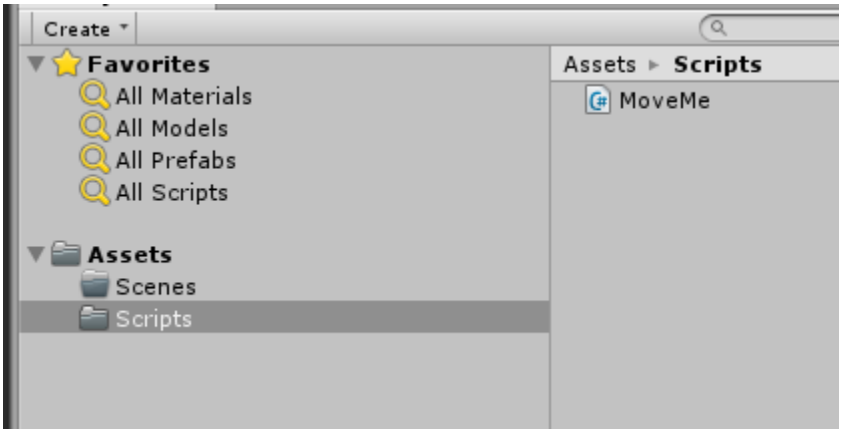
Your first coding project

Setup

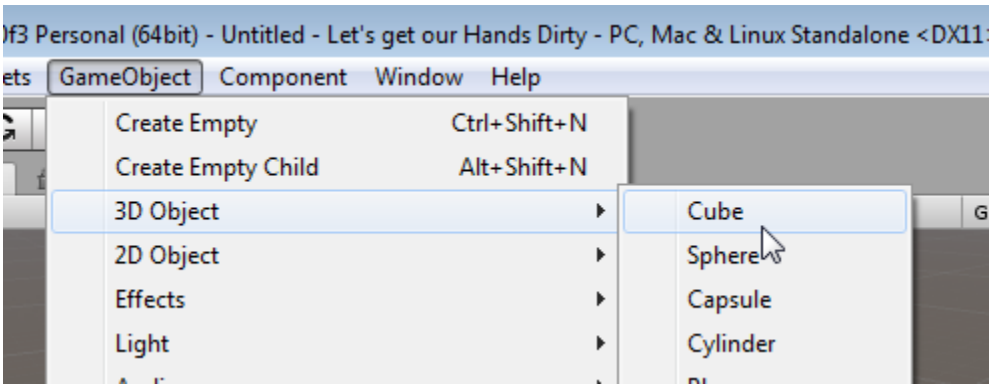
Start a new project and add **Scenes** and **Scripts** folders



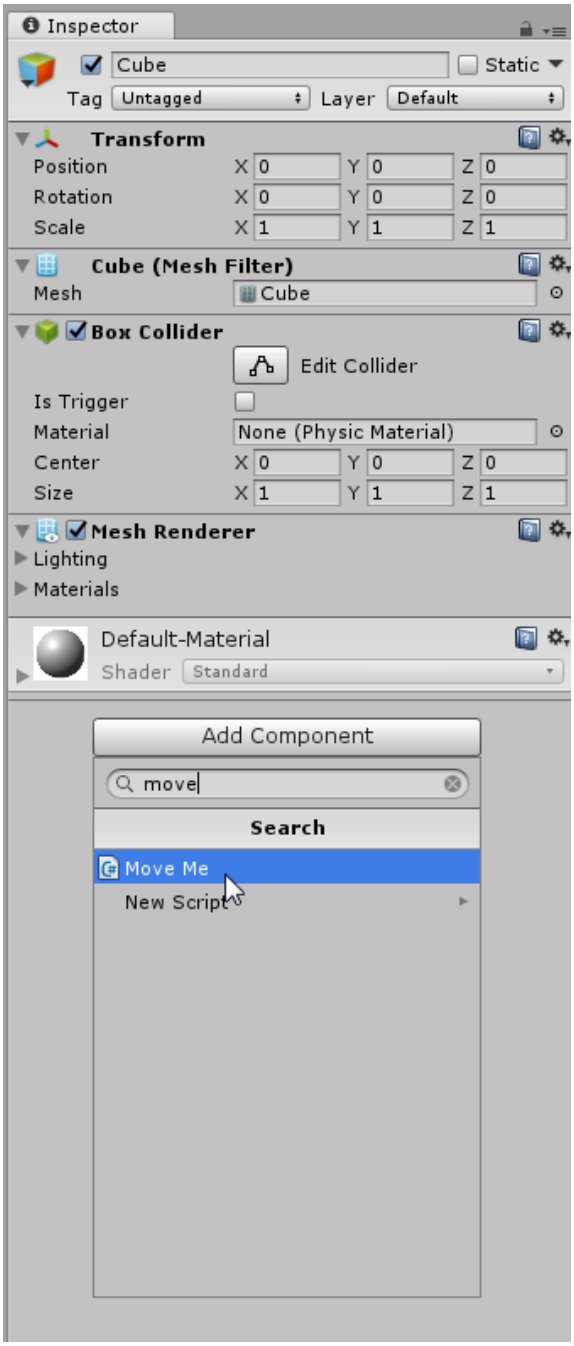
In the Scripts folder create a script called '**MoveMe**'



