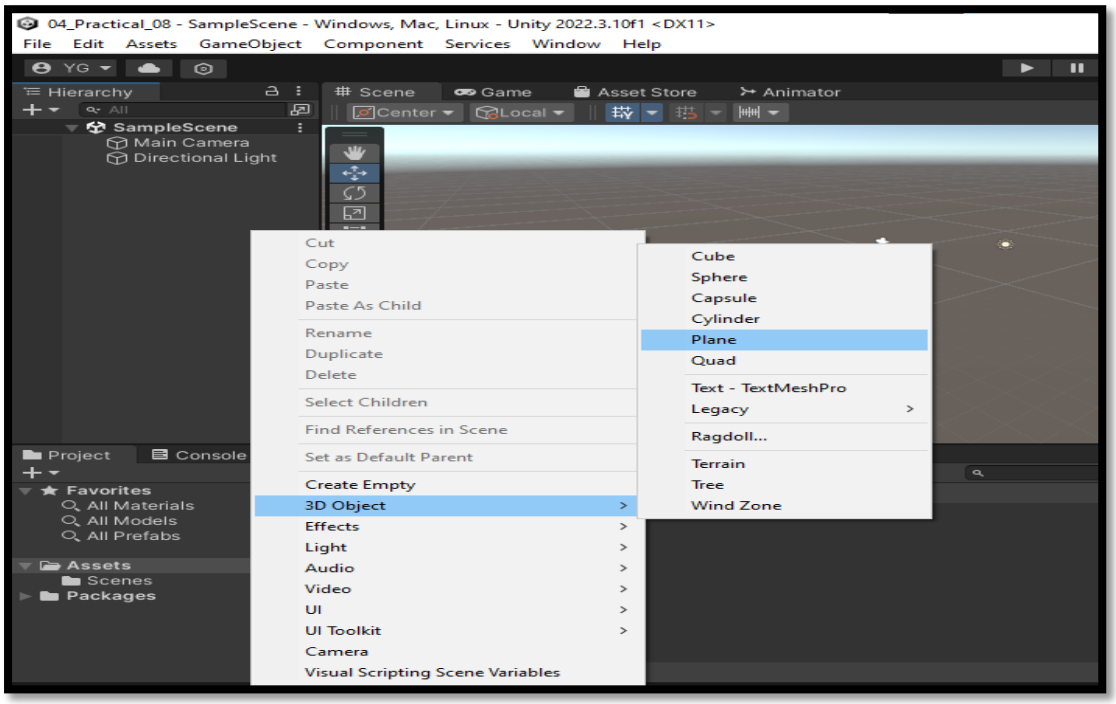
Practical No. 08

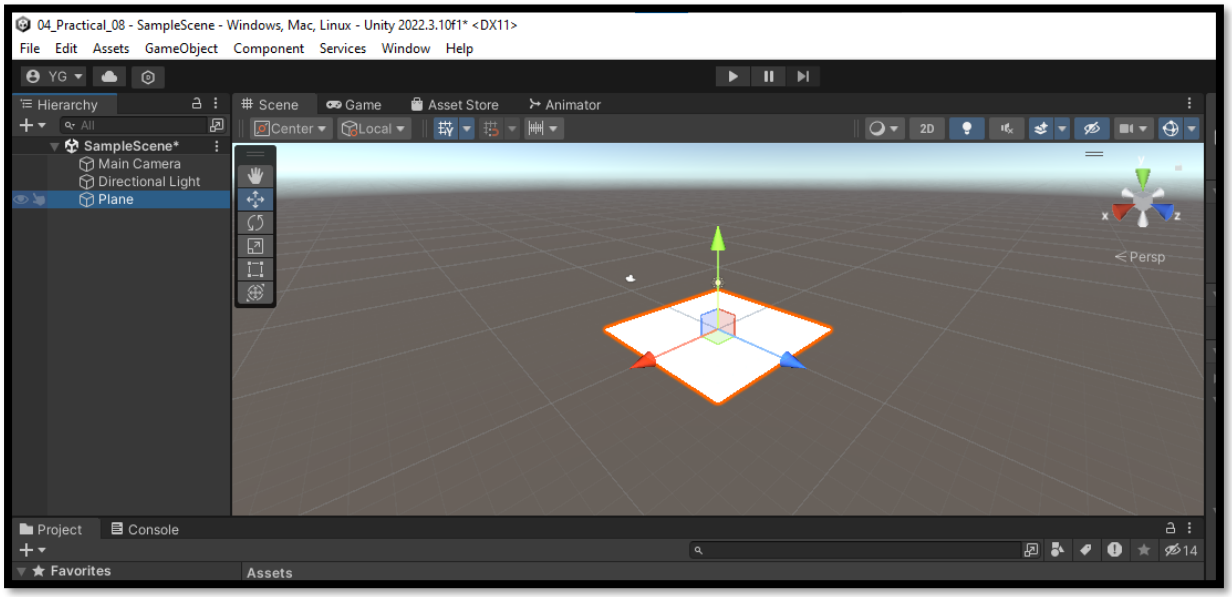
Aim: Implement Rotating a Character in the Direction of Movement.

Steps:

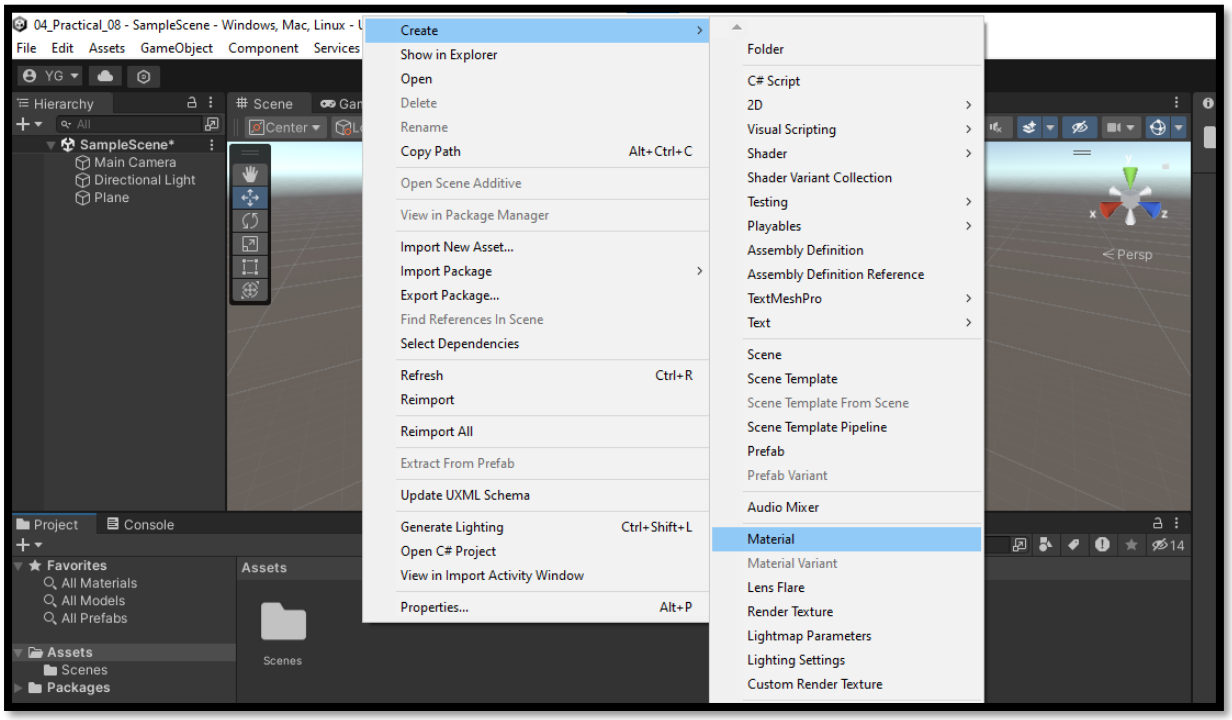
Setting up the scene

Step 1: Firstly, add a plane gameObject.

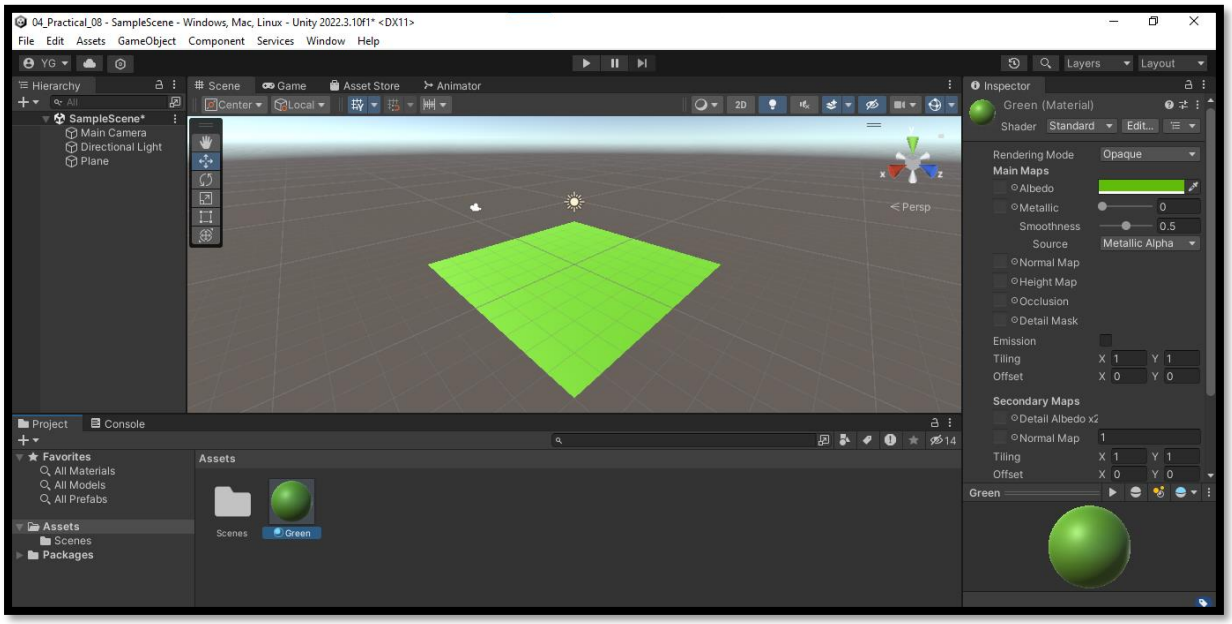




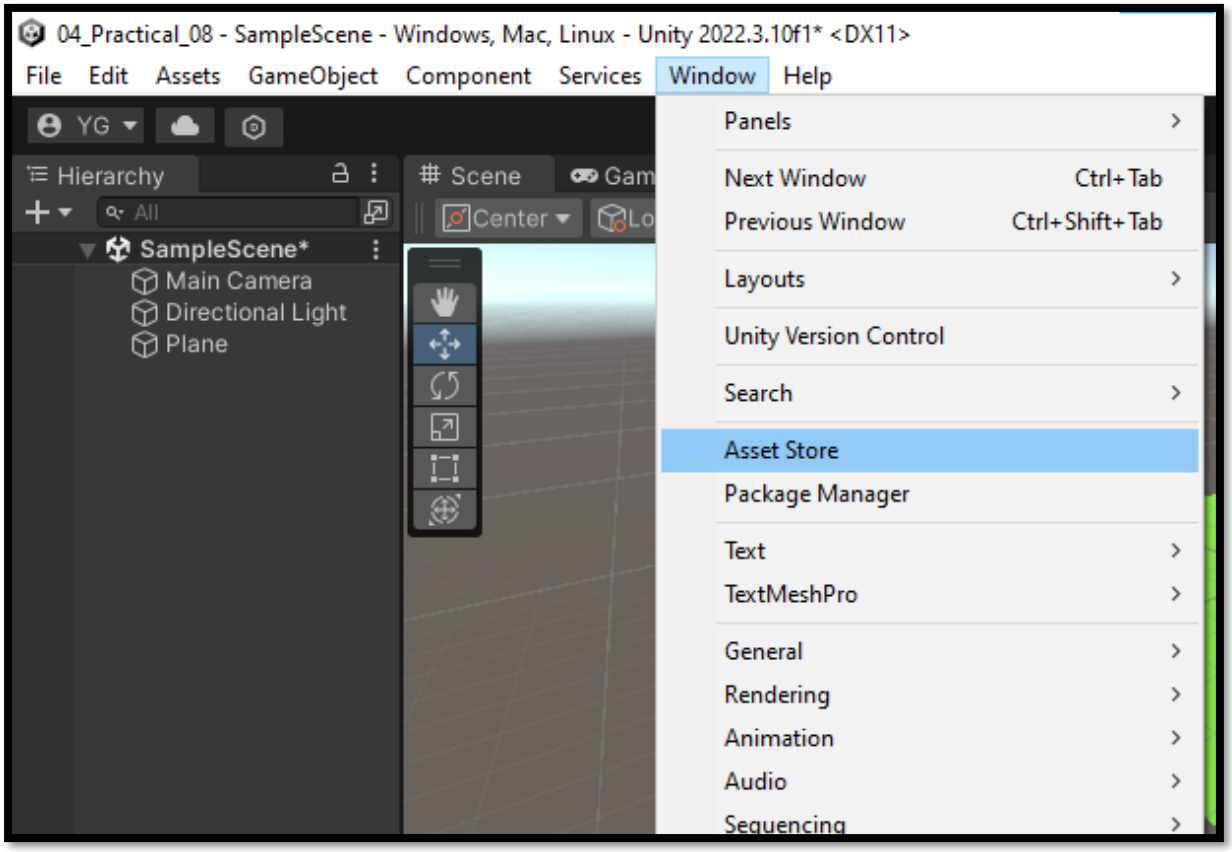
Step 2: Add material for plane.



