TUTORIAL ONE: MOVEMENT (3D)

Begin by setting up a new unity 3D project
Create a 3D object plane, by right-clicking anywhere in the asset box and navigating to and press keyboard shortcut 'R' to scale it up
Create a 3D object sphere
Optional: To make it easier to differentiate between your objects, right-click in the assets box and go to create > material. You can now name your material as appropriate and amend the colour in the inspector. Drag your material from your assets onto your object within the scene to apply your material to it.
In the hierarchy, it's good practice to rename your objects. To do this, simply left-click once on the object name in the hierarchy, right-click and select 'rename'.

To create your movement script, right-click in the assets compartment and go create > C# script. Unity will create a 'New Behaviour' script that you should first rename to 'Player'.
We can then double left-click our script to open it up in visual studios. There are a couple things here that visual studios pre-generate for us; the *void start* section, and the *void update* loop.
It's important to understand what these two sections are for – typing inside the start section tells unity to call upon that piece of code or instruction once at the start; whereas what is put inside the update section will be called upon every frame.
To move our player, we first need to set up what is known as a 'float'. A float is a variable with a fractional value. We use these when we want to represent a 'floating point', or a number. In this case, we want to set a number for how fast our player moves. And so, above your start section and beneath your class section, type:

```
public float moveSpeed;
```

Remember to begin this float with 'public', as that means that anything can access the variable at any time, and to end it with a semi-colon, which is essentially a full-stop in C#. Also pay attention to capitalisations as this is a common place of error.

Our next step is to set what we wish our moveSpeed to be. Because we want this to be checked once at the beginning by unity so we will enter our code into the start section. We also want to ensure that we type in between the set of curly brackets ({ }) that are already there for us. Taking all this into consideration, type:

```
{
        moveSpeed = 5f;
}
```

You can essentially set the '5' to any number you wish, but the 'f' is mandatory and there simply to indicate that this is a float.
Finally, we need to include our movement instructions and inputs. These will be entered in our update section, again between our curly brackets. The brackets are a way to section off our instructions.
Our first move is to type:

```
{
      transform.Translate
}
```

This is what takes our object (transform) and gives us the ability to move it (translate).
We'll follow this up with a vector, in this case, Vector3 which is each of the different points in a 3D space (x, y, z). But before we do this, it's important to know our 'inputs' and 'outputs'.

Minimise visual studios and go back to Unity. Here, you can navigate to input settings by going edit > project settings > input. This is where you can control your inputs. For example, we are going to be using "horizontal" and "vertical". Generally, the inputs are automatically left and right arrows or keys 'a' and 'd' for our horizontal axis.

With this knowledge we can continue our instruction with:

```
{
      transform.Translate(moveSpeed *
Input.GetAxis("Horizontal") * Time.deltaTime, 0f);
}
```

What this does is tells unity that we want to move our object at our set speed along the axis associated with our input key, in this case our 'a' or 'd' key, and we want our object to move by time rather than by frame, as moving by frame can be effected by hardware power. time.deltaTime ensures that regardless of hardware, the object will move at the same pace.

We can now do the same, but for our "vertical" axis, and so adding onto the end of what we have currently entered in;

```
{
      transform.Translate(moveSpeed *
Input.GetAxis("Horizontal") * Time.deltaTime, 0f, moveSpeed *
Input.GetAxis("Vertical") * time.deltaTime);
}
```

This is the final piece of code that needs to be entered in order to move our 'player'. Your final script should look as follows;

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

0 references
public class Player : MonoBehaviour
{

    public float moveSpeed;

    // Use this for initialization
    0 references
    void Start()
    {
        moveSpeed = 5f;
    }

    // Update is called once per frame
    0 references
    void Update()
    {
        transform.Translate(moveSpeed * Input.GetAxis("Horizontal") * Time.deltaTime, 0f, moveSpeed * Input.GetAxis("Vertical") * Time.deltaTime);
    }
}
```

Save and exit your script and return to unity. Before pressing play on the console, you need to ensure that your script is attached to the object you wish it to affect. You can do this in one of two ways;
the first way: select your object in the hierarchy and scroll down its inspector and left click 'add component'. You can now search for the title of your script, select it, and it will be attached to your object.
The second, and quicker way, is to drag your script from assets onto your desired object name in hierarchy.

You can now press play and using either your arrow keys or "a", "d", "s", "w" to move your player.
Note: Notice how in the inspector if you scroll down to where your script sits, you have the option to change your moveSpeed. This is because we set it as public, instead of private, earlier on in our script. Having it as public makes it easier to experiment with our speed and find what best suits our player.