Create a **Cube** in your current scene



Add the **MoveMe** script to the new cube



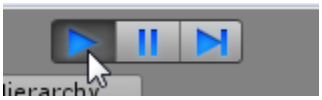
Lesson

*Note: All code highlighted in **RED** is new code or a line of code that has been changed*

Open the **MoveMe** script and put this line of code into the **Start()** function (between the curly brackets { }).

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

public class MoveMe : MonoBehaviour {

    // Use this for initialization
    void Start () {
        transform.position = transform.position + new Vector3(0, 0, 0.1f);
    }

    // Update is called once per frame
    void Update () {

    }
}
```

Run it and see what happens!



You may need to press play and stop several times to see the quick movement the cube makes.

So you should be able to see, everything that you put in the **Start()** function runs once when you press play.

Know that we can see this, let's move that line to the **Update()** function and run it again.

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

public class MoveMe : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        transform.position = transform.position + new Vector3(0, 0, 0.1f);
    }
}
```

What happens now? And what happens if you change of the 0's to another number?

You should be able to see that this code runs once every frame (about 120 times a second);

Now, we want this code only to run when we press a button.
To do this we encapsulate the line in an **If Statement**.

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

public class MoveMe : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        if (Input.GetKey(KeyCode.W))
        {
            transform.position = transform.position + new Vector3(0, 0, 0.1f);
        }

    }
}
```

Effectively what you are saying is "If the user presses the **'W'** key then run this line of code.

Run the project and test.

Let's put in another If Statement with slightly different numbers:

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

public class MoveMe : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        if (Input.GetKey(KeyCode.W))
        {
            transform.position = transform.position + new Vector3(0, 0, 0.1f);
        }

        if (Input.GetKey(KeyCode.S))
        {
            transform.position = transform.position + new Vector3(0, 0, -0.1f);
        }

    }
}
```

You can now press **W or S** to move the cube back or forward.

Can you continue this to make the cube move in 6 different directions (up, down, left, right, forwards, and backwards) by pressing 6 different keys?

Results

You should be able to see what the **Update()** and the **Start()** functions do as well as know how to get basic input from the player.

Let's give it some more control

Setup

Continued from the previous lesson.

Lesson

Here are four more commands for you to try.

See if you can work out what they do before you run them

```
public class MoveMe : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

        if (Input.GetKey(KeyCode.H))
        {
            transform.position = transform.position + (transform.forward * 0.1f);
        }

        if (Input.GetKey(KeyCode.J))
        {
            transform.Rotate(new Vector3(0, 1, 0));
        }

        if (Input.GetKey(KeyCode.K))
        {
            transform.localScale = transform.localScale + (Vector3.one * 0.1f);
        }

        if (Input.GetKey(KeyCode.L))
        {
            gameObject.SetActive(false);
        }
    }
}
```

Now we have modified two of the if statements, try these and again, see if you can guess what they do before you run the project.

Two of them seem very similar, which two and what is the actual difference between them?

```
public class MoveMe : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

        if (Input.GetKeyDown(KeyCode.H))
        {
            transform.position = transform.position + (transform.forward * 0.1f);
        }

        if (Input.GetKeyUp(KeyCode.J))
        {
            transform.Rotate(new Vector3(0, 1, 0));
        }

        if (Input.GetKey(KeyCode.K))
        {
            transform.localScale = transform.localScale + (Vector3.one * 0.1f);
        }

        if (Input.GetKey(KeyCode.L))
        {
            gameObject.SetActive(false);
        }
    }
}
```

Results

You should get the idea of being able to swap in and out basic code and be able to see how changing numbers affects what happens in the game.