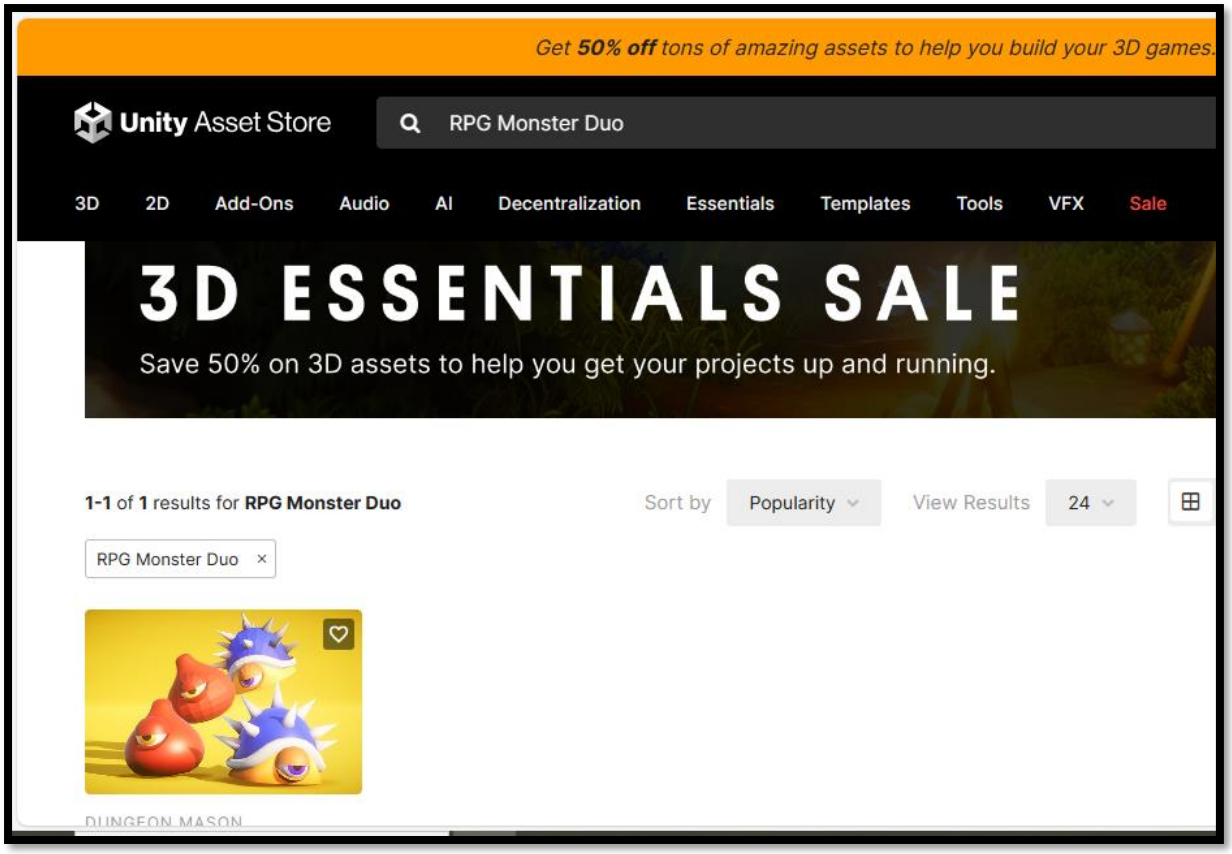
Drag and drop the Green Color on the plane.



