***Basic player Jumping Mechanic Tutorial in unity***

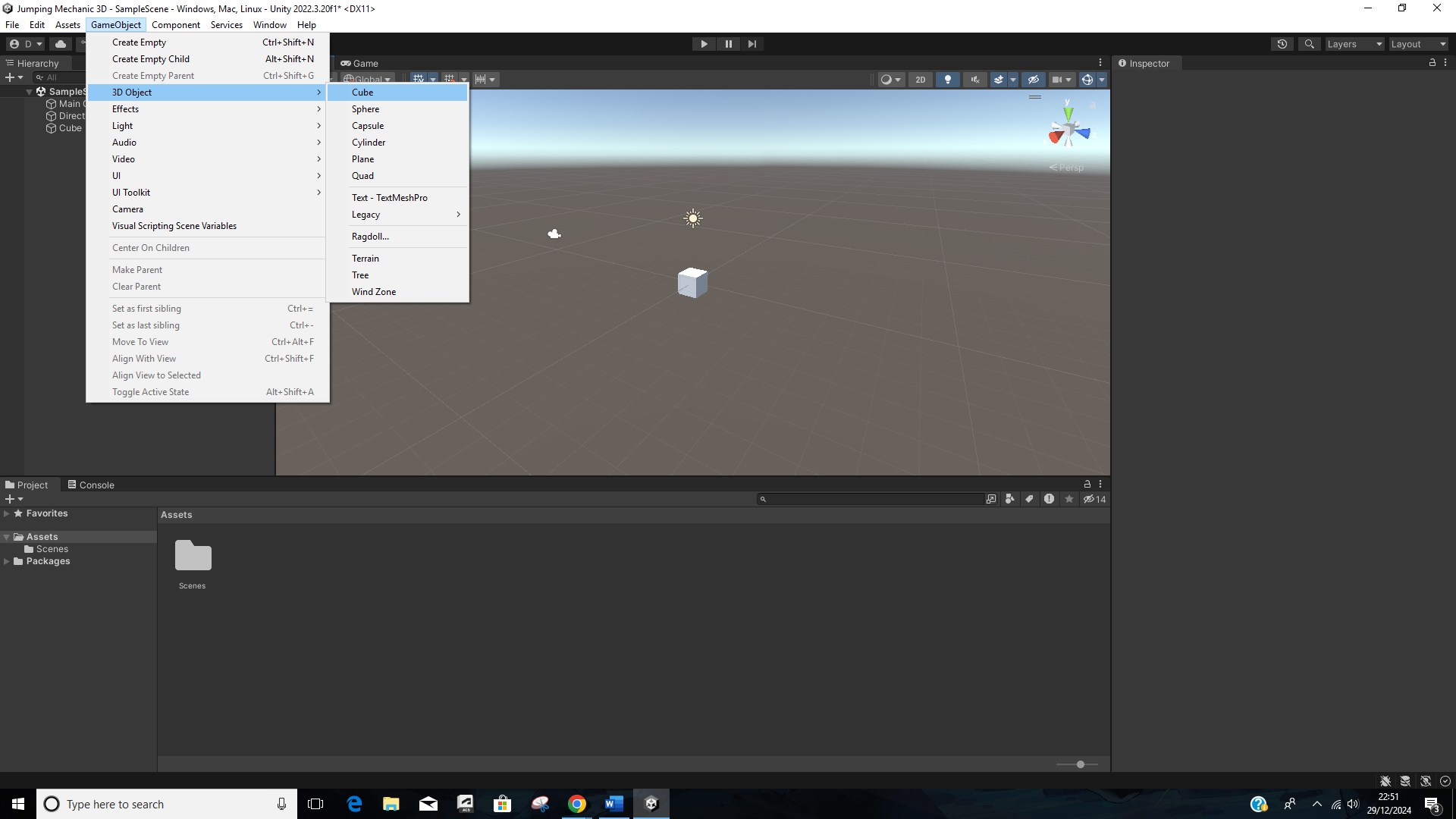
Introduction:

This tutorial will teach you how to make a game object jump when the player presses a specific key, such as the spacebar. It uses Unity's physics system to achieve realistic movement.

