**Character Movement Tutorial**

**The first step in making a functional script in Unity, is to first practice on a placeholder. To make your placeholder ‘character’ click on Assets at the top of the screen.**

**Next, go down to Create > 2D > Sprites and pick any of the options shown in the window. Here, you can name your temporary place holder, so it’s easier to identify among other sprites in later use.**

**Now you can drag and drop your sprite onto the main within the camera view indicated by the large white rectangle.**

A screenshot of a computer

Description automatically generatedA screenshot of a video

Description automatically generatedA grid with white dots

Description automatically generated

**After making a temporary character, you need to make the script that gives the character controls. To do this, go to Assets > Create, and then click C# Script.**

**After clicking this, you will be prompted to name the script. I recommend naming it something along the lines of “PlayerMove” so that it is clear what the script does.**

**Now you have a script and a sprite, drag and drop the script from the ‘Assets’ menu at the bottom of the screen into the Sprite on the scene screen.**

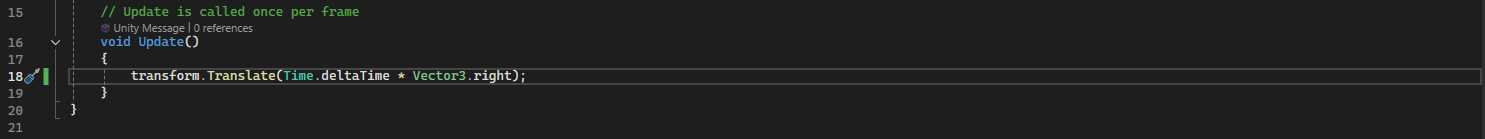
A screenshot of a computer

Description automatically generatedA white square with green symbol

Description automatically generated

**After placing the script onto the sprite, double click the script, and Unity will automatically open which ever code writing software you have set as the default.**

**To start the script, first look under the line that says, “void Update()” and type in “transform.Translate(Time.deltaTime \* Vector3.right);” between the curly brackets. This should make your character sprite move slowly to the right, or along the X axis.**



**The next step is to set a base speed for your player character. In this example, the speed default speed is considered ‘1’, and can be altered in the curly brackets bellow if needed.**

A computer screen with text and images

Description automatically generated

**The final step in allowing your character to move is to have the game check for inputs. To do this, you need to write two lines of code above your transform “transform.Translate” line.**

A screen shot of a computer code

Description automatically generated

**The phrase ‘Input’ in this context monitors for when a button is pressed. ‘GetAxis’ means that it is checking for any the directional keys. The “Horizontal” inside the brackets ensures that the game is only checking for ←,→,A, and D. The next line to add is a lot simpler. “Debug.Log(Input);” which works with the previous line, and the updated next line, in order to add the float value from the previous line to your “transform.Translate” equation.**