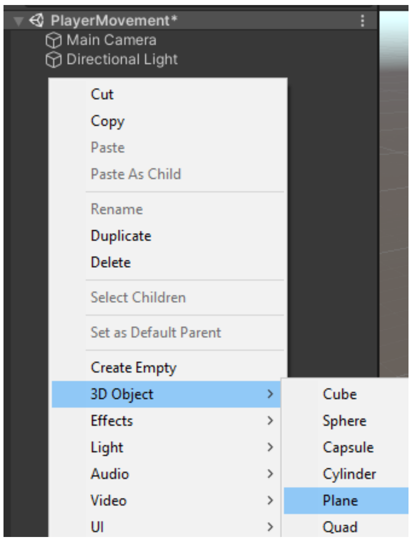
Oliver Mota

Programming Tutorials

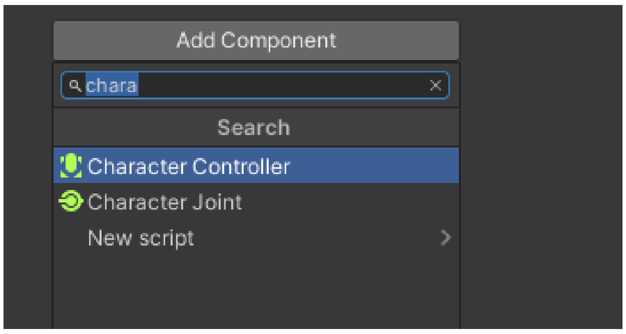
Tutorial 1: First Person Character Controller

I'm going to make a first person controller for my tutorial, I will be showing the code needed, what parts do and how to set it all up.

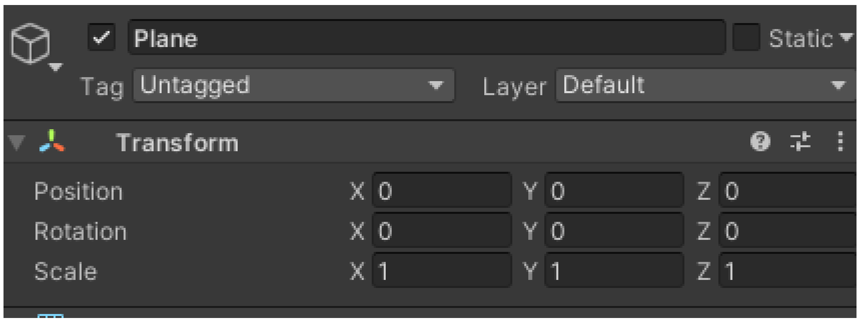
To start I have created a new scene and added a plane for the floor. I've done this by right clicking in the hierarchy -> 3D object -> Plane.



I'm then making an empty game object to hold my player controller. Again, right click in the hierarchy but this time click 'Create Empty', then name this 'FirstPersonPlayer'. With this still highlighted click 'add component' under the inspector and search for 'character controller'.



To make sure both objects are in the right place I'll send them to the middle of the scene. Select the plane, under 'transform' change the X,Y,Z values to 0. Then select the FirstPersonPlayer and do the same except change the Y value to 1 (this is so the player will be above the plane rather than inside).



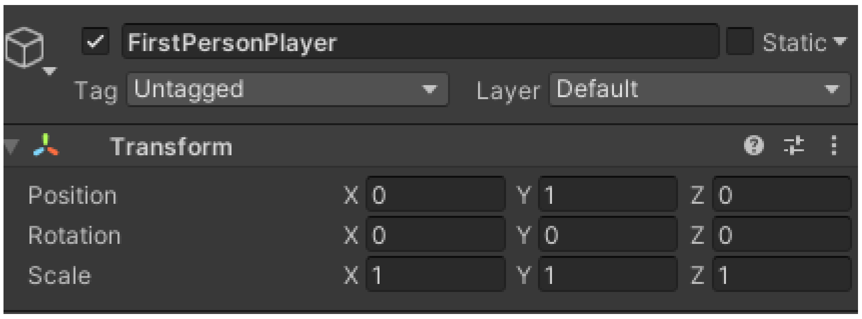
Plane

Tag Untagged Layer Default

Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Oliver Mota

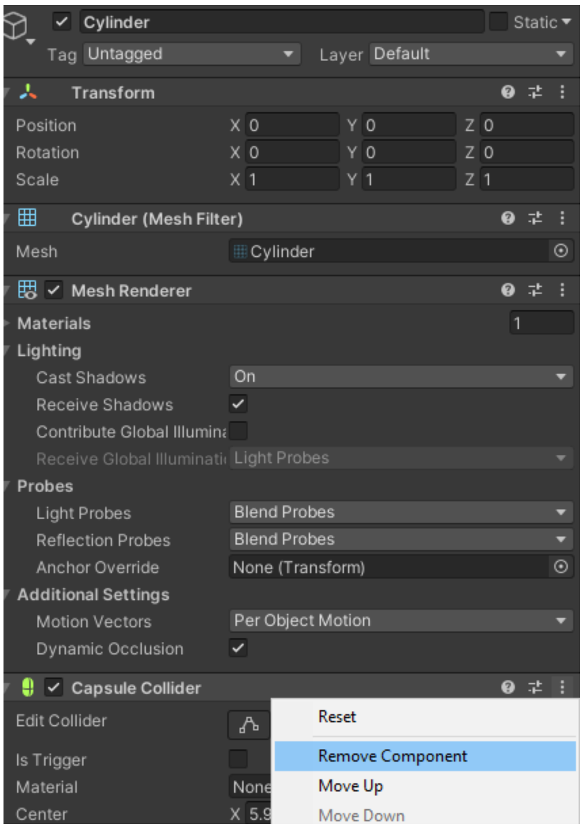


FirstPersonPlayer Static
Tag Untagged Layer Default

Transform

	X	Y	Z
Position	0	1	0
Rotation	0	0	0
Scale	1	1	1

Select the FirstPersonPlayer and add a cylinder. Right click in the hierarchy -> 3D object -> cylinder. This should be under the FirstPersonPlayer, then remove the capsule collider from the cylinder since we don't want it to interfere with the character controller.



Cylinder Static
Tag Untagged Layer Default

Transform

	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

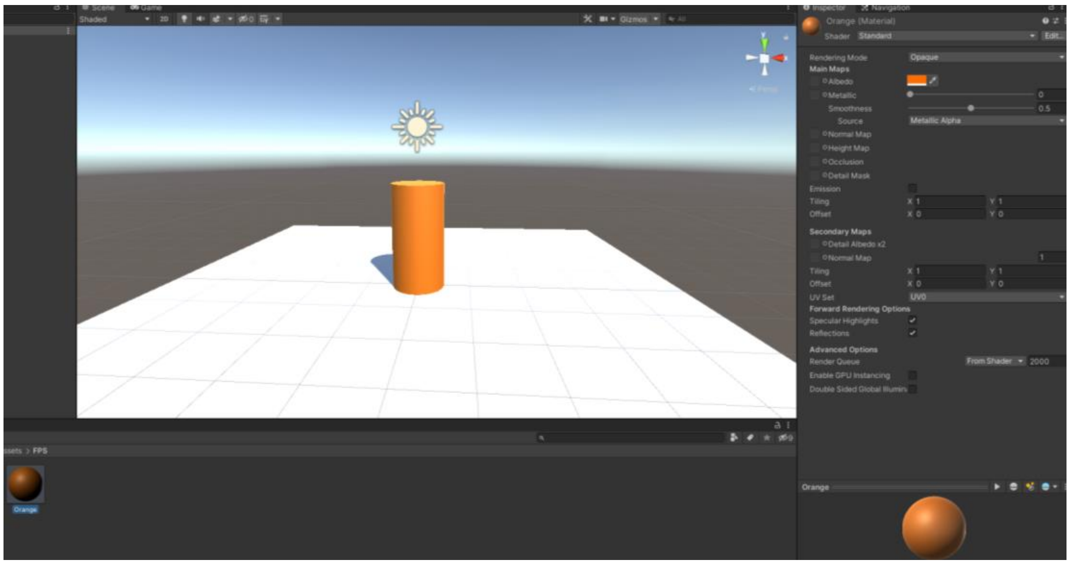
Cylinder (Mesh Filter)
Mesh Cylinder

Mesh Renderer
Materials 1
Lighting
Cast Shadows On
Receive Shadows ✓
Contribute Global Illumina
Receive Global Illuminati Light Probes
Probes
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Additional Settings
Motion Vectors Per Object Motion
Dynamic Occlusion ✓

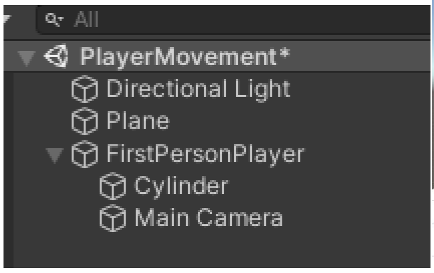
Capsule Collider
Edit Collider
Is Trigger
Material None
Center X 5.9

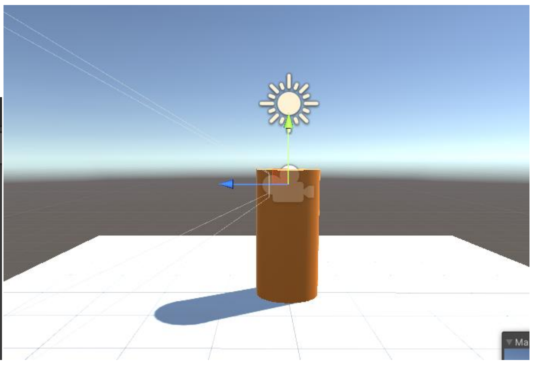
Next, we will create a colour, so the player object stands out and add the camera. For this right click in the project assets folder at the bottom of the screen and Right click -> Create -> Material. Pick a colour and drag the new material onto the cylinder object in the scene.

Oliver Mota

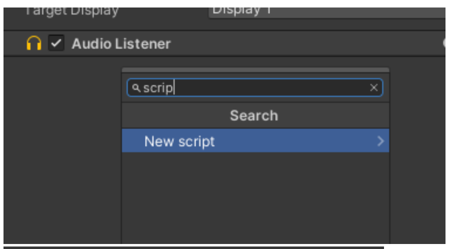


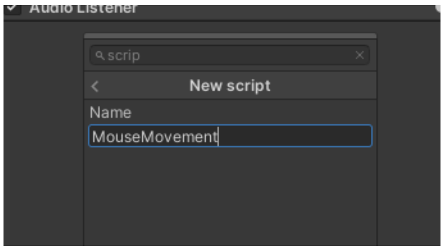
If there isn't a camera in the scene then add one from the hierarchy. Drag the camera (or main camera) onto the FirstPersonPlayer object and transform the location to 0,0,0. Drag the camera up a little bit to reach the top of the cylinder.

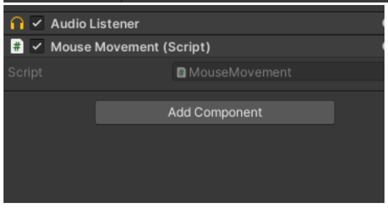




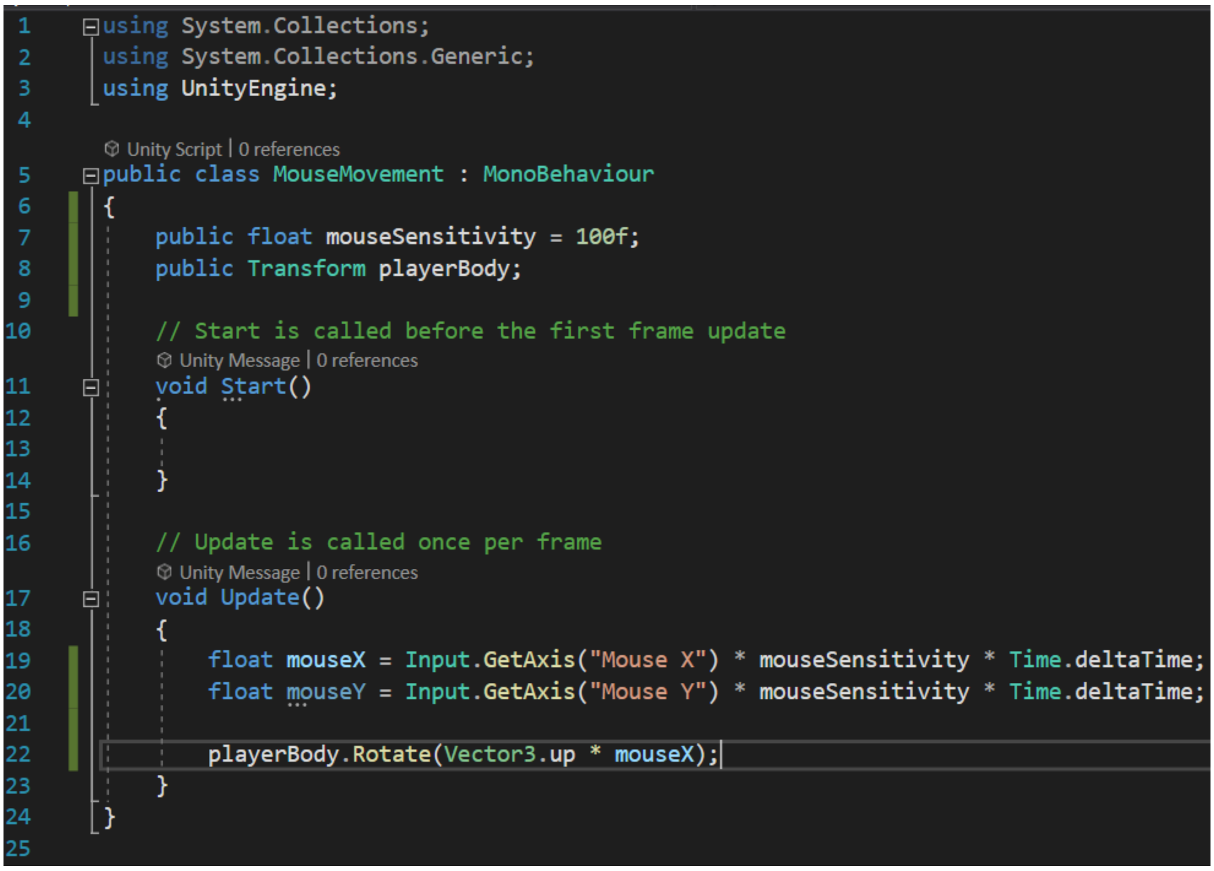
Next is the look script, to let the player look around, later we will add the actual movement so they can work together. Start by clicking on the camera and clicking 'Add Component', search script and click it. Call it 'MouseMovement'. Double click on it to open.