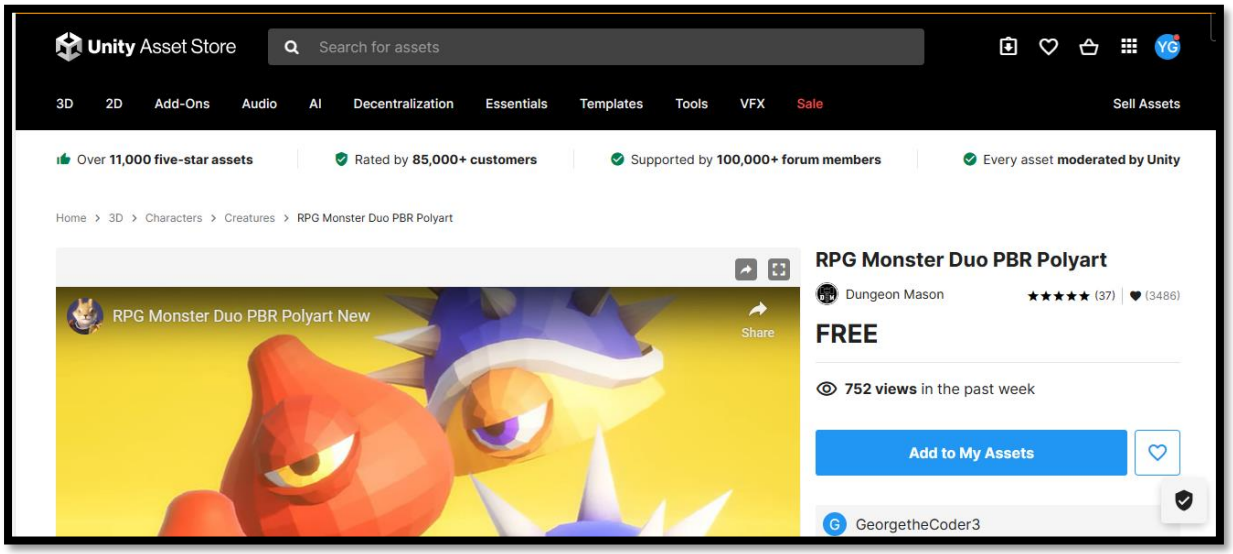
Step 3: Click on the window panel and select Asset Store.



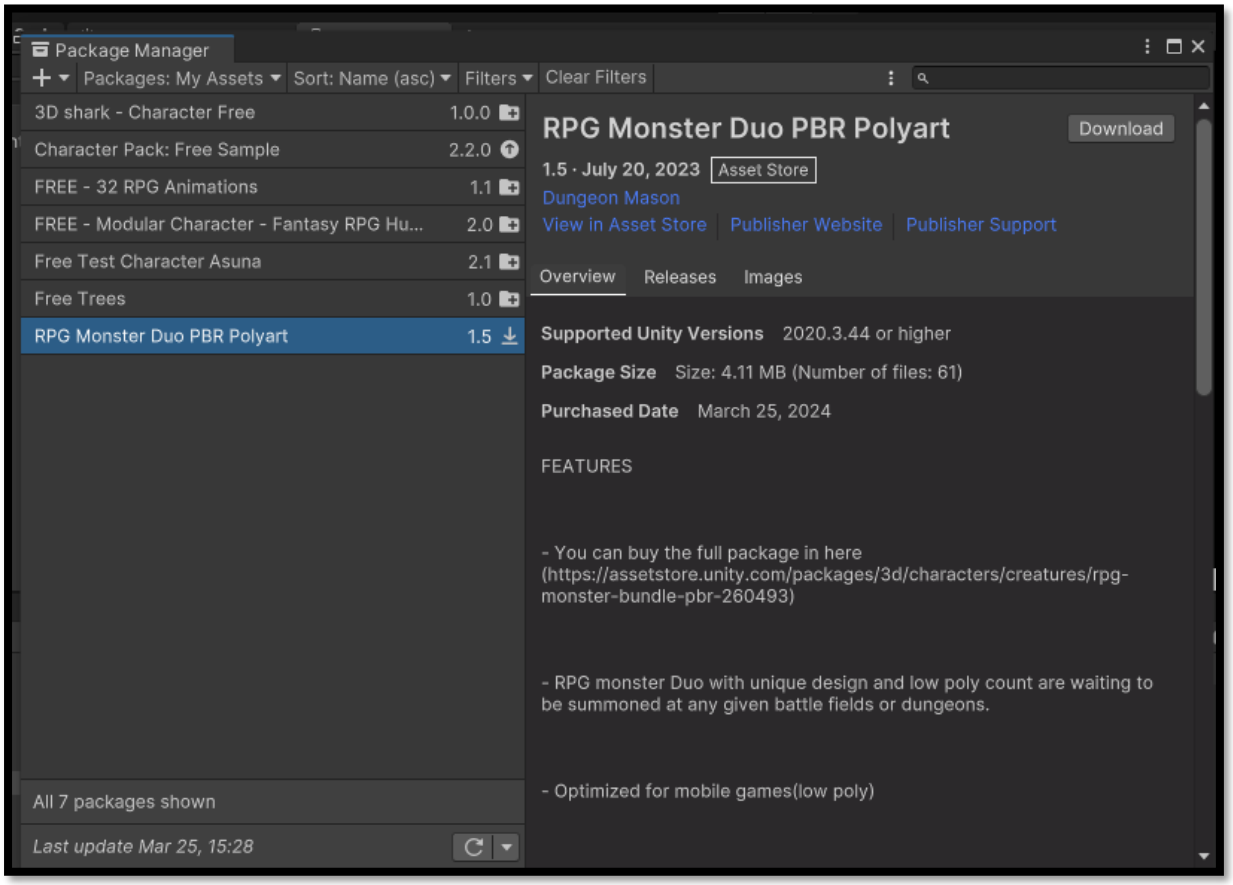
Step 4: In Asset Store search for RPG Monster Duo.



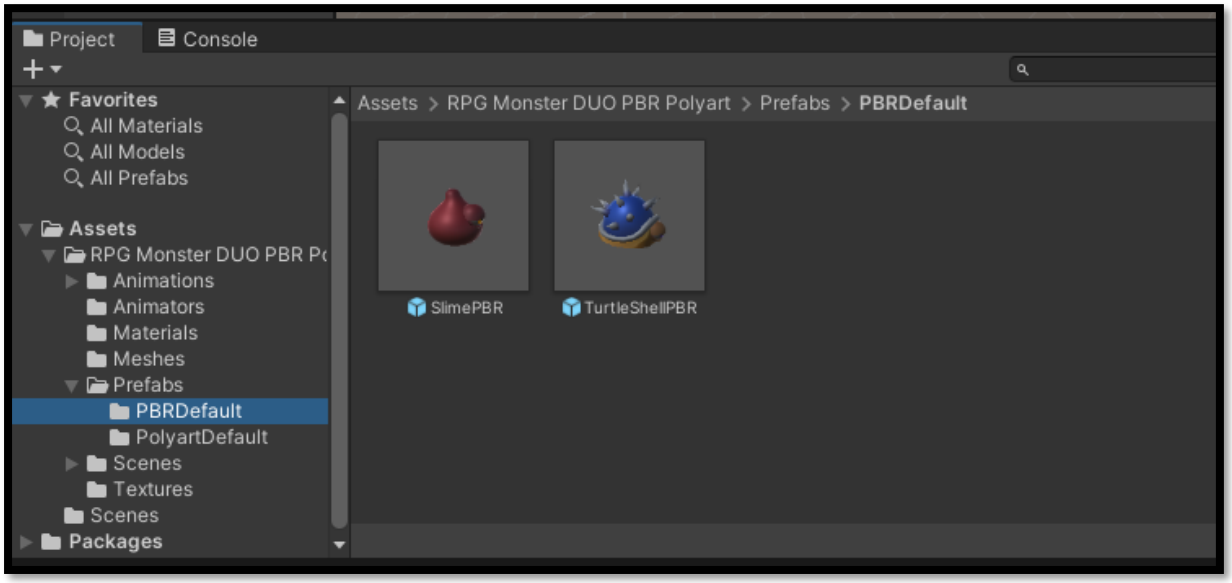
Step 5: Click on the Add to My Assets.