**Step 1: Setting up the Game Object**

**1.Create a 3D Objects:**

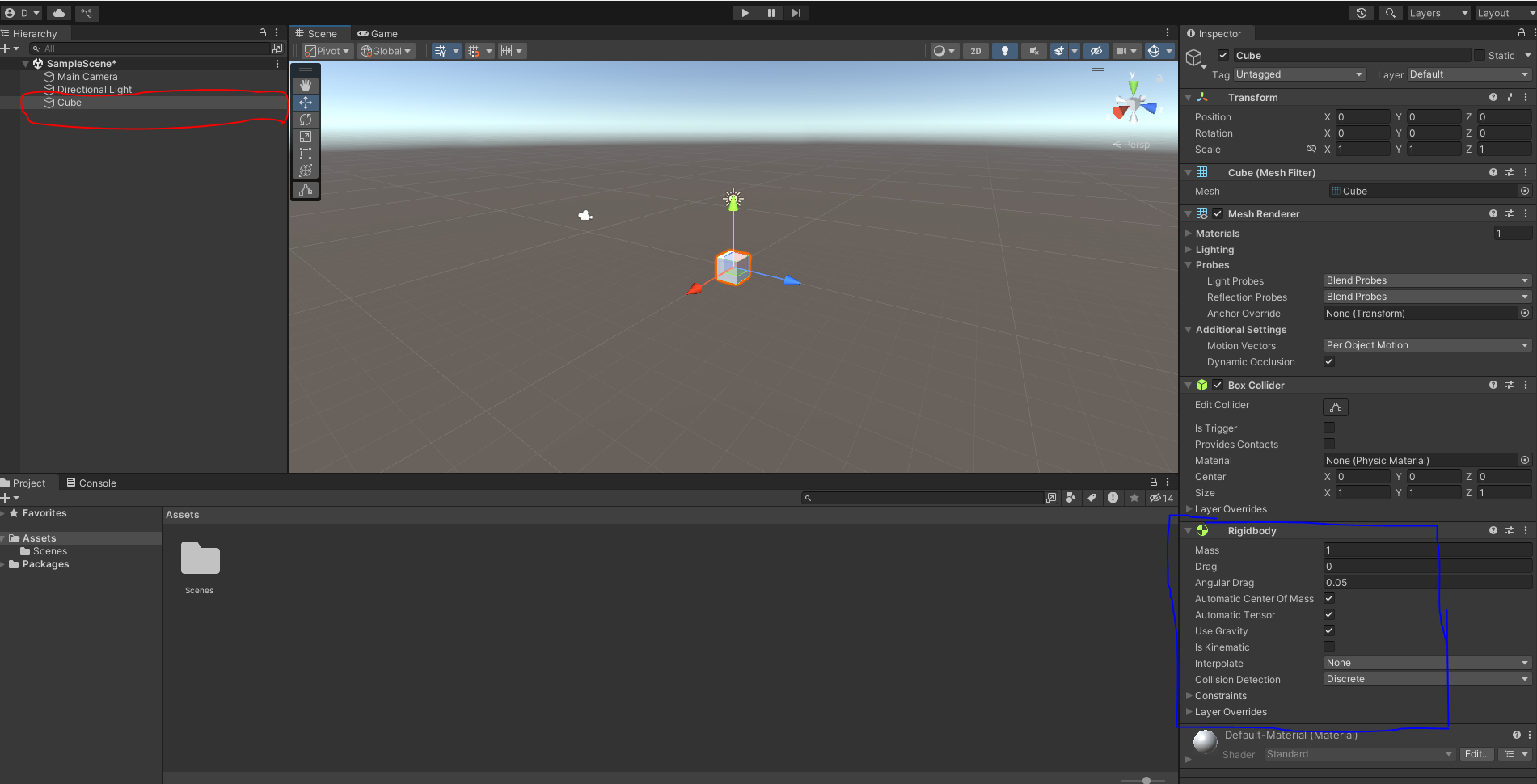
* Open Unity, go to the Game Object menu, and create a 3D object such as a Cube.