For the script we will be taking the mouse input and multiplying that by a speed that we set – this speed is the sensitivity. In the future this could be adapted by the player in a menu, for now it will be set at certain speed. Then we need to actually apply this to the cylinder so It can rotate. Here is the script so far:

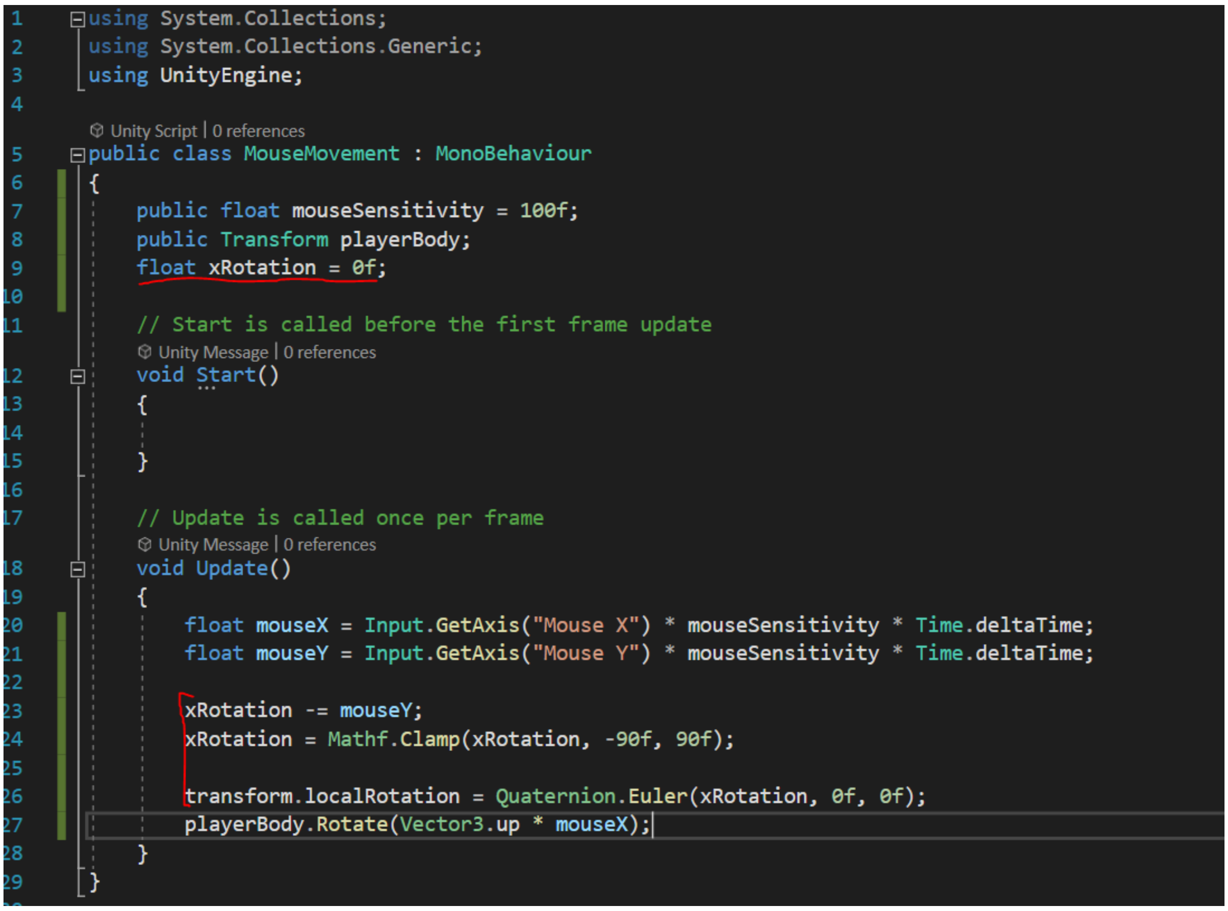


```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
    Unity Script | 0 references
5   public class MouseMovement : MonoBehaviour
6   {
7       public float mouseSensitivity = 100f;
8       public Transform playerBody;
9
10      // Start is called before the first frame update
        Unity Message | 0 references
11      void Start()
12      {
13
14      }
15
16      // Update is called once per frame
        Unity Message | 0 references
17      void Update()
18      {
19          float mouseX = Input.GetAxis("Mouse X") * mouseSensitivity * Time.deltaTime;
20          float mouseY = Input.GetAxis("Mouse Y") * mouseSensitivity * Time.deltaTime;
21
22          playerBody.Rotate(Vector3.up * mouseX);
23      }
24  }
25
```

‘public’ means we can see it in unity and change the values. On line 7&8 we are setting a speed ‘100f’ for the mouse and ‘Transform’ is like before when I moved the cylinder, only this time we will be rotating it. The White text are names which you set, these are important so whatever the name/ variation of the you choose to type, it must be the same everywhere. – for the tutorial just use the same names I have. (Capital letters are important)

To set up the X rotation we need to create a few things, I will also stop the camera from rotation too far down and back otherwise the player will be able to do a 360 through the player.

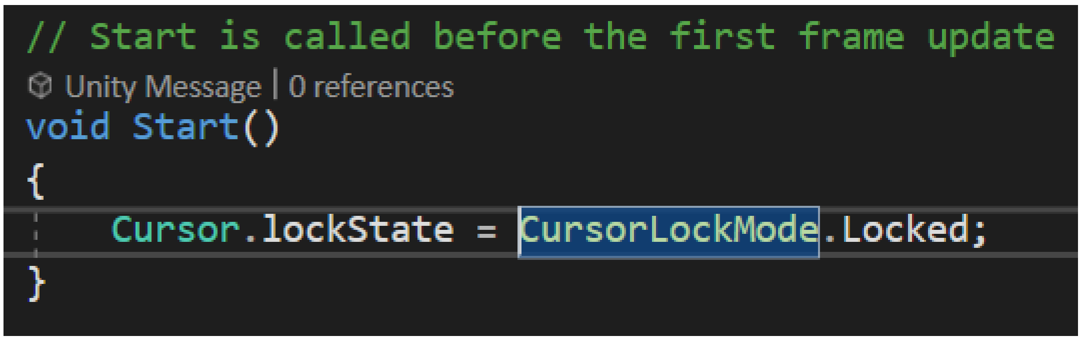
Oliver Mota



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

Unity Script | 0 references
public class MouseMovement : MonoBehaviour
{
    public float mouseSensitivity = 100f;
    public Transform playerBody;
    float xRotation = 0f;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        float mouseX = Input.GetAxis("Mouse X") * mouseSensitivity * Time.deltaTime;
        float mouseY = Input.GetAxis("Mouse Y") * mouseSensitivity * Time.deltaTime;

        xRotation -= mouseY;
        xRotation = Mathf.Clamp(xRotation, -90f, 90f);

        transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);
        playerBody.Rotate(Vector3.up * mouseX);
    }
}
```

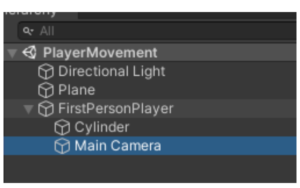
I've highlighted the added code. The 'Mathf.Clamp' is responsible for how far the player can move their head. These 2 values are the minimum and maximum angle the player can rotate the camera.

The mouse can leave the play window when testing if this works so to stop that and keep the mouse in the scene, under the start function add this:

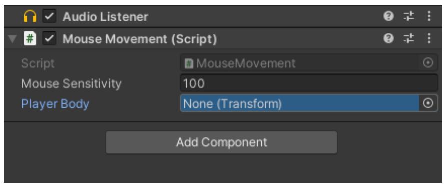


```
// Start is called before the first frame update
Unity Message | 0 references
void Start()
{
    Cursor.lockState = CursorLockMode.Locked;
}
```

Before we test if this works we need to drag the FirstPersonPlayer onto the empty slot we've created. Once that's done we test by clicking the play button in Unity and moving the mouse. – the speed for now doesn't matter as this can be changed when the movement is set up.



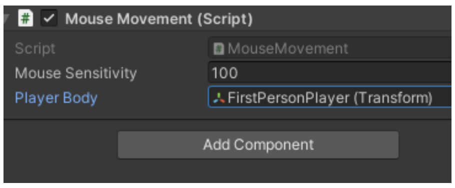
```
PlayerMovement
  Directional Light
  Plane
  FirstPersonPlayer
    Cylinder
    Main Camera
```



Audio Listener
Mouse Movement (Script)

Script	MouseMovement
Mouse Sensitivity	100
Player Body	None (Transform)

Add Component

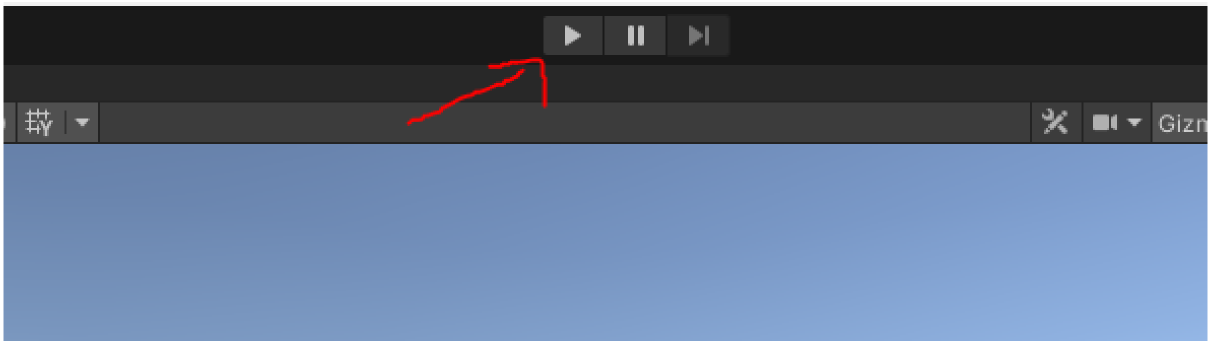


Mouse Movement (Script)

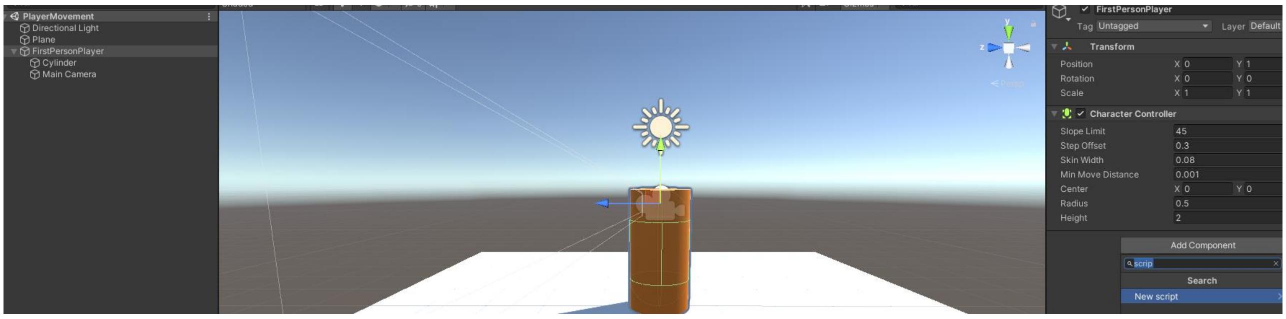
Script	MouseMovement
Mouse Sensitivity	100
Player Body	FirstPersonPlayer (Transform)

Add Component

Oliver Mota



Next I will setting up the movement so the player can actually move around. Select the FirstPersonPlayer and select ‘new script’ under the ‘add component’. Load up the script.


PlayerMovement
Directional Light
Plane
FirstPersonPlayer
Cylinder
Main Camera

FirstPersonPlayer
Tag Untagged Layer Default

Transform

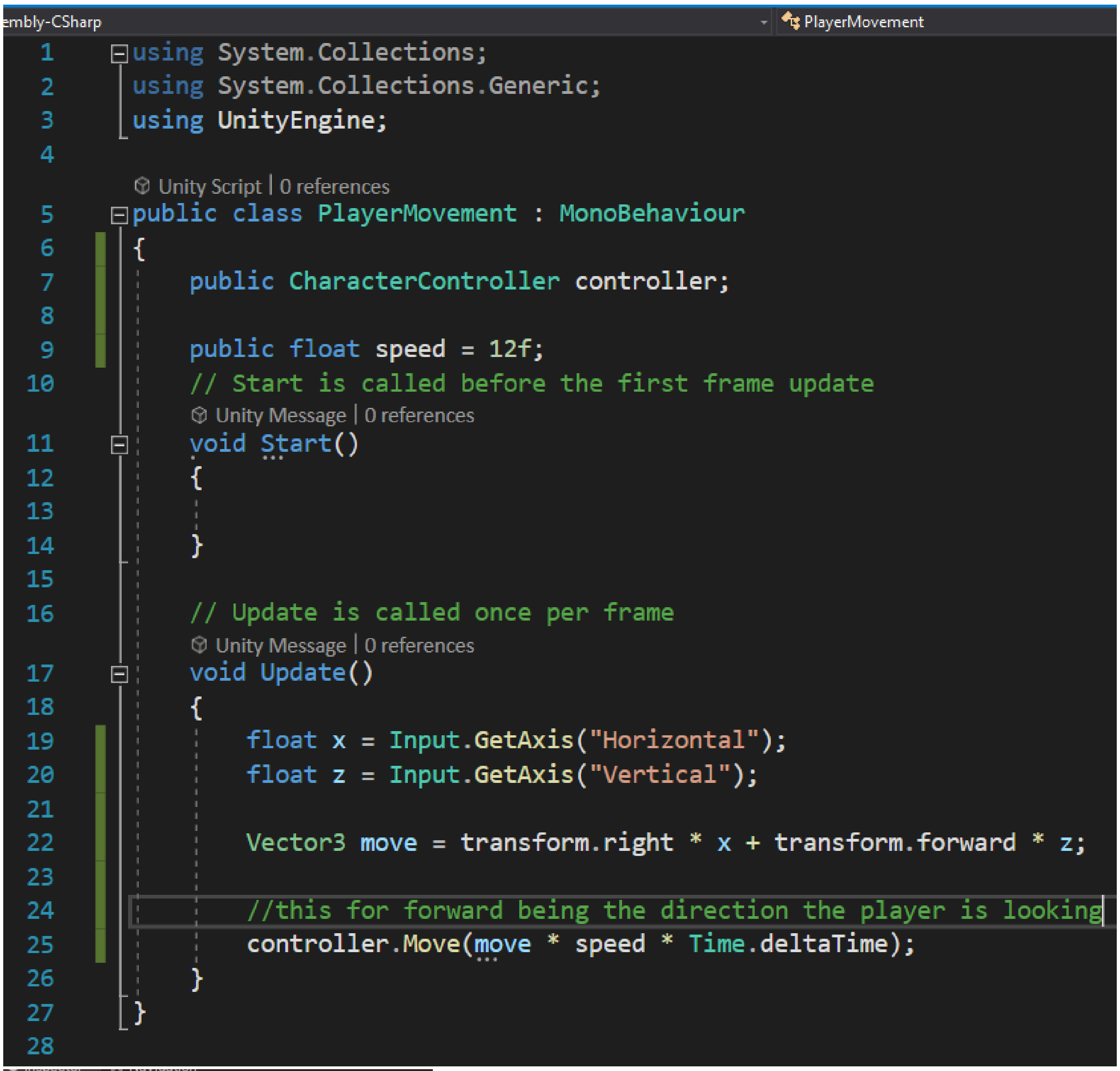
Position	X 0	Y 1
Rotation	X 0	Y 0
Scale	X 1	Y 1

Character Controller

Slope Limit	45	
Step Offset	0.3	
Skin Width	0.08	
Min Move Distance	0.001	
Center	X 0	Y 0
Radius	0.5	
Height	2	

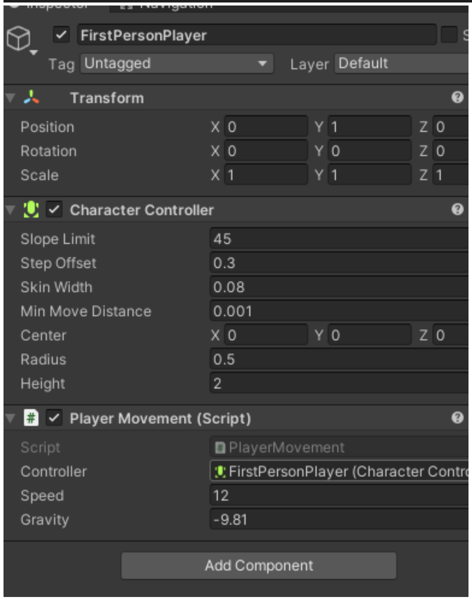
I’ve called my script ‘PlayerMovement’. Again, we take the input of the player but this time its from the keyboard and we change that into a direction using ‘transform’, this time instead of rotation we’re updating the position.

Oliver Mota



embly-CSharp PlayerMovement

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

Unity Script | 0 references
public class PlayerMovement : MonoBehaviour
{
    public CharacterController controller;

    public float speed = 12f;
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        float x = Input.GetAxis("Horizontal");
        float z = Input.GetAxis("Vertical");

        Vector3 move = transform.right * x + transform.forward * z;

        //this for forward being the direction the player is looking
        controller.Move(move * speed * Time.deltaTime);
    }
}