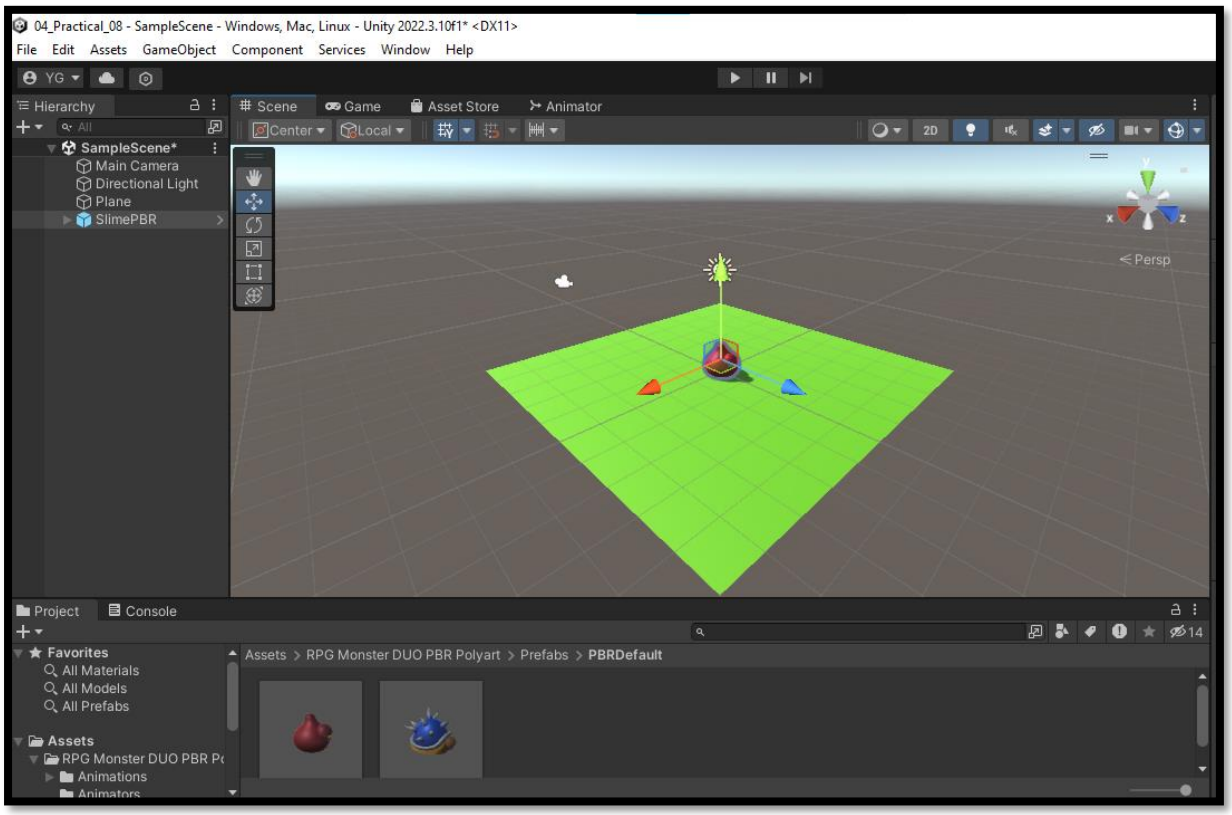
Step 6: Download and import this asset.



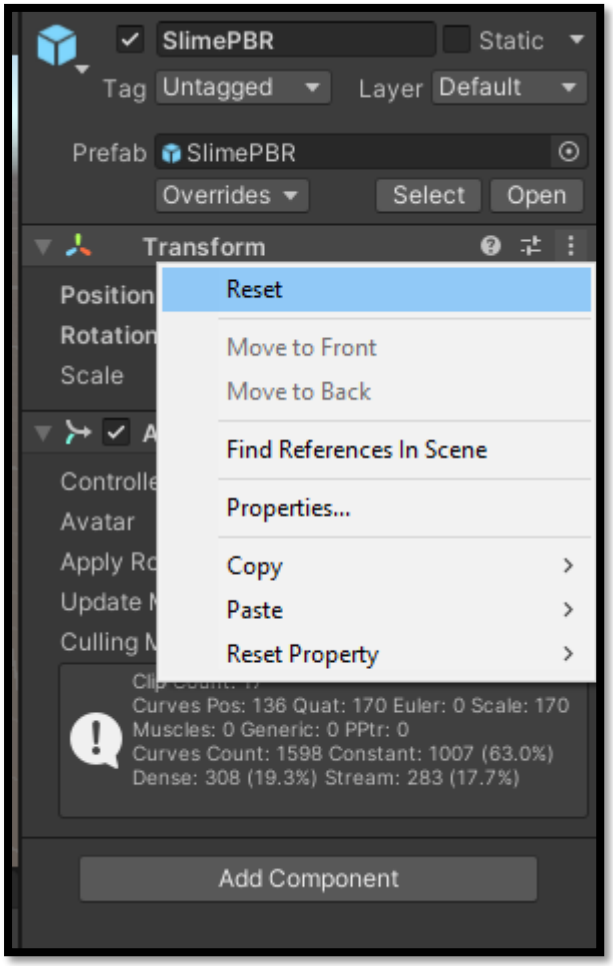
Step 7: In Assets you will have your RPG monster assets folder.