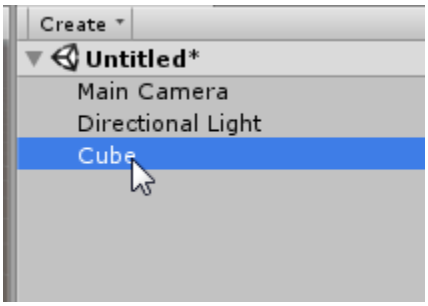
Add some physics to this puppy

Setup

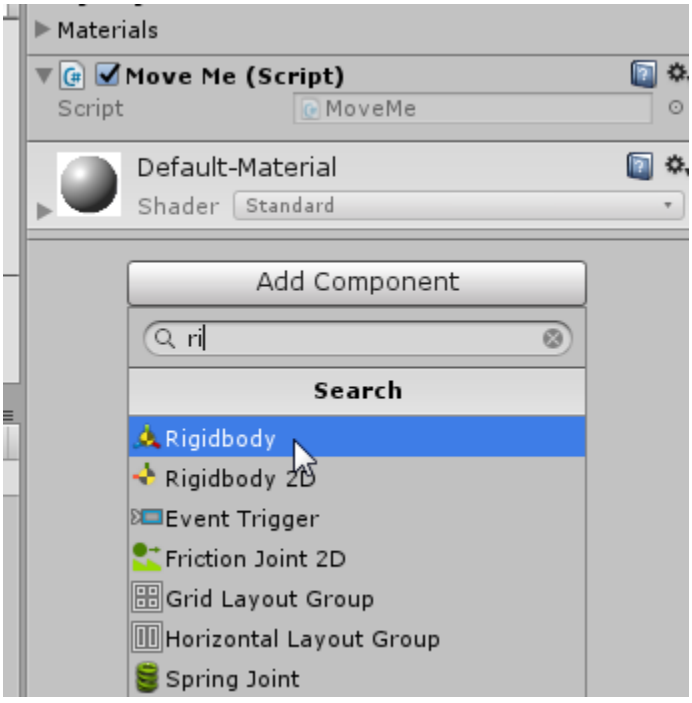
Continued from the previous lesson

Lesson

Select your cube



In the **Inspector** click on **Add Component** and select **RigidBody**



This is effectively saying "**Have the physics engine interact with this object**"

When you press play now you should see the cube drop with gravity.

Now previously we were moving the cube with direct movement. Effectively between each frame we were picking up the cube and placing it in a different spot. This is done so quickly that it looks like the cube is moving.

Now we are going to use a different style of moving the cube. Instead of directly moving it, we are going to add force to the cube and allow Unity's physics engine to work out where it should be after each frame.

These following commands only work with objects that have a **RigidBody** attached to them.

Change your script and see what they do

```
public class MoveMe : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

        if (Input.GetKeyDown(KeyCode.W))
        {
            gameObject.GetComponent<Rigidbody>().AddForce(Vector3.up * 20);
        }
    }
}
```

```
public class MoveMe : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

        if (Input.GetKeyDown(KeyCode.W))
        {
            gameObject.GetComponent<Rigidbody>().AddForce(transform.up * 20);
        }
    }
}
```

```
public class MoveMe : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

        if (Input.GetKeyDown(KeyCode.W))
        {
            gameObject.GetComponent<Rigidbody>().AddTorque(Vector3.down * 20);
        }
    }
}
```

```
public class MoveMe : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

        if (Input.GetKeyDown(KeyCode.W))
        {
            gameObject.GetComponent<Rigidbody>().useGravity = false;
        }
    }
}
```

Again, two of these commands seem very similar, which two and what is the actual difference between them?