```



FirstPersonPlayer
Tag Untagged   Layer Default

Transform

Position	X 0	Y 1	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

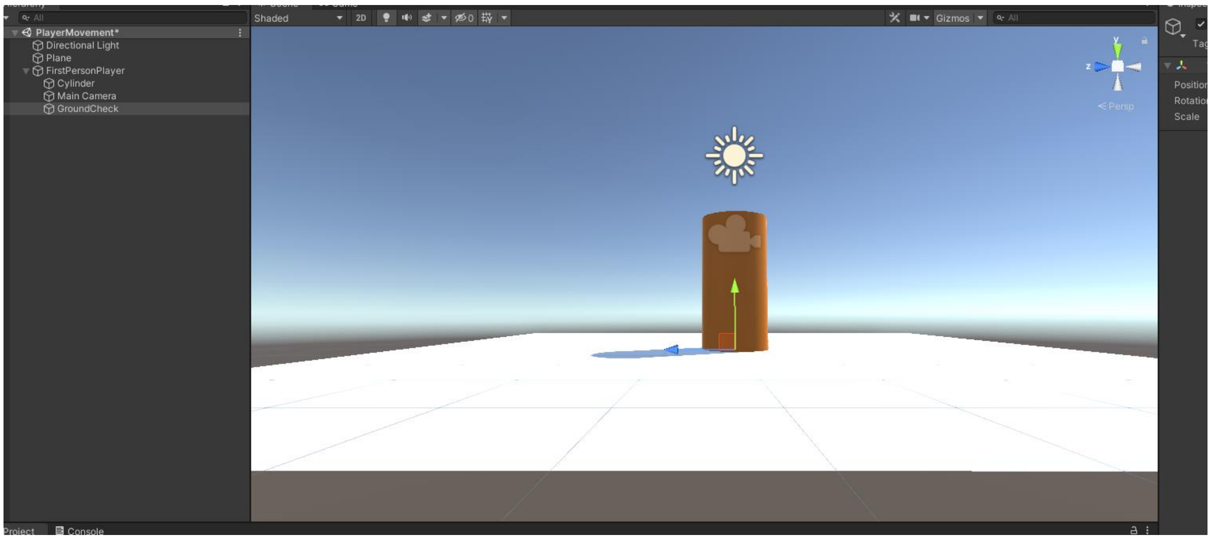
Character Controller

Slope Limit	45
Step Offset	0.3
Skin Width	0.08
Min Move Distance	0.001
Center	X 0 Y 0 Z 0
Radius	0.5
Height	2

Player Movement (Script)

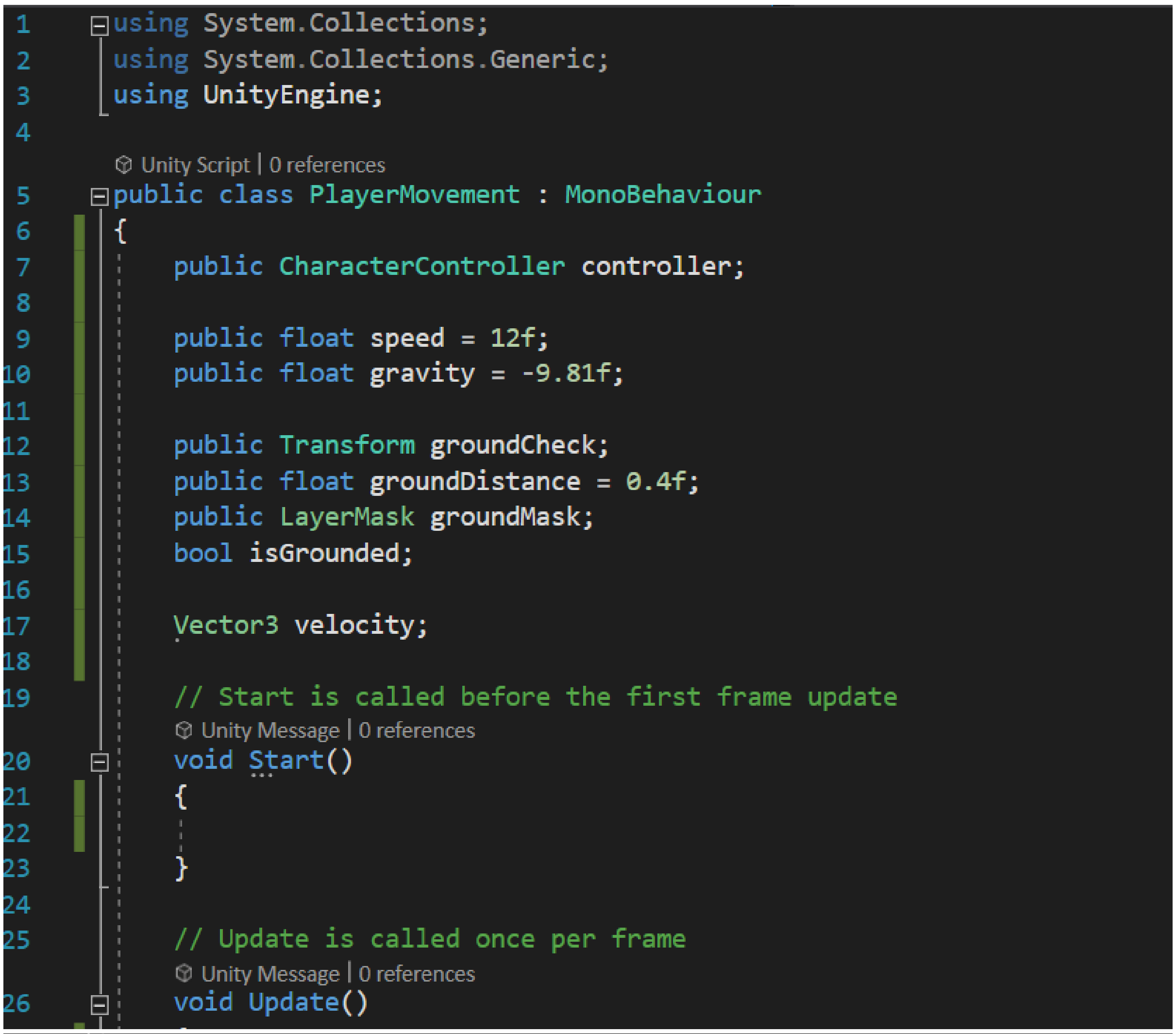
Script	PlayerMovement
Controller	FirstPersonPlayer (Character Contro
Speed	12
Gravity	-9.81

^Drag the charactercontroller into the script. Again, click play to test if this is working. Finally, I'll add gravity to the player so they can fall, if the player was to walk off a ledge they would drop until they reach ground. We will also need a ground check to check when the player is touching the ground, this stops them from constantly building momentum. For the ground check, Create an empty gameObject in the hierarchy and move it to the bottom of the cylinder.

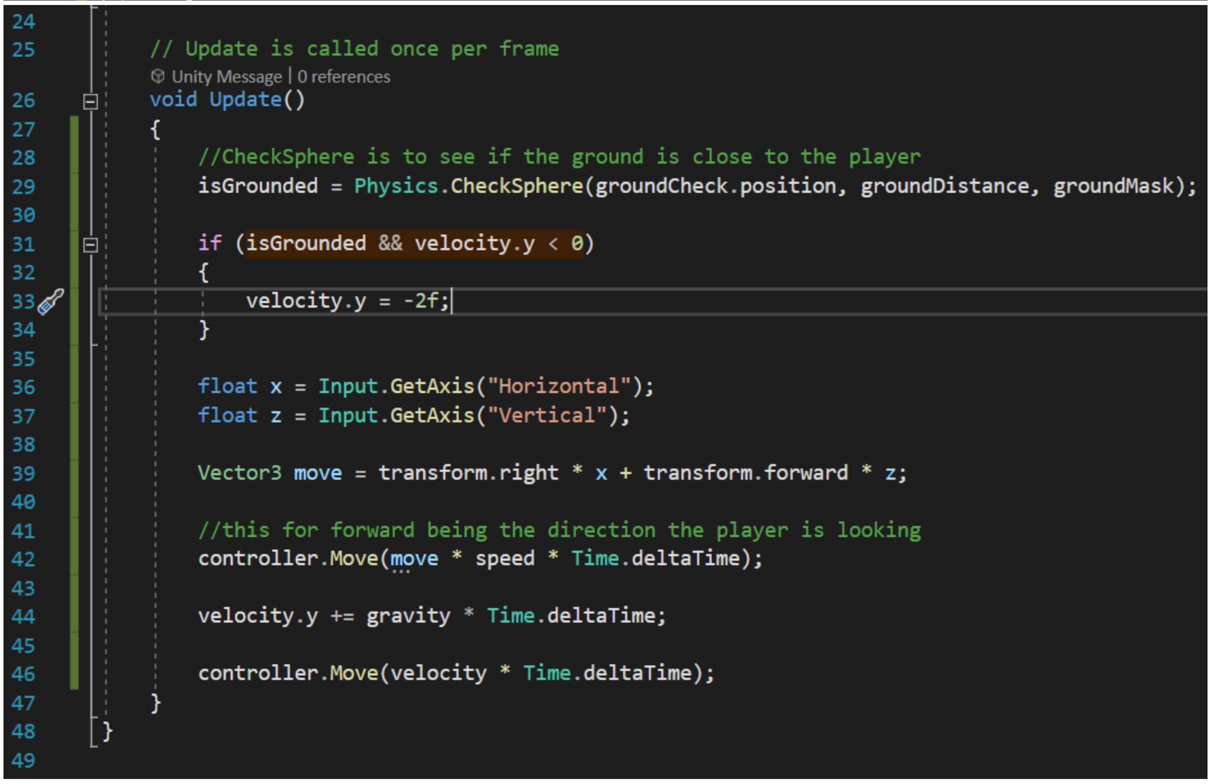


Next we go back to the playermovement script and add the code for both the ground check and gravity.