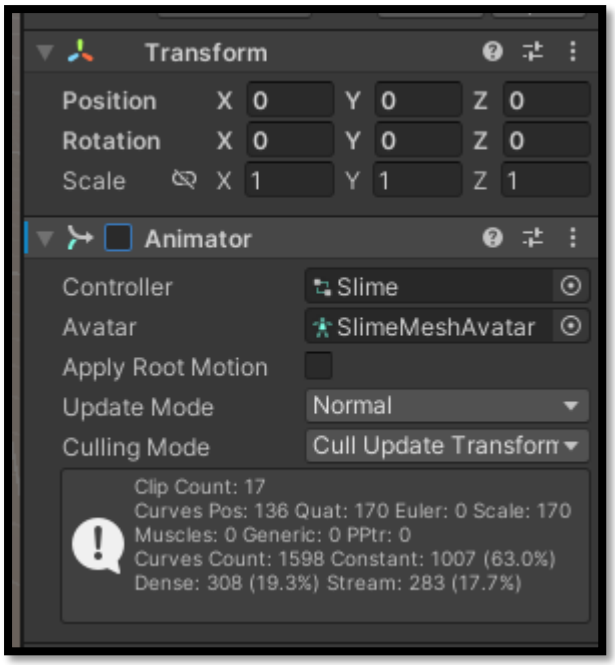
Step 8: Go to Prefab and drag the character to scene.



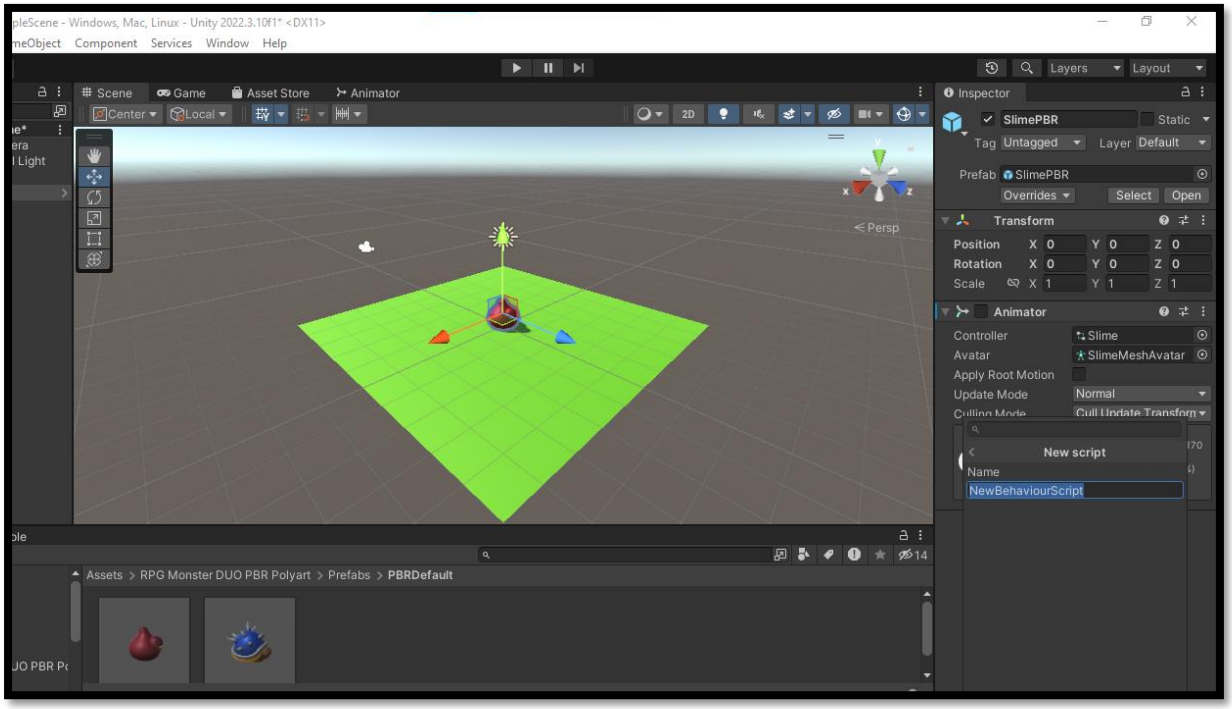
Step 9: Reset Transform for monster character

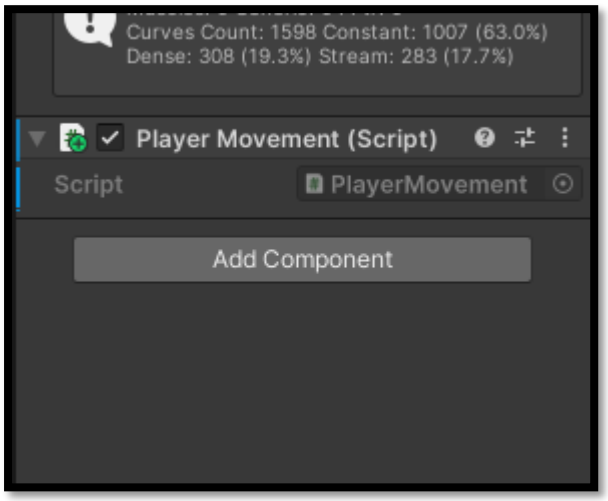


Step 10: Disable / Uncheck animator component.



Step 11: Creating a script to make the character move.