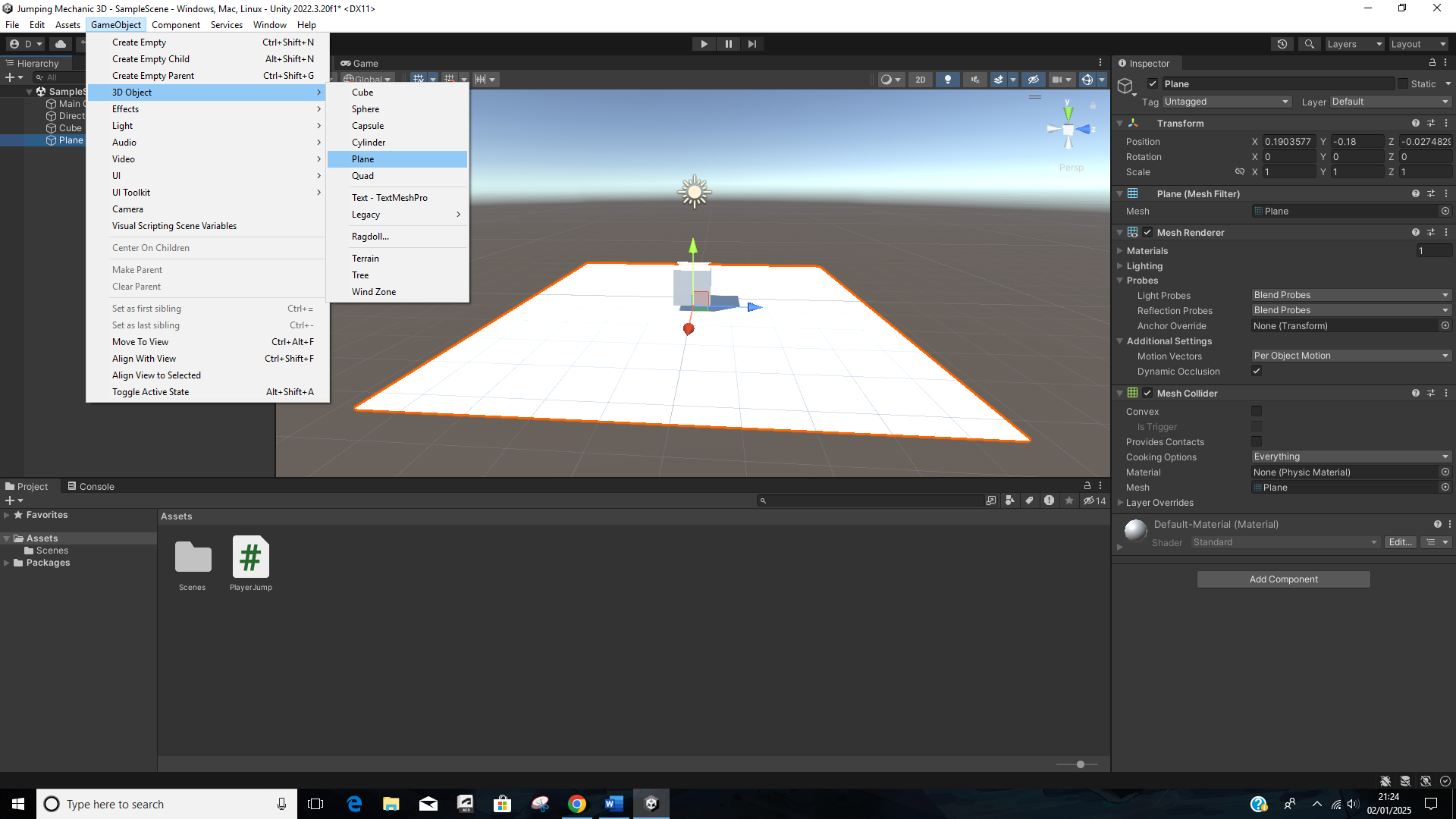
* This will serve as the player character.

**2**. **Add a Rigid body Component:**

* Select the cube in the Hierarchy.
* In the Inspector, click Add Component and add a Rigid body component.
* This allows the cube to interact with Unity's physics system.