Oliver Mota



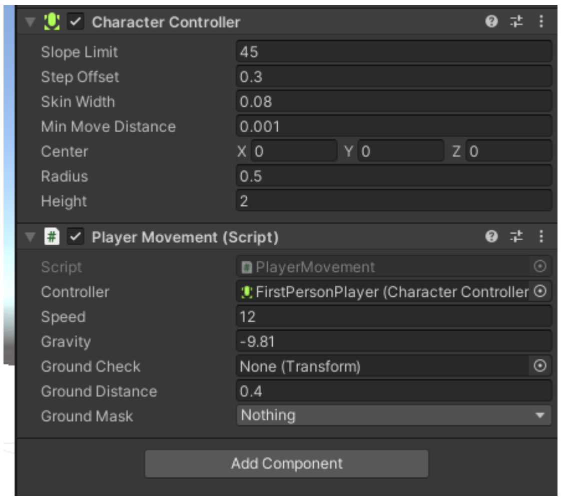
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script | 0 references
public class PlayerMovement : MonoBehaviour
{
    public CharacterController controller;

    public float speed = 12f;
    public float gravity = -9.81f;

    public Transform groundCheck;
    public float groundDistance = 0.4f;
    public LayerMask groundMask;
    bool isGrounded;

    Vector3 velocity;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
```



```
    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        //CheckSphere is to see if the ground is close to the player
        isGrounded = Physics.CheckSphere(groundCheck.position, groundDistance, groundMask);

        if (isGrounded && velocity.y < 0)
        {
            velocity.y = -2f;
        }

        float x = Input.GetAxis("Horizontal");
        float z = Input.GetAxis("Vertical");

        Vector3 move = transform.right * x + transform.forward * z;

        //this for forward being the direction the player is looking
        controller.Move(move * speed * Time.deltaTime);

        velocity.y += gravity * Time.deltaTime;

        controller.Move(velocity * Time.deltaTime);
    }
}
```

This is the updated version of the player movement script which adds gravity to the player and checks when they are on the ground. If the player is on the ground it resets the players velocity back to 0. This stops the player from constantly gaining velocity. (this would be an issue if the player walked off a ledge, they would fall really fast).

Finally theres a couple things before we can test this. The first is to drag the groundCheck from the hieracy to the play controller. Then we need to create a ‘Tag’ called Ground, then add this new ‘ ground tag’ to any objects which are ground, like the plane.

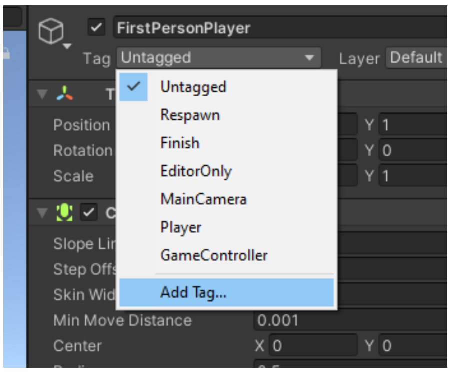


Character Controller

Slope Limit	45
Step Offset	0.3
Skin Width	0.08
Min Move Distance	0.001
Center	X 0 Y 0 Z 0
Radius	0.5
Height	2

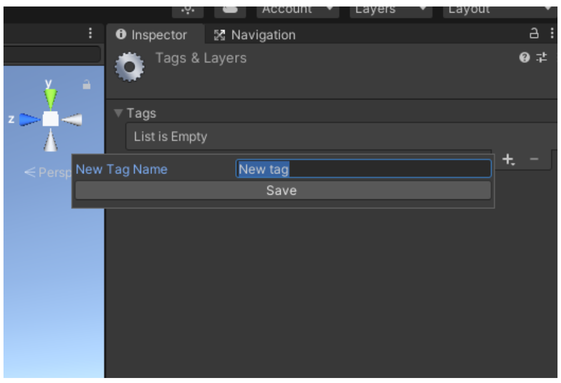
Player Movement (Script)

Script	PlayerMovement
Controller	FirstPersonPlayer (Character Controller
Speed	12
Gravity	-9.81
Ground Check	None (Transform)
Ground Distance	0.4
Ground Mask	Nothing



FirstPersonPlayer

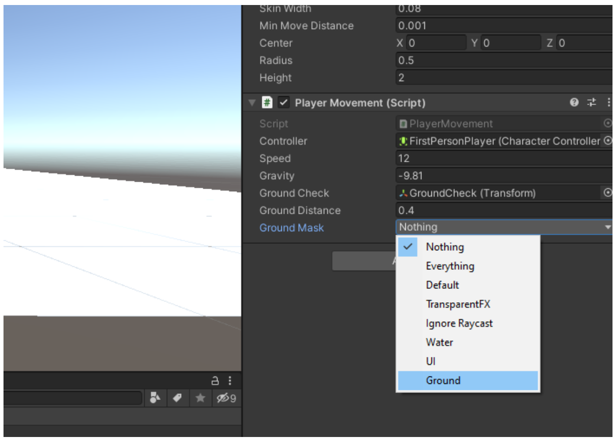
Tag Untagged Layer Default



Tags

List is Empty

New Tag Name New tag



Skin Width	0.08
Min Move Distance	0.001
Center	X 0 Y 0 Z 0
Radius	0.5
Height	2

Player Movement (Script)

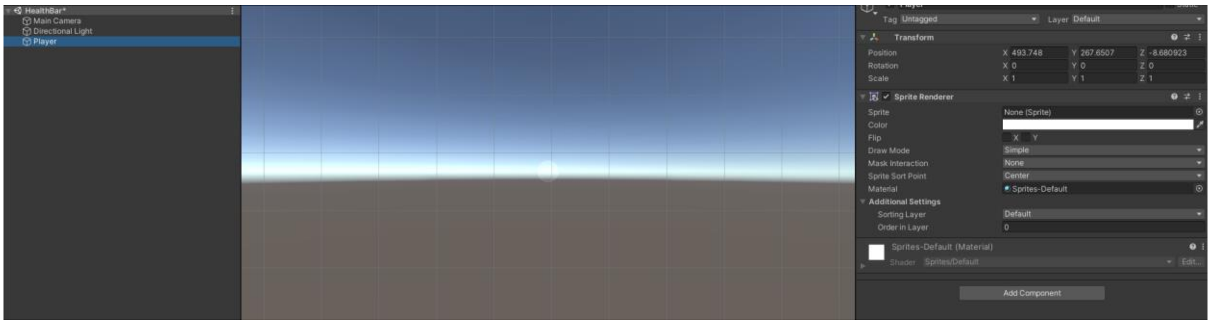
Script	PlayerMovement
Controller	FirstPersonPlayer (Character Controller
Speed	12
Gravity	-9.81
Ground Check	GroundCheck (Transform)
Ground Distance	0.4
Ground Mask	Nothing

That’s it, you should now have a moving first person character with gravity. This can be adapted to have different features like jumping or crouching.

Tutorial 2: HealthBar

This tutorial is creating a visible health bar for the player. This could be changed and adapted to represent something else rather than health but for this tutorial that's its purpose.

To start I need to create an empty gameobject in the hierarchy and I've called this player, then I've added a component under the inspector (right side of the screen) of that gameobject and added 'sprite renderer'.



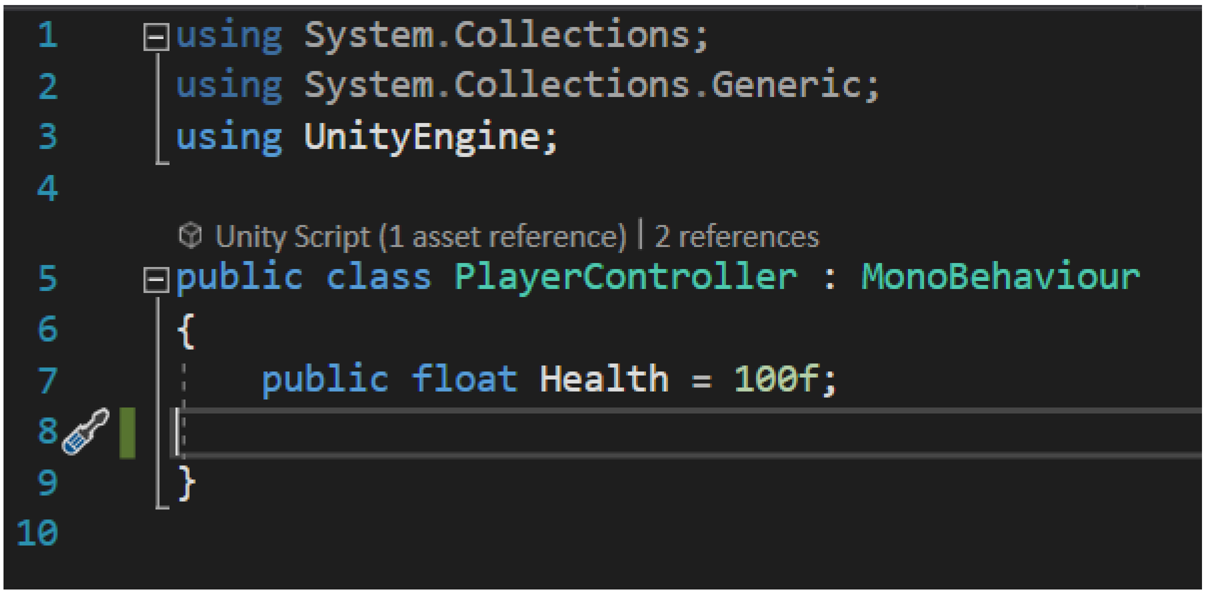
Tag Untagged Layer Default

Transform
Position X 493.748 Y 267.6507 Z -8.680923
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

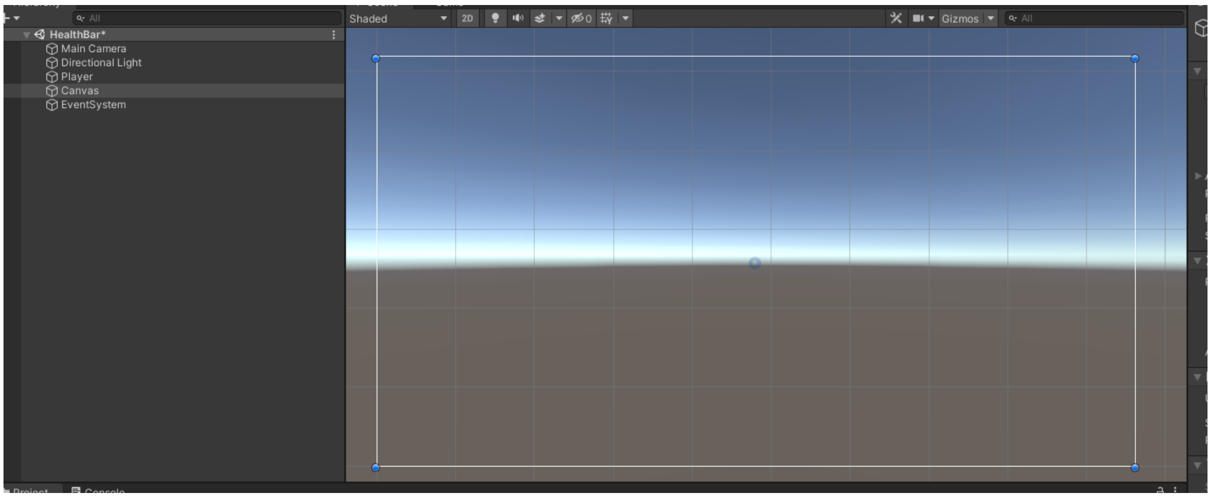
Sprite Renderer
Sprite None (Sprite)
Color
Flip X Y
Draw Mode Simple
Mask Interaction None
Sprite Sort Point Center
Material Sprites-Default
Additional Settings
Sorting Layer Default
Order in Layer 0

Sprites-Default (Material)
Shader Sprites/Default

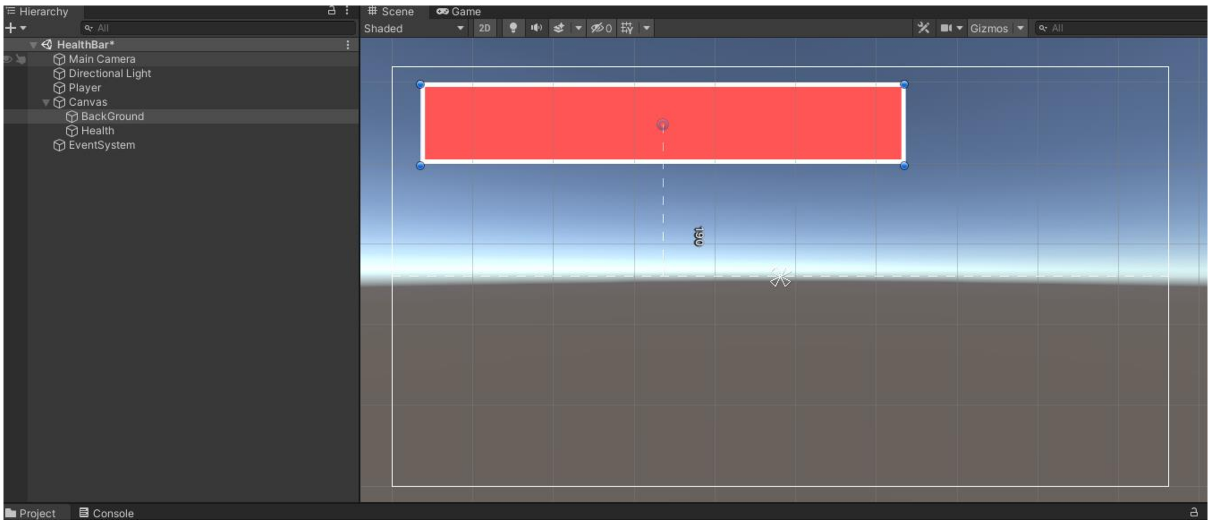
Since I'm creating UI we need to click '2d' towards the top of the screen so we can see the UI we will create. To create the UI I need a canvas, under the hierarchy, rightclick -> UI -> Canvas. Then zoom out so you can see the whole thing. I also need to create a script for the 'player' object which will be the health, its very simple with only 1 line of code.



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

Unity Script (1 asset reference) | 2 references
public class PlayerController : MonoBehaviour
{
    public float Health = 100f;

}