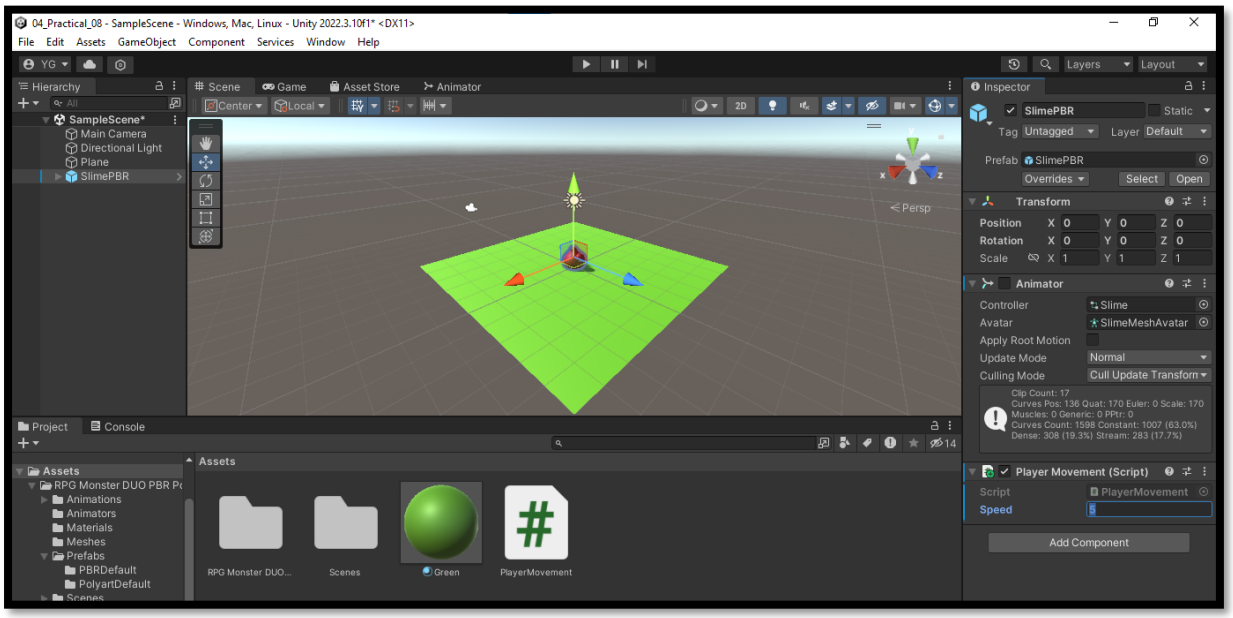


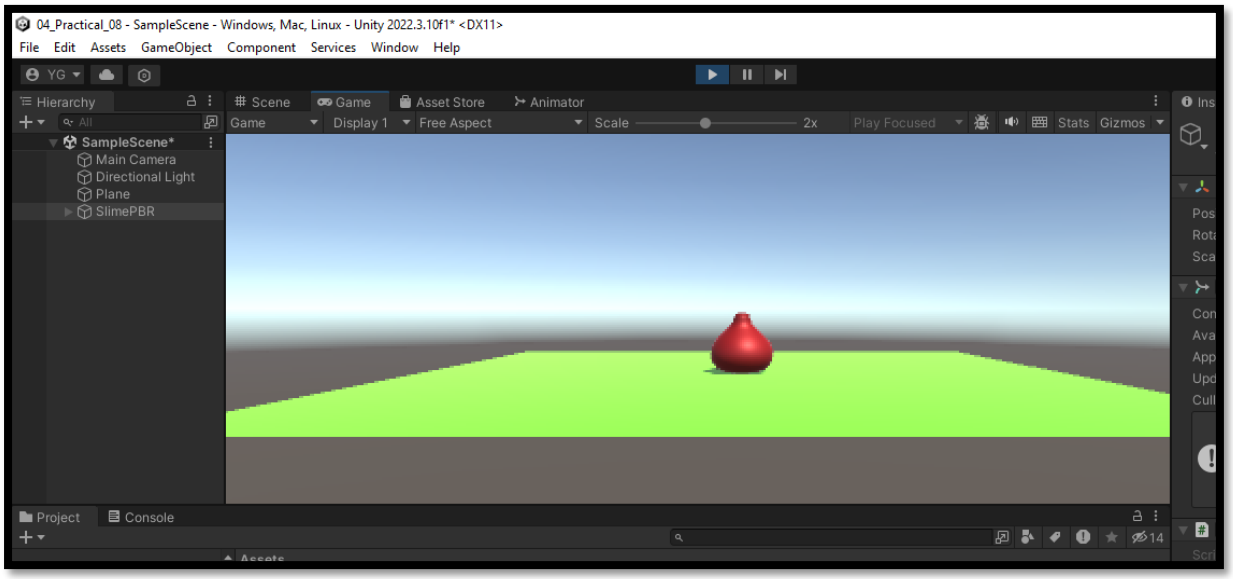
Step 12: Double click on script and write below code in Visual Studio.

Code:

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

public class PlayerMovement : MonoBehaviour
{
   public float speed; //determines how fast the player will move.
   // Start is called before the first frame update
   void Start()
   {

   }

   // Update is called once per frame
   void Update()
   {
      // Gets the horizontal input from the player (e.g., left or right arrow keys, or A/D keys).
      float horizontalInput = Input.GetAxis("Horizontal");

      // Gets the vertical input from the player (e.g., up or down arrow keys, or W/S keys).
      float verticalInput = Input.GetAxis("Vertical");

      // Creates a new Vector3 variable representing the direction the player should move
based on the input.
      Vector3 movementDirection = new Vector3(horizontalInput, 0, verticalInput);

      // Normalizes the movement direction vector to ensure consistent speed regardless of
direction (diagonal or straight).
      movementDirection.Normalize();