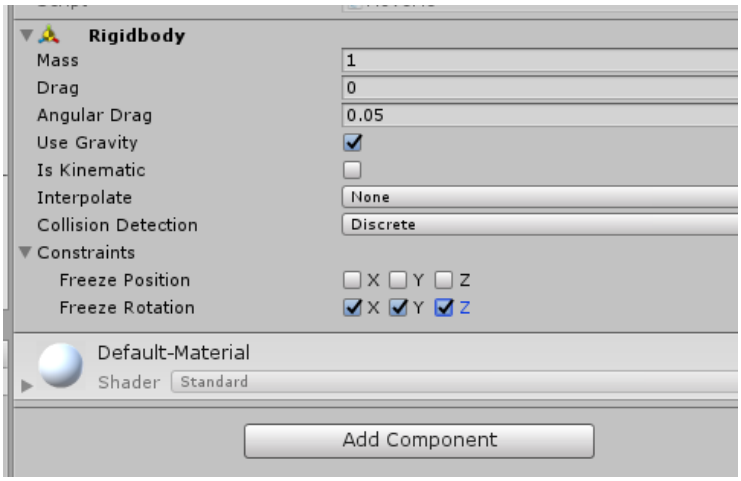
Below is a block of code to create a simple thrusting controller.

```
public class MoveMe : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

        if (Input.GetKey(KeyCode.Space))
        {
            gameObject.GetComponent<Rigidbody>().AddForce(Vector3.up * 20);
        }

        if (Input.GetKey(KeyCode.W))
        {
            gameObject.GetComponent<Rigidbody>().AddForce(Vector3.forward * 5);
        }

        if (Input.GetKey(KeyCode.S))
        {
            gameObject.GetComponent<Rigidbody>().AddForce(Vector3.forward * -5);
        }

        if (Input.GetKey(KeyCode.A))
        {
            gameObject.GetComponent<Rigidbody>().AddForce(Vector3.right * -5);
        }

        if (Input.GetKey(KeyCode.D))
        {
            gameObject.GetComponent<Rigidbody>().AddForce(Vector3.right * 5);
        }
    }
}
```

With this code you should be able to move round in a moonlander/spaceship style game.

Put a few more cubes into the scene (without RigidBodies) and fly your cube spacecraft from one cube to another.

You may notice that your cube will start to spin if you hit the ground at different angles.
You can fix this by setting constraints in the cube's **RigidBody**



Results

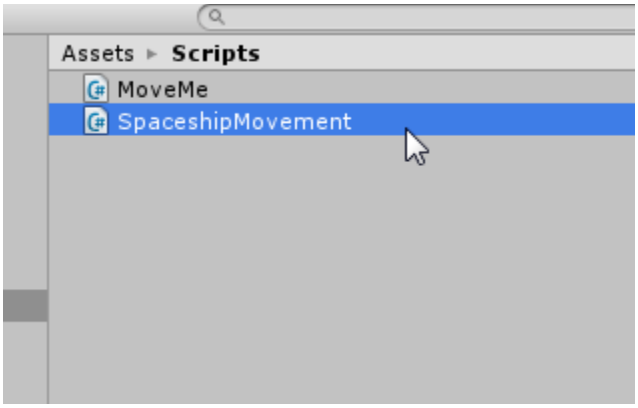
You should be able to see the difference between direct control and using the physics system to move a player.

Let's give it some more control

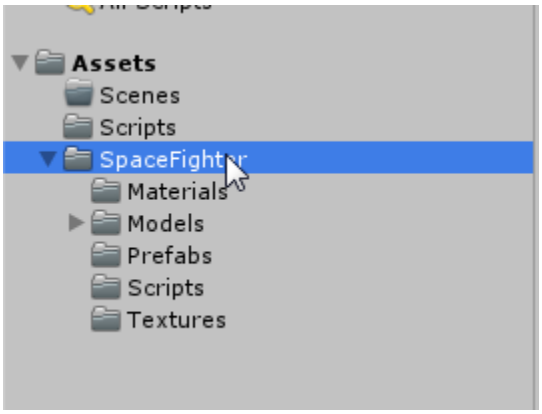
Setup

Save the scene you are working on and create a new scene.