```

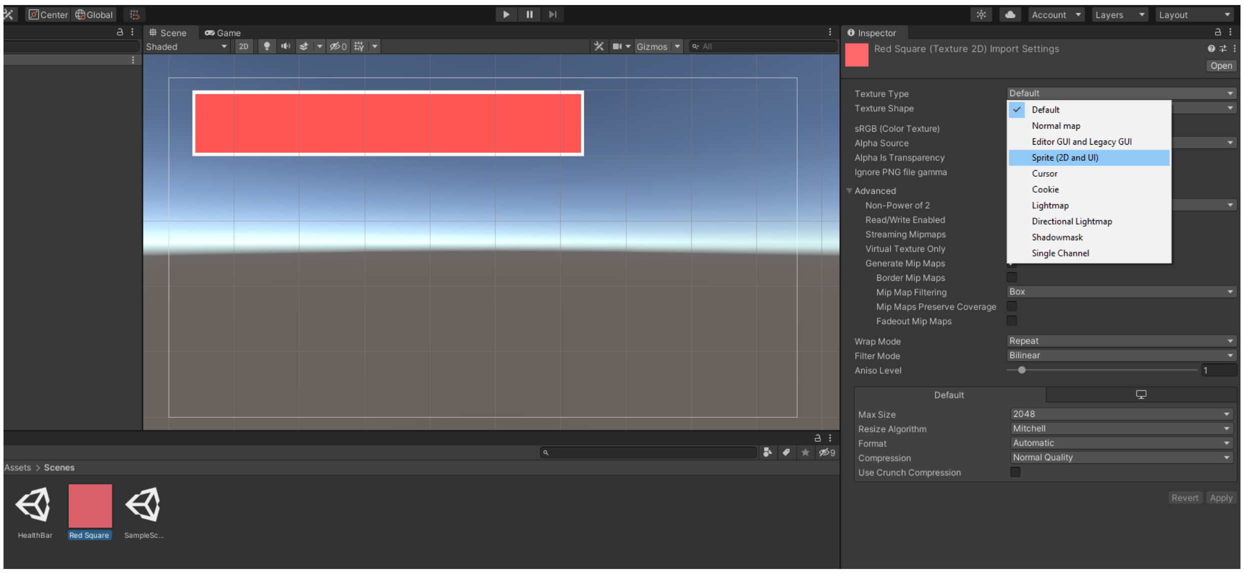


Next I need to create 2 images, one is for the background of the health bar and one is for the healthbar itself. To create these select the canvas and rightclick -> UI -> Image. Do this twice and name one, background, and the other Health. I've changed the colour of the health image to red so I can see the difference between them, I've moved them around and changed the size. This is just for now and all this can be changed later. The Background image is slightly larger and behind the health image.



Next add an image to replace the health bar, I'm using a red/pink square I've created in photoshop. Drag the image you wish to use into the project files at the bottom of the screen. Select it and under the inspection click on the drop down next to 'image type' and select 2D Sprite. When you click off it should ask you to apply, click apply.

Oliver Mota



Red Square (Texture 2D) Import Settings

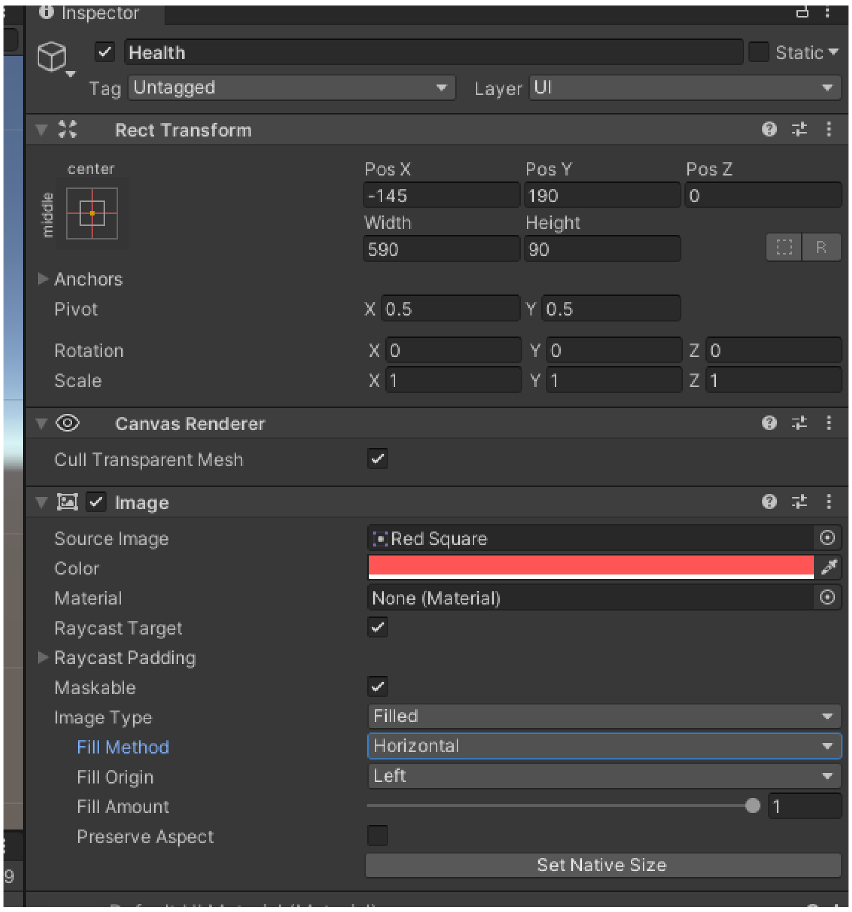
Texture Type	Default
Texture Shape	
sRGB (Color Texture)	
Alpha Source	
Alpha Is Transparency	
Ignore PNG file gamma	
Advanced	
Non-Power of 2	
Read/Write Enabled	
Streaming Mipmaps	
Virtual Texture Only	
Generate Mip Maps	
Border Mip Maps	
Mip Map Filtering	Box
Mip Maps Preserve Coverage	
Fadeout Mip Maps	
Wrap Mode	Repeat
Filter Mode	Bilinear
Aniso Level	1

Texture Type menu: ✓ Default, Normal map, Editor GUI and Legacy GUI, Sprite (2D and UI), Cursor, Cookie, Lightmap, Directional Lightmap, Shadowmask, Single Channel

Default

Max Size	2048
Resize Algorithm	Mitchell
Format	Automatic
Compression	Normal Quality
Use Crunch Compression	

Select the ‘health’ image under the hierarchy and drag the image you’ve just imported to the ‘source image’ slot. Slightly further down change ‘Image Type’ to ‘Filled’ and ‘Fill Method’ to ‘Horizontal’.



Inspector

Health ✓ — Static — Tag Untagged — Layer UI

Rect Transform

center / middle	Pos X	Pos Y	Pos Z
	-145	190	0
	Width	Height	
	590	90	
Anchors			
Pivot	X 0.5	Y 0.5	
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

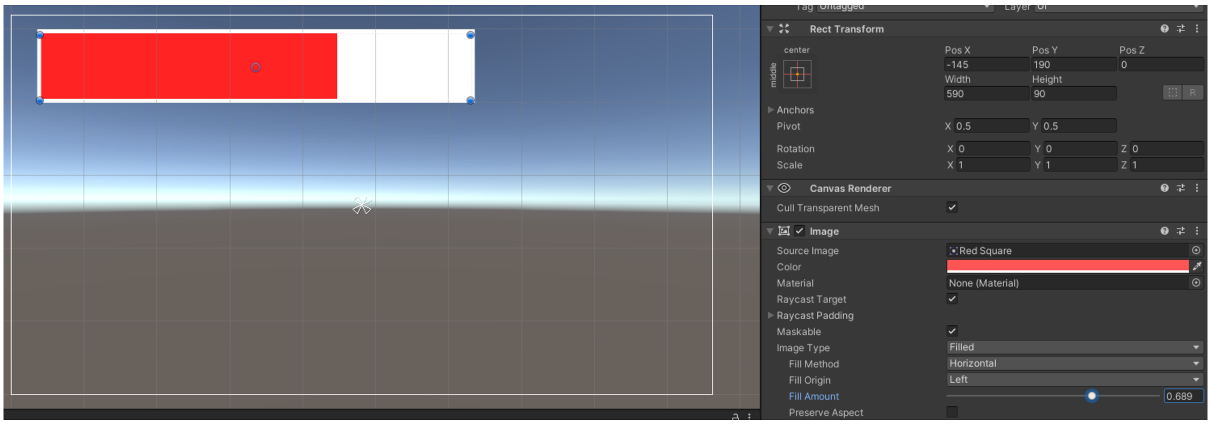
Canvas Renderer

Cull Transparent Mesh	✓

Image ✓

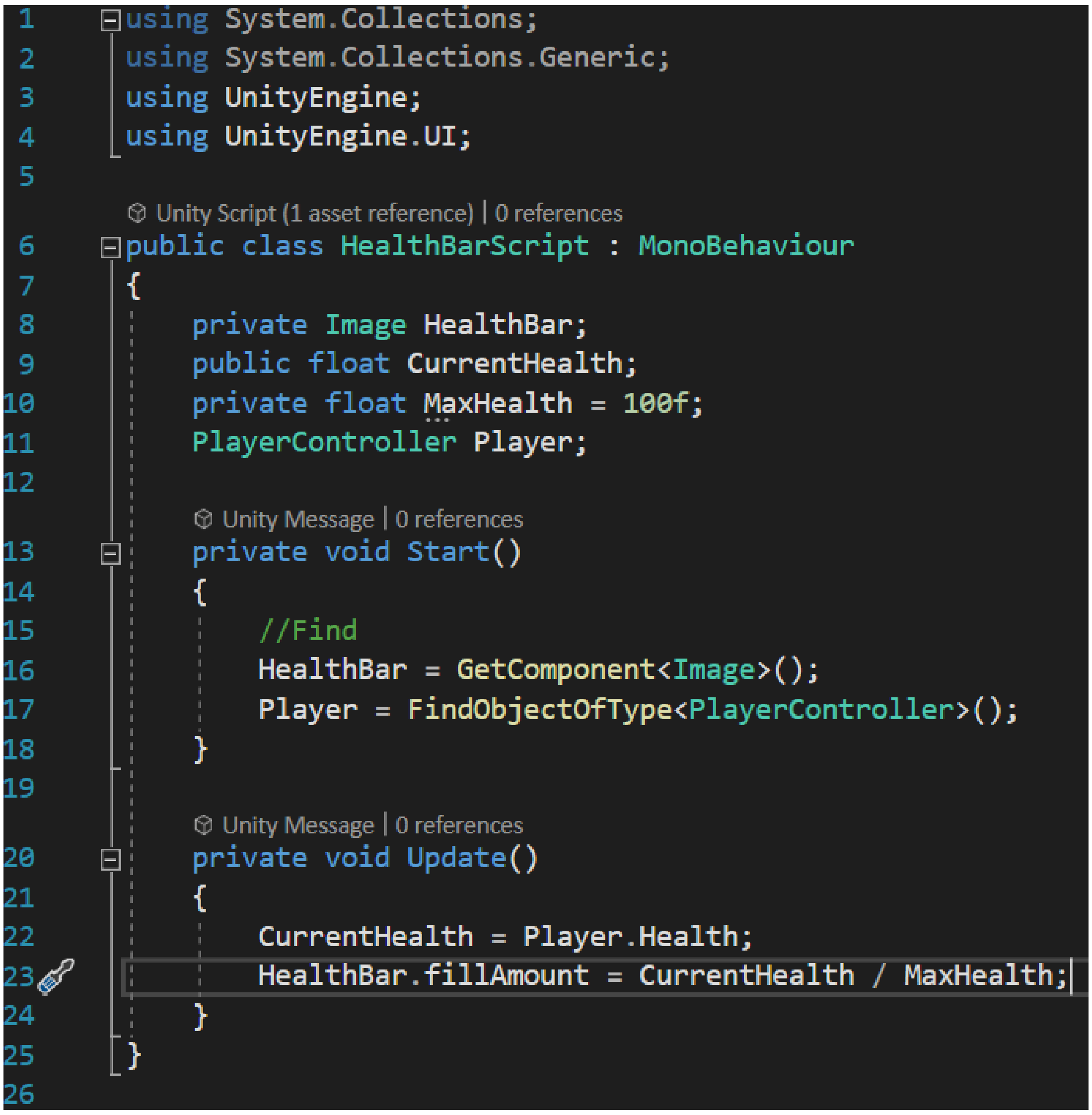
Source Image	Red Square
Color	🟥
Material	None (Material)
Raycast Target	✓
Raycast Padding	
Maskable	✓
Image Type	Filled
Fill Method	Horizontal
Fill Origin	Left
Fill Amount	1
Preserve Aspect	

You can test if this is working by dragging the ‘fill amount’ slider and seeing it update in the scene view.



```
Rect Transform
center  middle
Pos X    Pos Y    Pos Z
-145     190      0
Width    Height
590      90
Anchors
Pivot       X 0.5   Y 0.5
Rotation    X 0     Y 0     Z 0
Scale       X 1     Y 1     Z 1

Canvas Renderer
Cull Transparent Mesh   ✓

Image
Source Image       Red Square
Color
Material           None (Material)
Raycast Target     ✓
Raycast Padding
Maskable           ✓
Image Type         Filled
  Fill Method      Horizontal
  Fill Origin      Left
  Fill Amount      0.689
  Preserve Aspect
```

With the health bar still selected create a new script through the ‘Get Component’ and call it ‘HealthBarScript’. We need to add ‘using UnityEngine.UI’ at the top of the script since we are using UI elements. This script is about calculting the % of health the player has and applying that to a fill amount in the slider. So if the player has 100% health the slider will be at 100%.