      //Moves the player object based on the normalized movement direction, speed, and the
time since the last frame.
      transform.Translate(movementDirection * speed * Time.deltaTime, Space.World);
   }
}
```

Step 13: Now go to Unity and set speed to 5 and play in game mode.





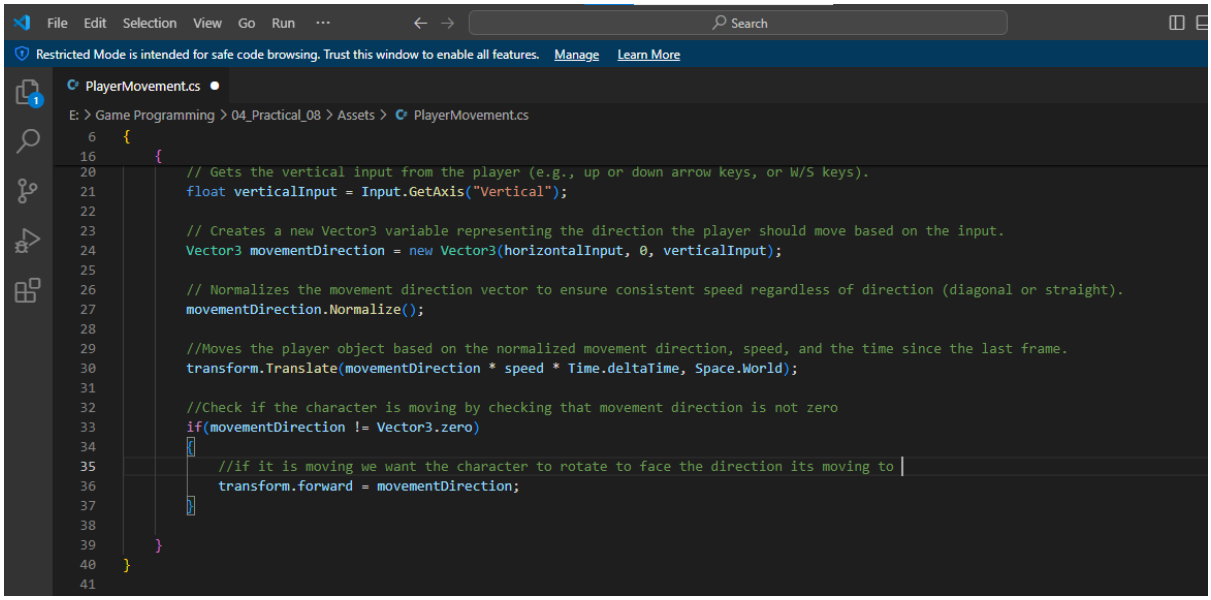
And here your character is moving but as the changes are done the character face isn't visible.

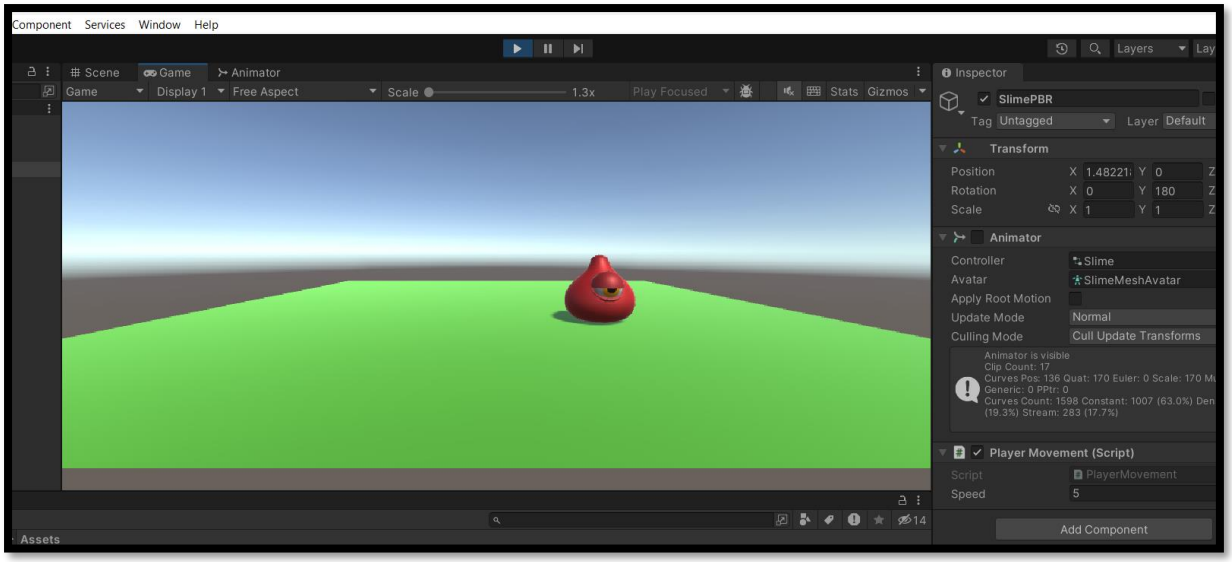
Step 14: Changing the forward direction of the Character's Transform to face the direction of movement

Add following marked line of code to check and added save and come back to Unity to check if this works.

Code:

```
//Check if the character is moving by checking that movement direction is not zero
if(movementDirection != Vector3.zero)
{
   //if it is moving we want the character to rotate to face the direction its moving to
   transform.forward = movementDirection;
}
```





Using Quaternion.RotateTowards to smoothly change the rotation of the character

Code:

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

public class PlayerMovement : MonoBehaviour
{
   public float speed; //determines how fast the player will move.
   public float rotationSpeed;// Determines how fast the player will rotate.
   // Start is called before the first frame update
   void Start()
   {

   }

   // Update is called once per frame
   void Update()
   {
      // Gets the horizontal input from the player (e.g., left or right arrow keys, or A/D keys).
      float horizontalInput = Input.GetAxis("Horizontal");

      // Gets the vertical input from the player (e.g., up or down arrow keys, or W/S keys).
      float verticalInput = Input.GetAxis("Vertical");

      // Creates a new Vector3 variable representing the direction the player should move
based on the input.
      Vector3 movementDirection = new Vector3(horizontalInput, 0, verticalInput);

      // Normalizes the movement direction vector to ensure consistent speed regardless of
direction (diagonal or straight).
      movementDirection.Normalize();

      //Moves the player object based on the normalized movement direction, speed, and the
time since the last frame.
      transform.Translate(movementDirection * speed * Time.deltaTime, Space.World);

      //Check if the character is moving by checking that movement direction is not zero
      if(movementDirection != Vector3.zero)

      {
         // Calculate the rotation to look in the movement direction
         Quaternion toRotation = Quaternion.LookRotation(movementDirection, Vector3.up);


         // Smoothly rotate towards the calculated rotation based on rotation speed
```

```
            transform.rotation = Quaternion.RotateTowards(transform.rotation, toRotation,
rotationSpeed * Time.deltaTime);
        }
    }
}
```