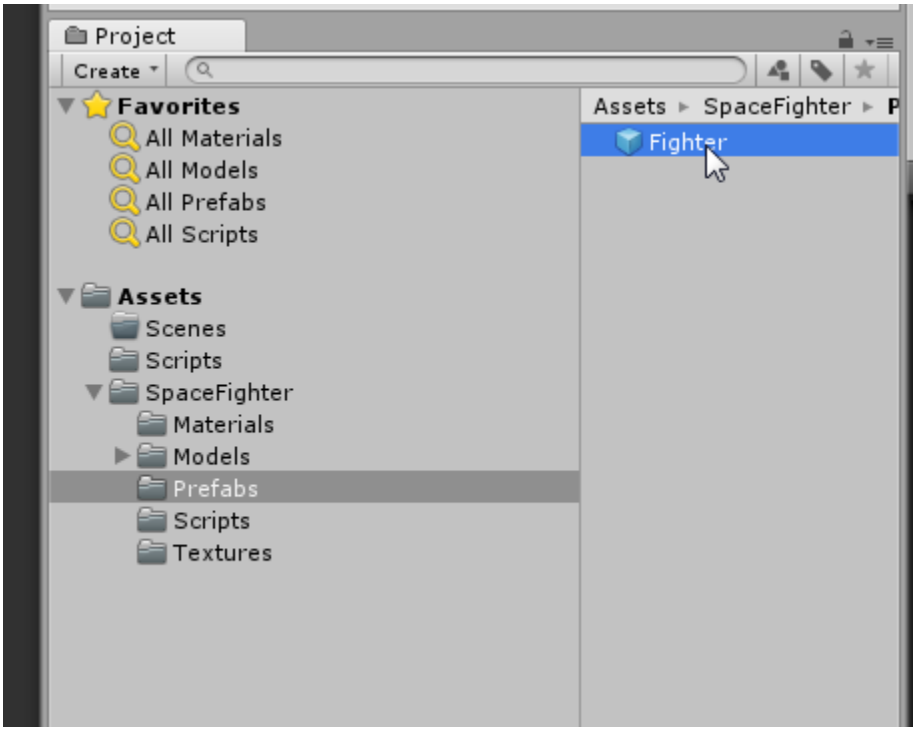
Create a new script in the **Scripts** folder called '**SpaceshipMovement**'



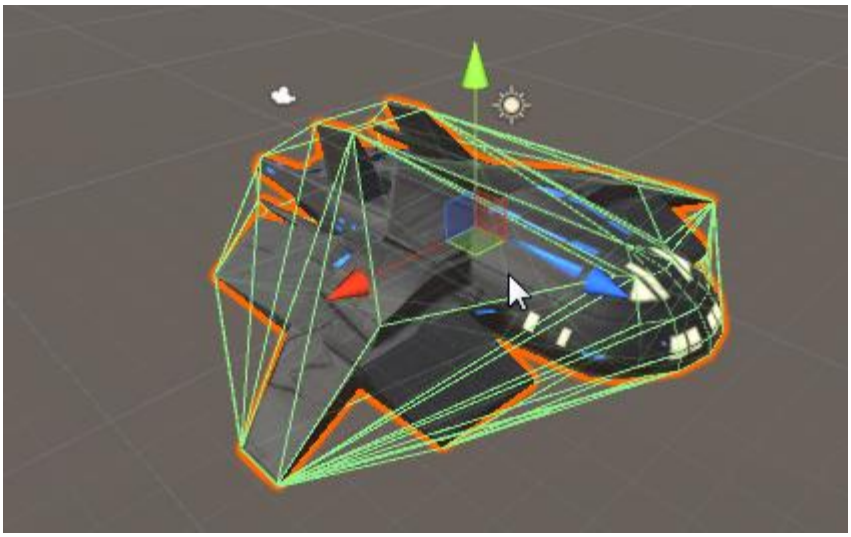
We have also provided you a Unity package called **SpaceFighter** that you can add to this project.
Once installed you should have a whole lot of new files in your project