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

Unity Script (1 asset reference) | 0 references
public class HealthBarScript : MonoBehaviour
{
    private Image HealthBar;
    public float CurrentHealth;
    private float MaxHealth = 100f;
    PlayerController Player;

    Unity Message | 0 references
    private void Start()
    {
        //Find
        HealthBar = GetComponent<Image>();
        Player = FindObjectOfType<PlayerController>();
    }

    Unity Message | 0 references
    private void Update()
    {
        CurrentHealth = Player.Health;
        HealthBar.fillAmount = CurrentHealth / MaxHealth;
    }
}
```

Now if we click run and play the game, the bar should work when we slide the amount. This has to be done through the ‘player’ script we created as this is the players health, the other script is just used to calculate and update the healthbar.

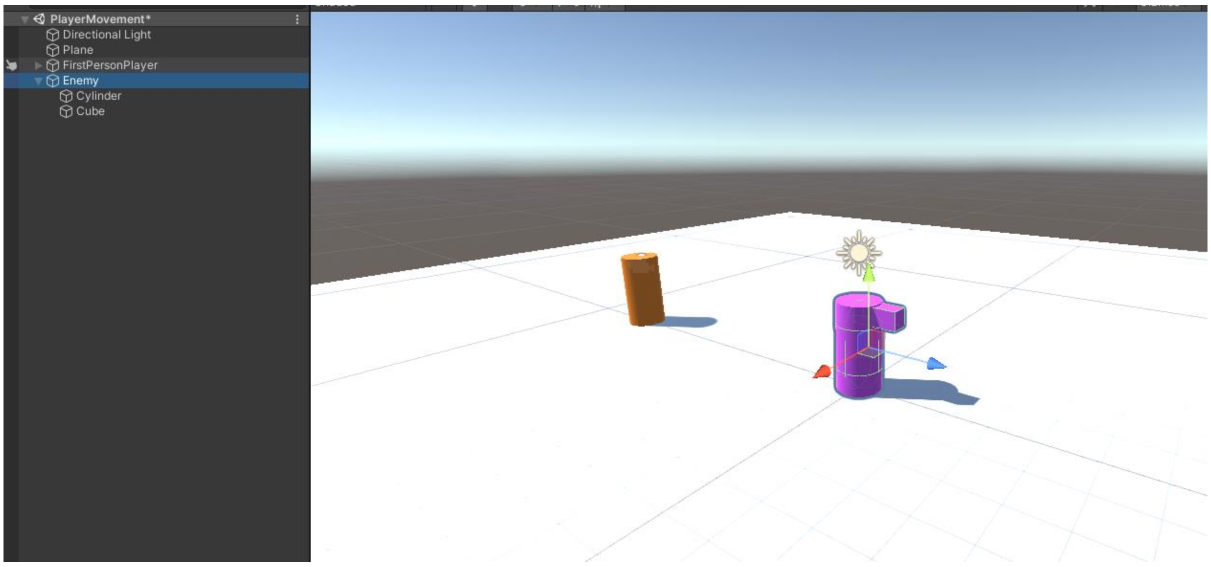
This script can be updated and used with others, so maybe when the player gets hit it removes a number from the players health which would then update in the health bar.

Tutorial 3: Follow AI

For this tutorial I am going to make a simple follow AI. I'm going to use a first person controller for this project, you can download one or use the one created in the previous tutorial as I am.

First I need a plane for the floor and the first person controller (it should be if you don't have first person controller, just a block instead and when we go to test, move it in the scene view). If your using the first person controller make sure you delete the main camera in the new scene.

I'm going to create an empty gameObject in the hierarchy and call it enemy, I'm then going to make a cylinder and a box, the cylinder for the enemy body and the box so we can see the direction they are pointing. I'm also going to create a new material and put that on the enemy, the material is just a simple colour change so they aren't the same as the player or ground.

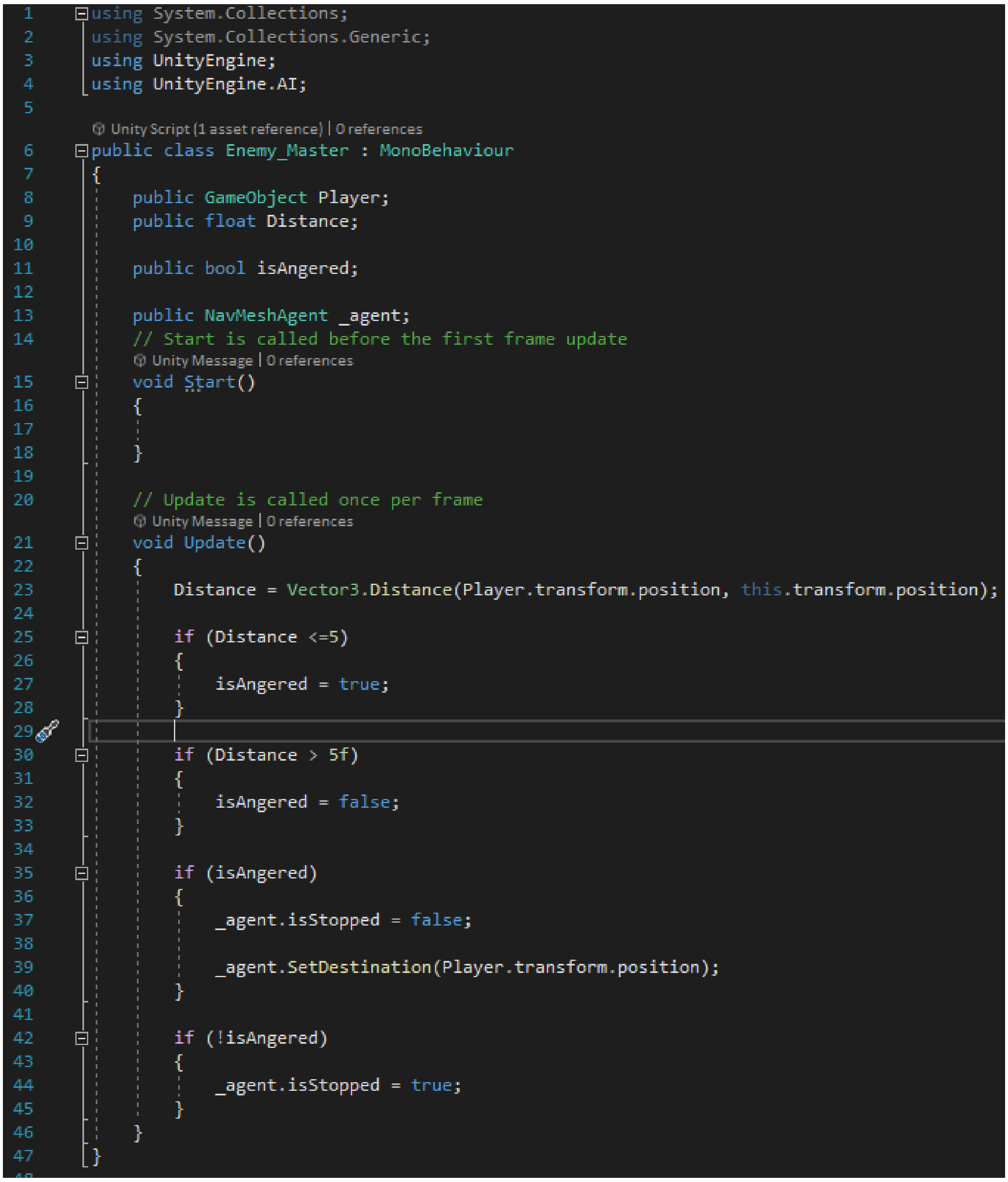


Now we need to add a script to the 'enemy' gameobject we've just created. For this select the Enemy and click 'add component' then search script, call this 'Enemy_Master'.

This script will calculate the distance between the player and the enemy. The enemy will only follow the player if the player is close enough. If the player leaves the distance the enemy will follow then it will stop following.

This will update the enemy's location and make them move towards the players location.

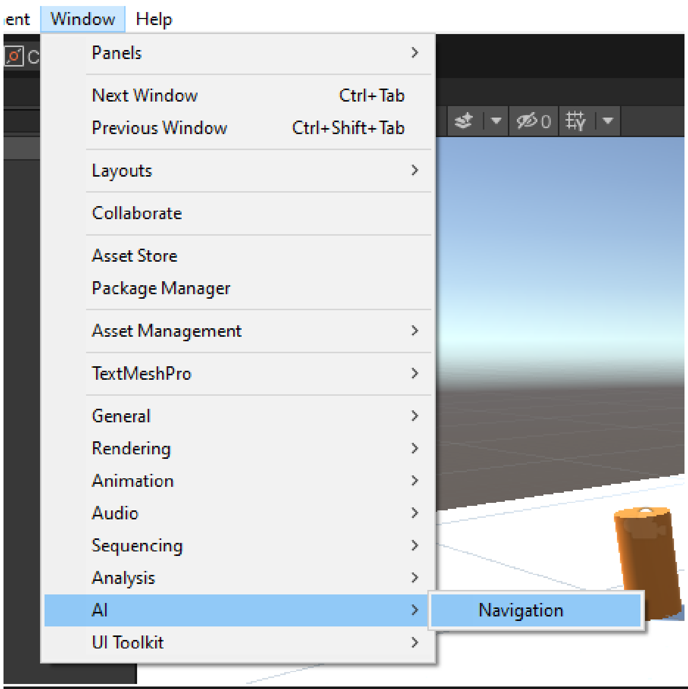
Oliver Mota

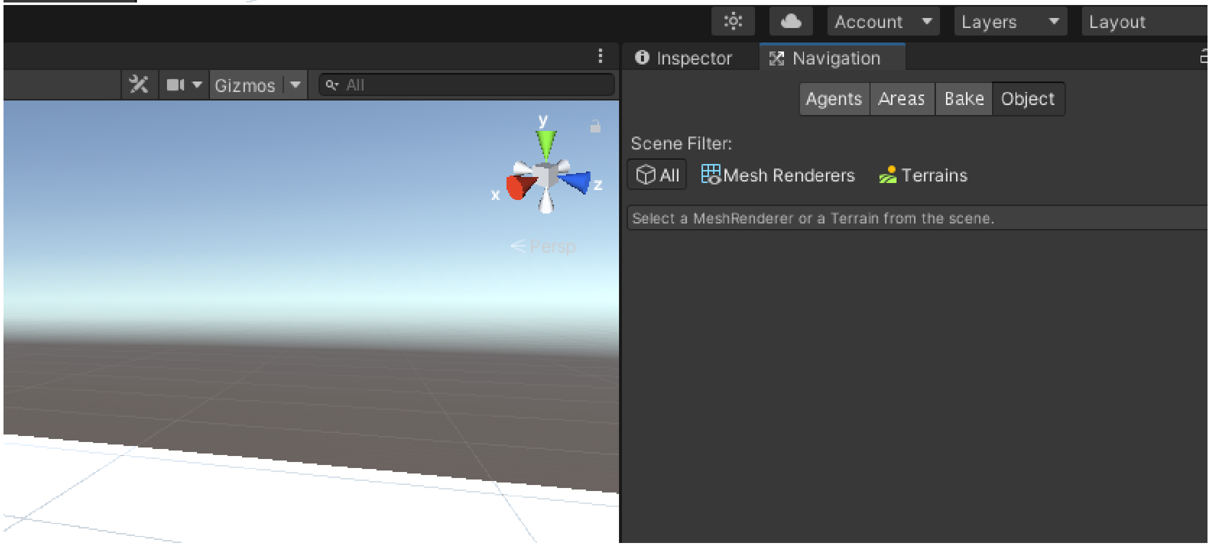


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;

Unity Script (1 asset reference) | 0 references
public class Enemy_Master : MonoBehaviour
{
    public GameObject Player;
    public float Distance;

    public bool isAngered;

    public NavMeshAgent _agent;
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        Distance = Vector3.Distance(Player.transform.position, this.transform.position);

        if (Distance <=5)
        {
            isAngered = true;
        }

        if (Distance > 5f)
        {
            isAngered = false;
        }

        if (isAngered)
        {
            _agent.isStopped = false;

            _agent.SetDestination(Player.transform.position);
        }

        if (!isAngered)
        {
            _agent.isStopped = true;
        }
    }
}
```

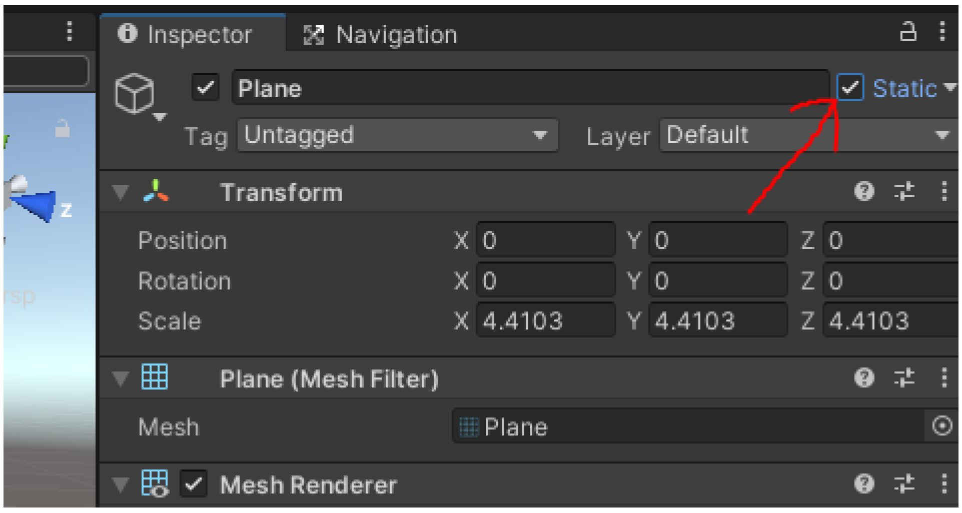
Once this is written out we need to give the enemy AI an idea of the layout they can walk on. For this go to Window (top of the screen) -> AI -> Navigation. This will open a new window which can be placed to right of the screen with the inspector.