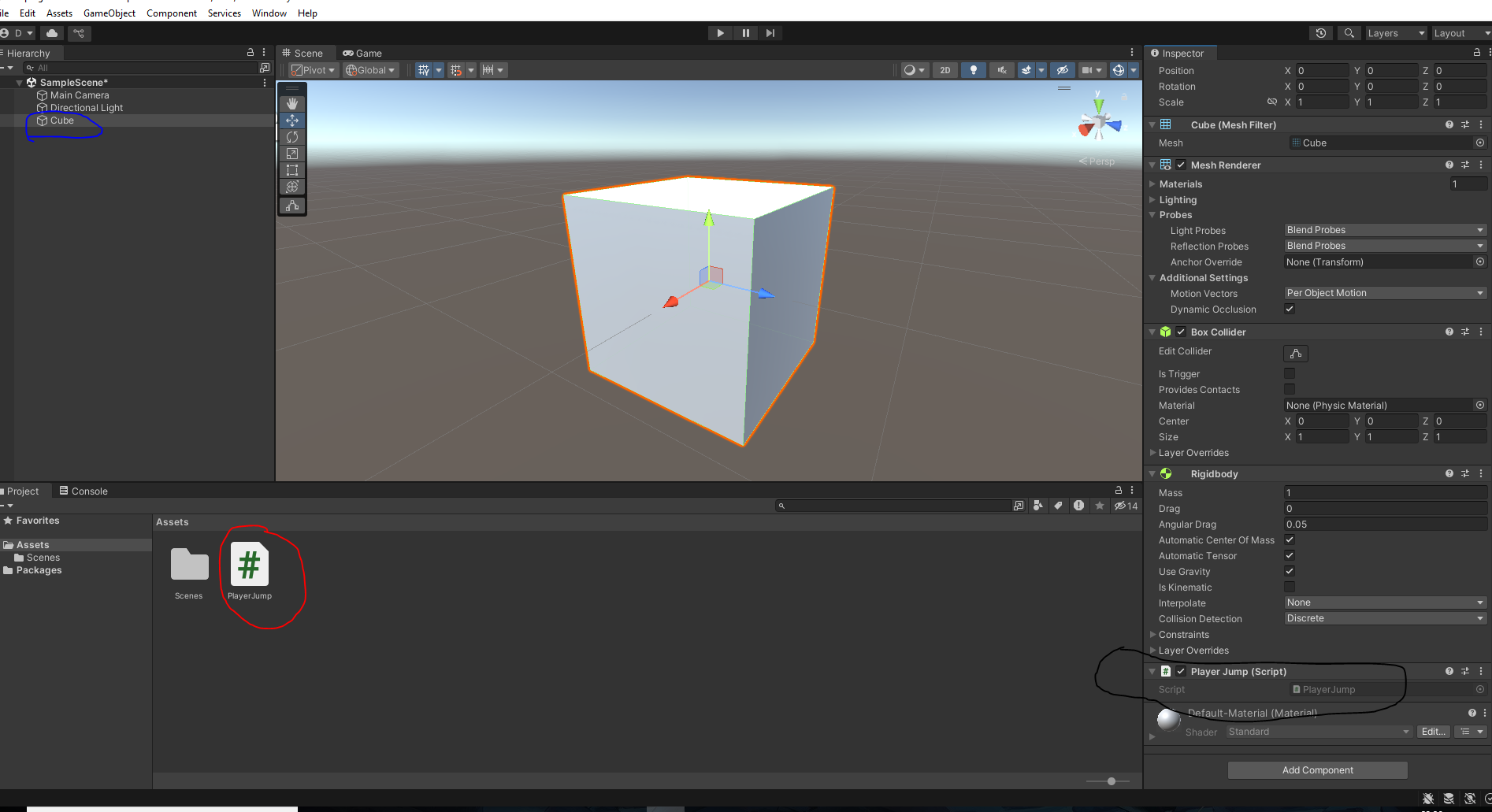
**3.** Next Create a plane so that the cube has a surface to land on because of the rigid body.