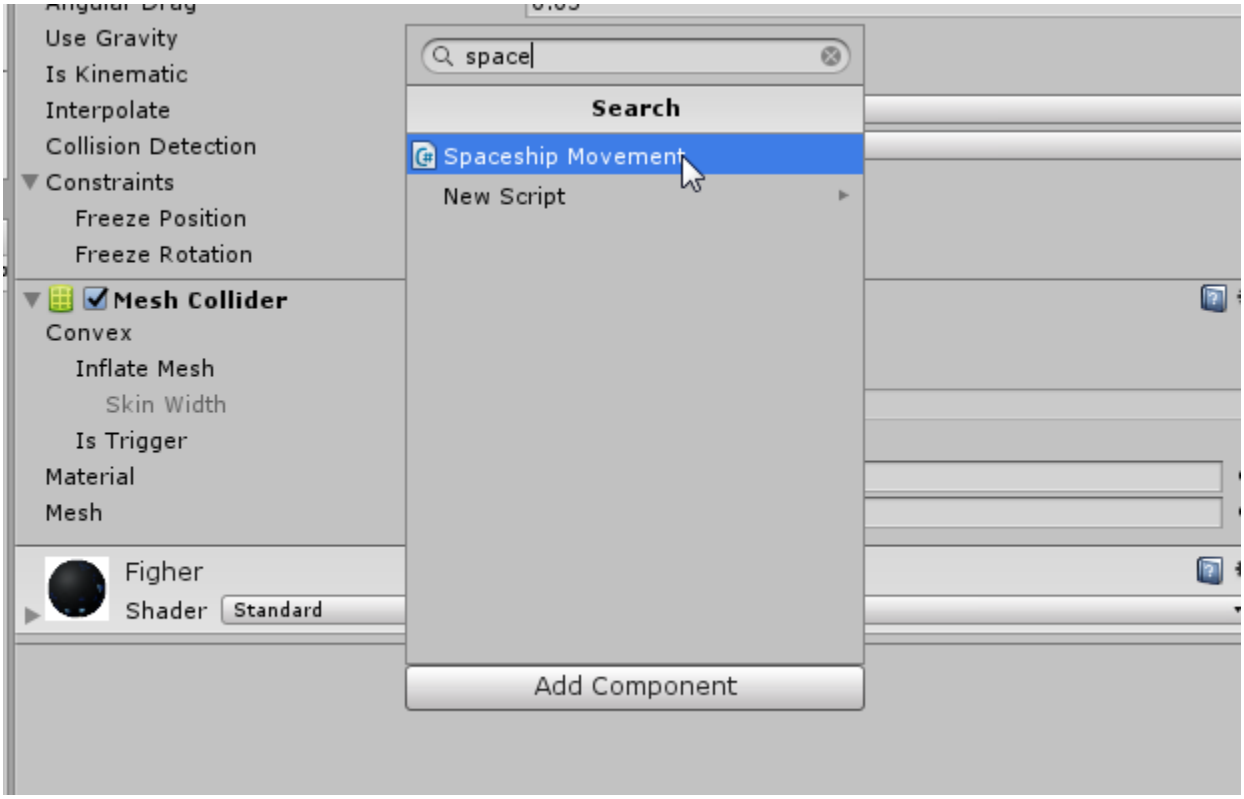
Lesson



Locate the **Fighter** prefab and drag it into the scene



Once done, add your new '**SpaceshipMovement**' script onto the spaceship.



Open up the script and add the following

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

public class SpaceshipMovement : MonoBehaviour {

    public float upwardsThrust = 40;
    public float rotationalThrust = 5f;

    // Update is called once per frame
    void Update()
    {

        if (Input.GetKey(KeyCode.Space))
        {
            gameObject.GetComponent<Rigidbody>().AddForce(transform.up * upwardsThrust);
        }

        if (Input.GetKey(KeyCode.W))
        {
            gameObject.GetComponent<Rigidbody>().AddTorque(transform.right * rotationalThrust);
        }

        if (Input.GetKey(KeyCode.S))
        {
            gameObject.GetComponent<Rigidbody>().AddTorque(transform.right * -rotationalThrust);
        }

        if (Input.GetKey(KeyCode.A))
        {
            gameObject.GetComponent<Rigidbody>().AddTorque(transform.forward * rotationalThrust);
        }

        if (Input.GetKey(KeyCode.D))
        {
            gameObject.GetComponent<Rigidbody>().AddTorque(transform.forward * -rotationalThrust);
        }

        if (Input.GetKey(KeyCode.Q))
        {
            gameObject.GetComponent<Rigidbody>().AddTorque(transform.up * -rotationalThrust);
        }