Oliver Mota





For the ground that the enemy will walk on we need to set it to ‘static’ so select the plane and at the top right of the inspector I need to tick the static box.



Inspector | Navigation

Plane ✓ Static

Tag Untagged | Layer Default

Transform

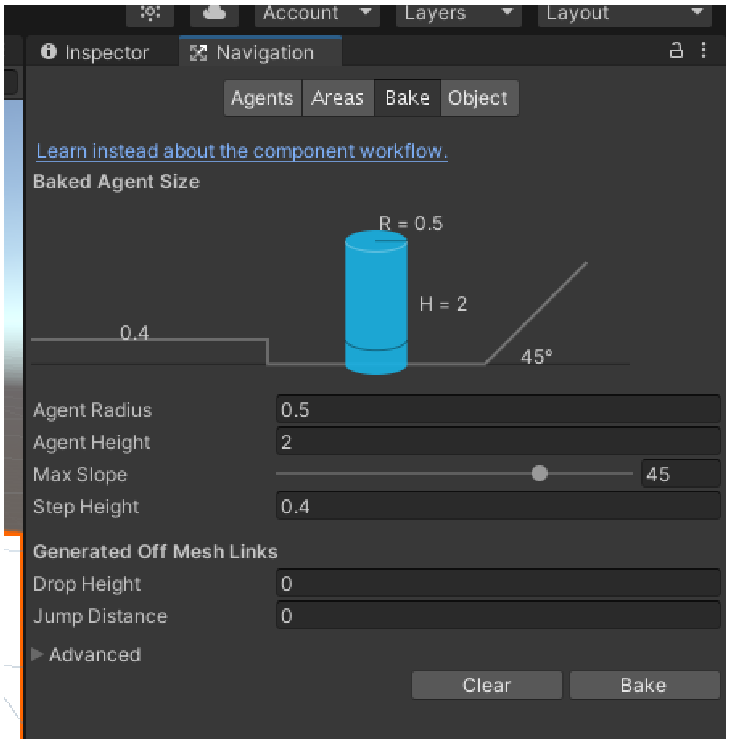
Position	X	0	Y	0	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	4.4103	Y	4.4103	Z	4.4103

Plane (Mesh Filter)

Mesh Plane

Mesh Renderer

Next is baking, select the navigation window to the right and click on the 'bake' tab. There will be a button at the bottom of that tab called 'bake'. Click this and a clear layer should be applied over the floor.

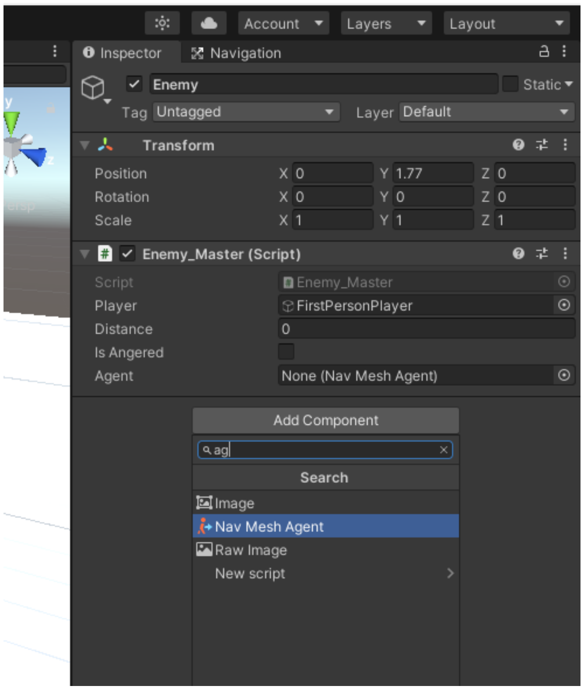


Agent Radius	0.5
Agent Height	2
Max Slope	45
Step Height	0.4

Generated Off Mesh Links

Drop Height	0
Jump Distance	0

Finally is setting up the enemy, if we select the enemy we can see that there are empty slots we've created which need to be filled in. For the player drag in your firstpersoncontroller or a box you have created to represent the player. For the 'Agent' we need to create a NavMesh agent and drag that in. Click on Add Component and search for 'Nav Mesh Agent'. Select that and drag it in.



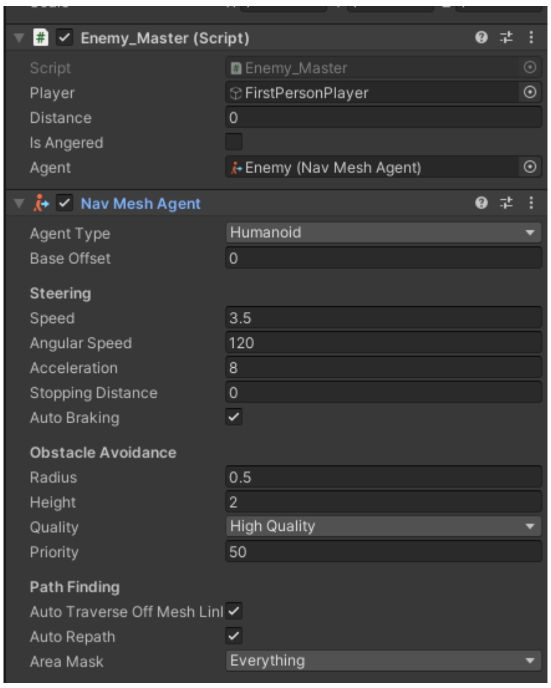
Enemy
Tag Untagged Layer Default

Transform

	X	Y	Z
Position	0	1.77	0
Rotation	0	0	0
Scale	1	1	1

Enemy_Master (Script)

Script	Enemy_Master
Player	FirstPersonPlayer
Distance	0
Is Angered	
Agent	None (Nav Mesh Agent)



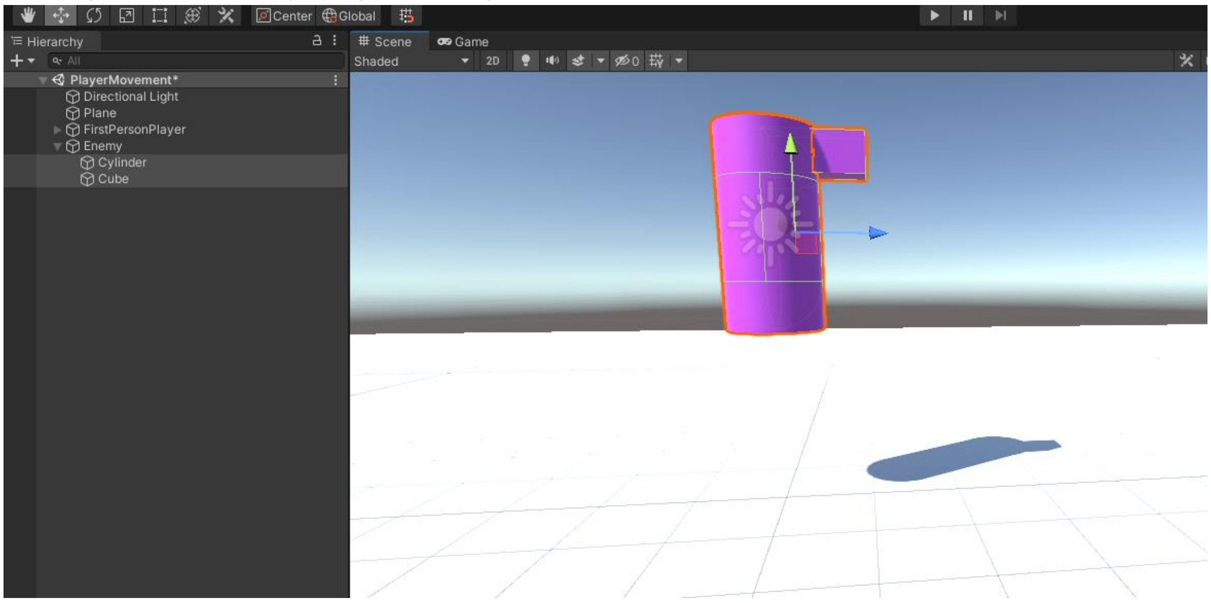
Enemy_Master (Script)

Script	Enemy_Master
Player	FirstPersonPlayer
Distance	0
Is Angered	
Agent	Enemy (Nav Mesh Agent)

Nav Mesh Agent

Agent Type	Humanoid
Base Offset	0
Steering	
Speed	3.5
Angular Speed	120
Acceleration	8
Stopping Distance	0
Auto Braking	✓
Obstacle Avoidance	
Radius	0.5
Height	2
Quality	High Quality
Priority	50
Path Finding	
Auto Traverse Off Mesh Linl	✓
Auto Repath	✓
Area Mask	Everything

Due to the Nav Mesh being higher up than the enemy model I've raised up the actual objects of the enemy (not the empty called enemy). When I click play the enemy shouldn't be in the ground but this might take some adjusting to look right.



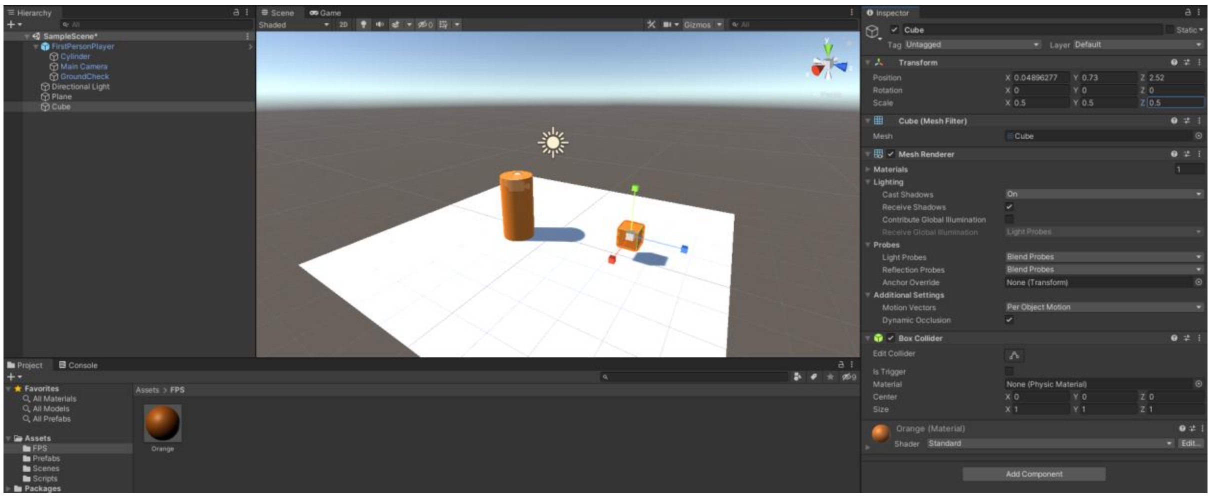
When you click play it should work. You can walk close to it and it will turn and start to follow. Or if your using a block to represent the player, click on scene view, select the block and move it around, close and away to see the enemy react.