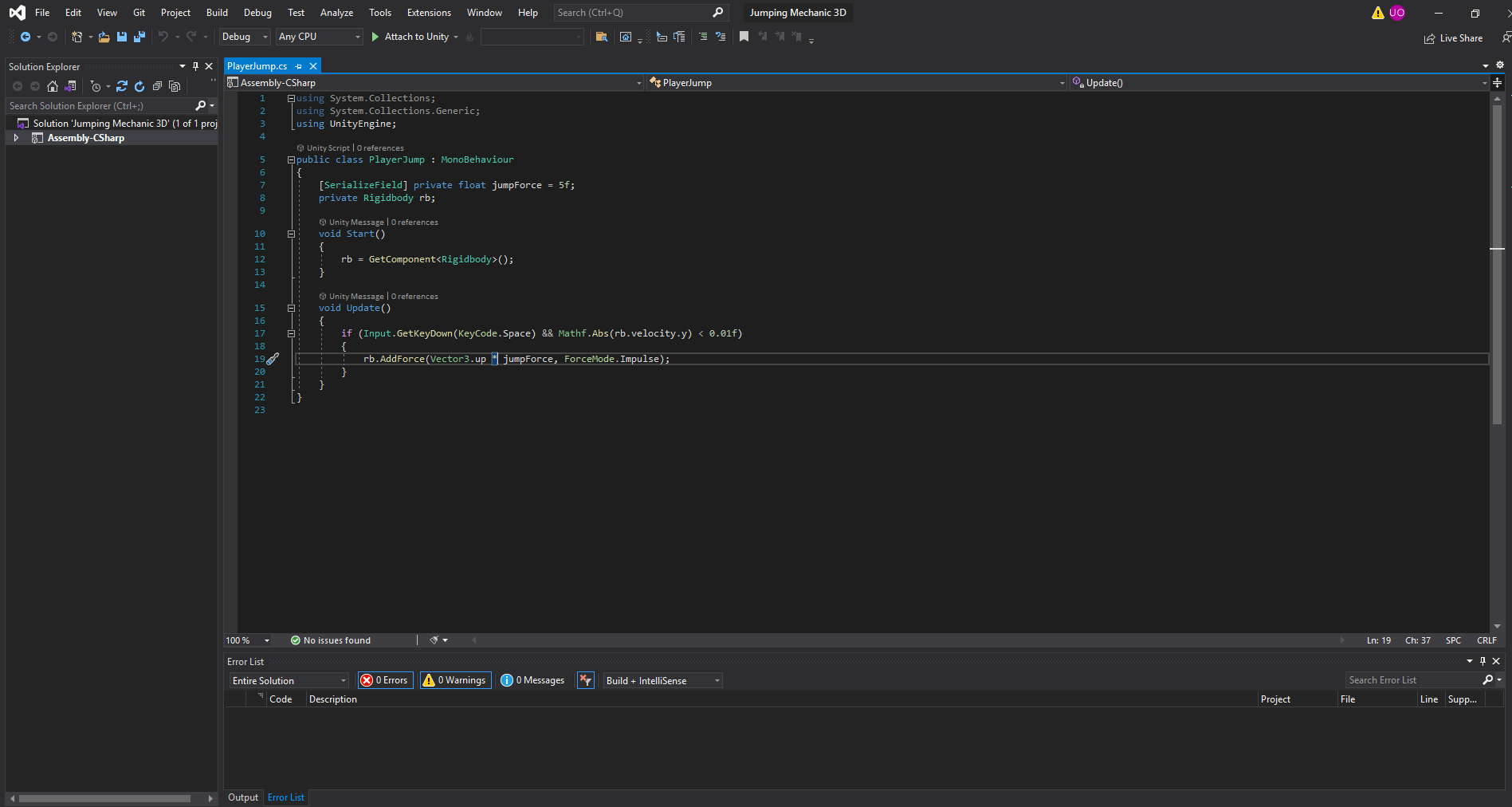
**Step 2: Creating the Script**

**1.*Create a C# Script:***

* In the Assets folder, create a new C# Script and name it PlayerJump.
* Attach the script to the cube by dragging it into the Inspector.



**2. *Script Explanation:***



1. ***Setting up the script***

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

* These lines import Unity's libraries so you can use features like physics, input handling, and game objects.

1. ***Declaring the class***

public class PlayerJump : MonoBehaviour

* PlayerJump: The name of the class. It matches the name of the script file.
* MonoBehaviour: This means the script is a unity component and can be attached to game objects

1. ***Declaring the Variables***

[SerializeField] private float jumpForce = 5f;

private Rigidbody rb;

* [SerializeField] This allows you to edit the jumpForce value directly in Unity's Inspector without making it public.
* private Float jumpforce = 5f; This sets the strength of the jump. You can adjust it to make the jump higher or lower.
* Private Rigidbody rb: This creates a variable to store the **Rigidbody** component of the game object. The Rigidbody handles physics like movement and gravity.

1. ***The Start Method***

* The Start () method runs once when the game starts.
* GetComponenet<Rigidbody>(): This finds the Rigidbody attached to the same game object as the script and assigns it to the rb variable. This allows us to control the object’s physics.

1. ***The Update Method and the code***

* **Update()** runs every frame of the game which make it ideal for checking player input.
* This means the code inside the Update() is constantly checking for actions like key presses.
* **Input.GetKeyDown** checks if the player pressed down a specific key at that exact moment.
* **KeyCode.Space** refers to the spacebar key.
* This condition ensures that the jump happens only when the spacebar is pressed.
* **Rb.velocity.y** this checks the vertical speed (y-axis) of the rigidbody
* **Mathf.Abs()** ensures the value is treated as positive, even if the object is falling (negative velocity).
* **<0.01f** this ensures the object is almost not moving vertically before allowing another jump. This is because it prevents the player from jumping while already in the air.
* **Rb.AddForce** this applies a force to the rigidbody to make the object move.
* **Vector3.up** This refers to the upward direction in unity’s 3D space (y-axis)
* **Jumpforce** the strength of the jump which you set earlier in the script.
* **ForceMode.Impulse** This tells unity to apply the force instantly, giving the object a sudden push.

**Conclusion:**

Now when you press space the script checks if the spacebar is pressed. It checks if the object is not already in the air. It applies an upward force making the object jump. In turn this makes sure a smooth and quite realistic jumping mechanic.