        if (Input.GetKey(KeyCode.E))
        {
            gameObject.GetComponent<Rigidbody>().AddTorque(transform.up * rotationalThrust);
        }
    }
}
```

Note that the **Start()** function has been removed, we have done this because we don't need to have any code initialised.

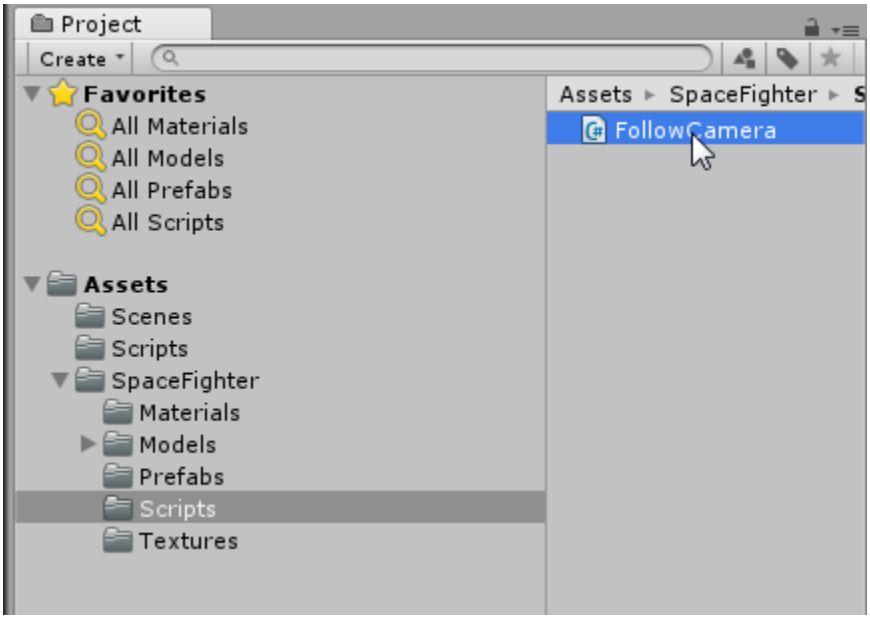
Run the project and control the spaceship…. Or try to.

Yes it is really hard.

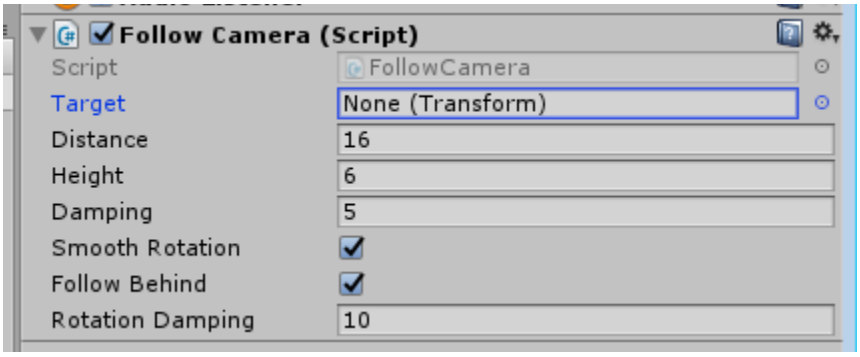
First let's make the camera follow the ship.

Select the **Main Camera**

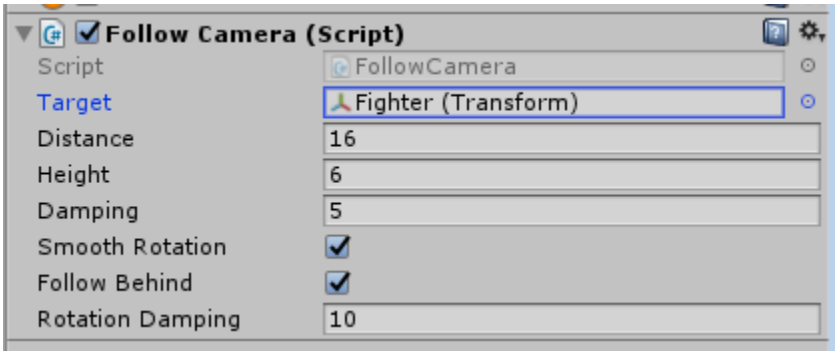
In your **SpaceFigher->Scripts** folder you should have a script called **'FollowCamera'**
Drag this onto your **Main Camera**



Now if you select the **Main Camera** you should see the script attached.



You you look at **Target** you will see there is nothing selected.
From the **Hierarchy** Drag the spaceship (called **Fighter**) to this script.



You might want to build a few objects around the level for your spaceship to fly around.

Now how do you make your spaceship easier to fly?
That is up to you!
You have learned various ways to move an object around. See if you can change the code to make the ship move the way you want it to move.

Good luck!