Oliver Mota

Tutorial 4: Block Pick-up / Drop

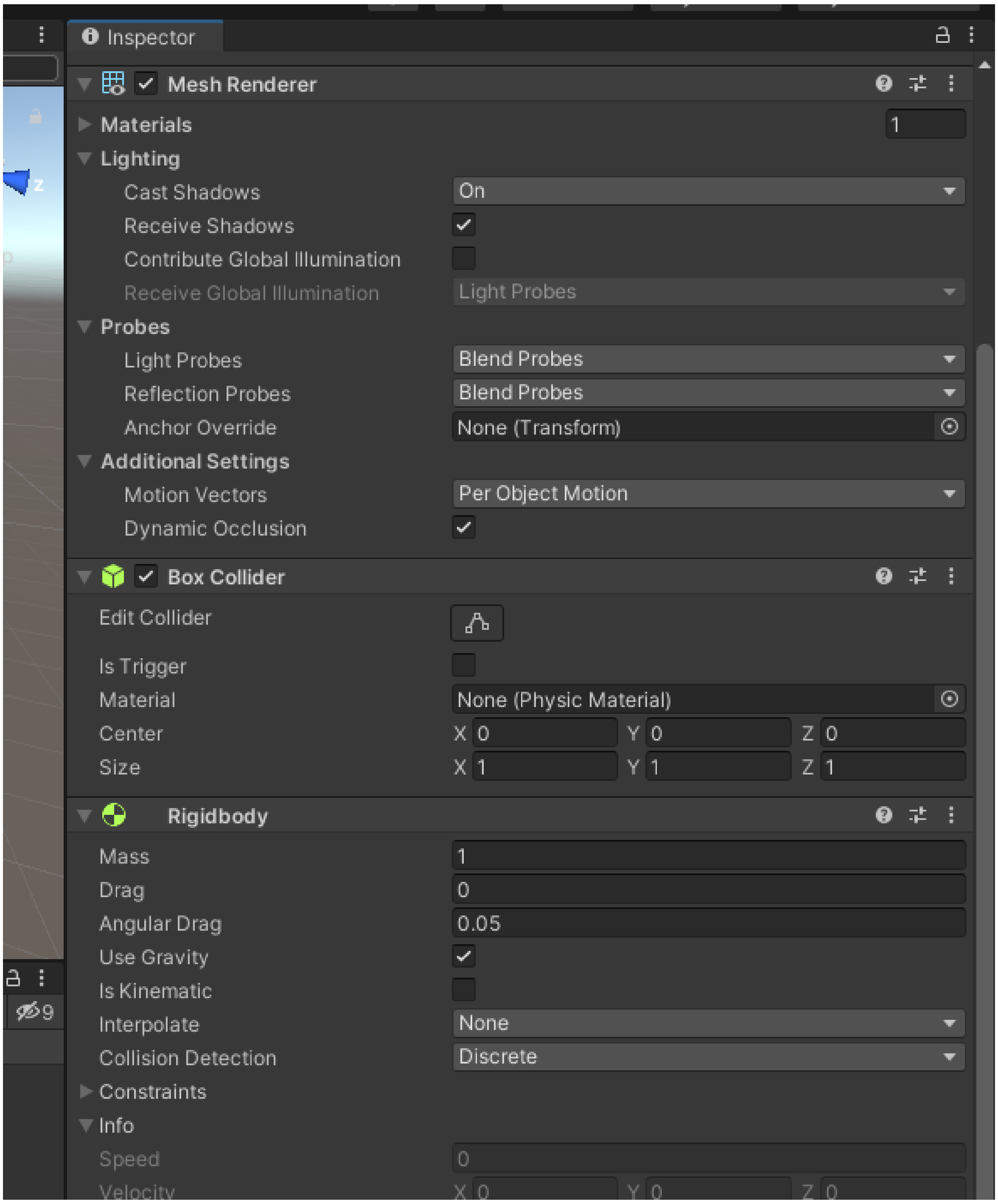
For my final tutorial I will be making a simple block pick-up/drop script. It works by the player camera and the block having direct line of sight, this means it would be tricky to use this script in a third person game.

To start with I'm using a first person controller I have made, you can download one or follow the previous tutorial. I have imported my FPC and created a plane as the ground. Next I've added a cube object which we will pick up. I've halved the size of the cube in the inspector by setting its scale to 0.5, 0.5, 0.5.



```
Cube                          Static
Tag Untagged     Layer Default
Transform
Position   X 0.04896277   Y 0.73   Z 2.52
Rotation   X 0            Y 0      Z 0
Scale      X 0.5          Y 0.5    Z 0.5
Cube (Mesh Filter)
Mesh                          Cube
Mesh Renderer
Materials                     1
Lighting
  Cast Shadows                On
  Receive Shadows             ✓
  Contribute Global Illumination
  Receive Global Illumination Light Probes
Probes
  Light Probes                Blend Probes
  Reflection Probes           Blend Probes
  Anchor Override             None (Transform)
Additional Settings
  Motion Vectors              Per Object Motion
  Dynamic Occlusion           ✓
Box Collider
Edit Collider
Is Trigger
Material                      None (Physic Material)
Center     X 0   Y 0   Z 0
Size       X 1   Y 1   Z 1
Orange (Material)
Shader   Standard            Edit...
```

Before I make the script I need to set up a couple of things, the first is the box needs a box collider to the box so we can actually interact with it and a rigidbody so when we drop it, its affected by gravity and drops to the ground. It should spawn with a box collider and we need to apply a rigidbody. To get the rigidbody select the box and under the inspector I need to click 'Add Component' then search for 'rigidbody' select that.



Inspector

Mesh Renderer ✓

Materials	1
Lighting	
Cast Shadows	On
Receive Shadows	✓
Contribute Global Illumination	
Receive Global Illumination	Light Probes
Probes	
Light Probes	Blend Probes
Reflection Probes	Blend Probes
Anchor Override	None (Transform)
Additional Settings	
Motion Vectors	Per Object Motion
Dynamic Occlusion	✓

Box Collider ✓

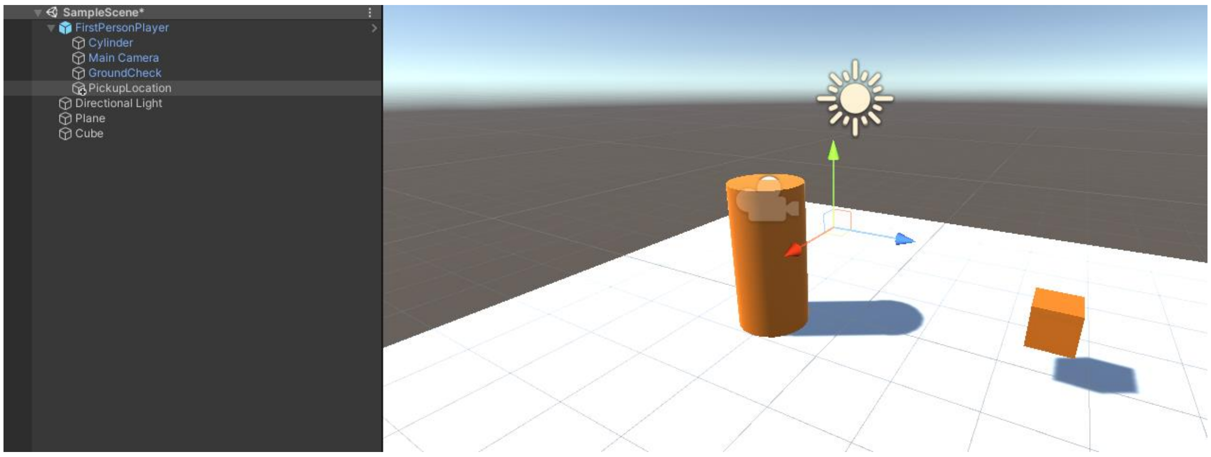
Edit Collider	
Is Trigger	
Material	None (Physic Material)
Center	X 0 Y 0 Z 0
Size	X 1 Y 1 Z 1

Rigidbody

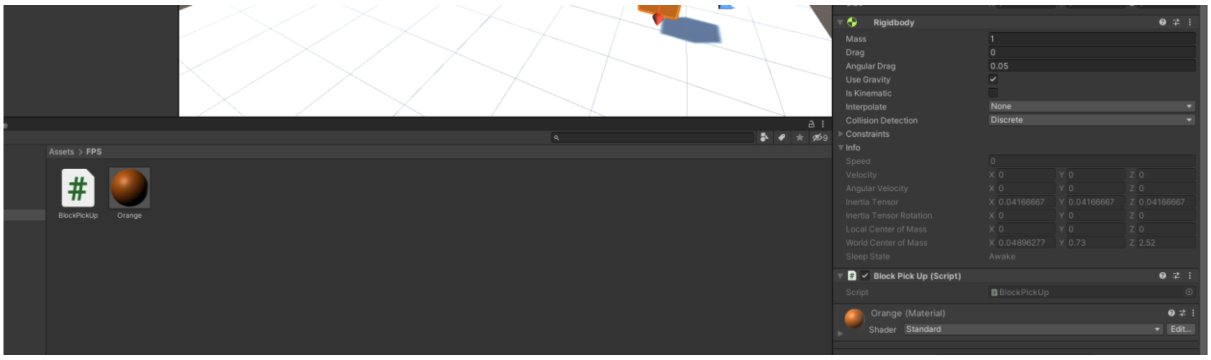
Mass	1
Drag	0
Angular Drag	0.05
Use Gravity	✓
Is Kinematic	
Interpolate	None
Collision Detection	Discrete
Constraints	
Info	
Speed	0
Velocity	X 0 Y 0 Z 0

Then I need to create an empty gameobject as a child object of the player. For this, select the player and right click, then empty game object, I've called this 'pickuplocation'. This is where the block will teleport to once we've picked it up. Making it a child means this location will move with the player rather than it being a static point in the world.

Oliver Mota



Now I’m going to make the script. Either select the box and click ‘add component’, search for script and call it ‘blockpickup’ or create a script in the project files by right clicking at the bottom of the screen, naming it and dragging into the inspector element of the box.



Rigidbody

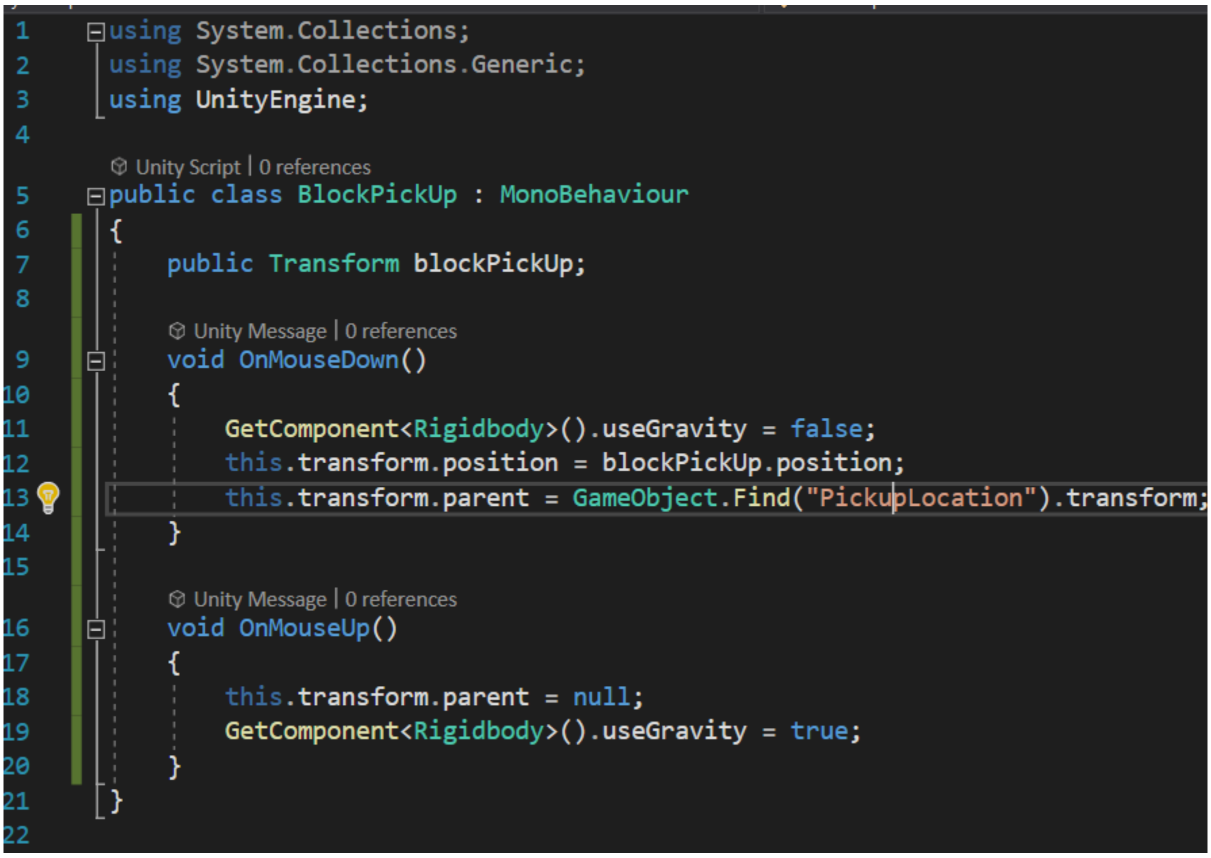
Mass	1
Drag	0
Angular Drag	0.05
Use Gravity	✓
Is Kinematic	
Interpolate	None
Collision Detection	Discrete

▸ Constraints
▾ Info

Speed	0		
Velocity	X 0	Y 0	Z 0
Angular Velocity	X 0	Y 0	Z 0
Inertia Tensor	X 0.04166667	Y 0.04166667	Z 0.04166667
Inertia Tensor Rotation	X 0	Y 0	Z 0
Local Center of Mass	X 0	Y 0	Z 0
World Center of Mass	X 0.04896277	Y 0.73	Z 2.52
Sleep State	Awake		

✓ Block Pick Up (Script)
Script BlockPickUp

Orange (Material)
Shader Standard Edit...



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

Unity Script | 0 references
public class BlockPickUp : MonoBehaviour
{
    public Transform blockPickUp;

    Unity Message | 0 references
    void OnMouseDown()
    {
        GetComponent<Rigidbody>().useGravity = false;
        this.transform.position = blockPickUp.position;
        this.transform.parent = GameObject.Find("PickupLocation").transform;
    }

    Unity Message | 0 references
    void OnMouseUp()
    {
        this.transform.parent = null;
        GetComponent<Rigidbody>().useGravity = true;
    }
}
```

Back in unity, an empty slot has appeared on the script on the box, which is where we need to drag out pickup location. Drag the pickuplocation gameobject from under the player and drag it into the slot.

Make sure the names of everything is correct, including capital letters. The “PickupLocation” in the script needs to be the same as the ‘PickupLocation’ empty gameobject in the scene. Otherwise the block will not move with the player.

That’s the end of my tutorials!