BEFORE STARTING

This tutorial assumes you already have a working player model in an existing 3D project with an interactable environment. It is recommended to have ramps, stairs, and ledges. For the sake of this tutorial, we will be using the First Person Movement tutorial but the Third Person should work just as well.

PART 1: GRAVITY

As it is now, your character should be able to climb tiny steps and ascend slopes (default 45 degrees). These values can be played around with from the Player Object's Character Controller component (Slope Limit and Step Offset). The issue is that gravity will not pull the character back down to the ground.

Add an empty object onto the First Person Player object named "GroundCheck". Move this object to the bottom of the player object.Add any terrain and the ground of our scene to a layer(s). Change PlayerMovement.cs code to the following (only new lines of code were added, nothing from before was edited):

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

public class PlayerMovement : MonoBehaviour
{
    public CharacterController controller;
    //object to be moved using arrow keys or WASD
    public float speed = 12f;
    //public speed variable (can be change in unity IDE)

    public Transform groundCheck;
    //Ground Check object (empty object placed at the bottom of player object)
    public float groundDistance = 0.4f;
    //AOE of our ground collision
    public LayerMask groundMask;
    //Objects the player can stand on (edit in IDE)

    public float gravity = -9.81f;
    //public variable for gravity.
    bool isGrounded;
    //check to see if player is touching ground to reset y velocity back to 0

    Vector3 velocity;

    // Update is called once per frame
    void Update()
    {
        isGrounded = Physics.CheckSphere(groundCheck.position, groundDistance, groundMask);
        //Checks if anything around our groundcheck, given its current position and the distance declared above, is touching our groundMask

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

        float x = Input.GetAxis("Horizontal");
        float z = Input.GetAxis("Vertical");
        //Grabs keyboard/controller input (Values expected: -1, 0 , 1)

        Vector move = tranform.right * x + transofrm.forward * z;
        //Creates a simple vector depending on the value of the keyboard input

        controller.Move(move * speed * time.deltaTime);
        //Applying the move vector to the player object

        velocity.y += gravity * Time.deltaTime;
        //incrementing gravity
        controller.Move(velocity * Time.deltaTime);
        //incrementing velocity
    }
}
```

Set your Ground Mask under Player Movement script to the layers of your terrain and ground.

PART 2: JUMP

Simply add the following lines of code. Edited lines are 18-19 and 49-54:

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class PlayerMovement : MonoBehaviour
6   {
7       public CharacterController controller;
8       //object to be moved using arrow keys or WASD
9       public float speed = 12f;
10      //public speed variable (can be change in unity IDE)
11
12      public Transform groundCheck;
13      //Ground Check object (empty object placed at the bottom of player object)
14      public float groundDistance = 0.4f;
15      //AOE of our ground collision
16      public LayerMask groundMask;
17      //Objects the player can stand on (edit in IDE)
18      public float jumpHeight = 3f;
19      //Jump height
20
21      public float gravity = -9.81f;
22      //public variable for gravity.
23      bool isGrounded;
24      //check to see if player is touching ground to reset y velocity back to 0
25
26      Vector3 velocity;
27
28      // Update is called once per frame
29      void Update()
30      {
31          isGrounded = Physics.CheckSphere(groundCheck.position, groundDistance, groundMask);
32          //Checks if anything around our groundcheck, given its current position and the distance declared above, is touching our groundMask
33
34          if(isGrounded && velocity.y < 0)
35          {
36              velocity.y = -2f;
37          }
38
39          float x = Input.GetAxis("Horizontal");
40          float z = Input.GetAxis("Vertical");
41          //Grabs keyboard/controller input (Values expected: -1, 0 , 1)
42
43          Vector move = tranform.right * x + transofrm.forward * z;
44          //Creates a simple vector depending on the value of the keyboard input
45
46          controller.Move(move * speed * time.deltaTime);
47          //Applying the move vector to the player object
48
49          if(Input.GetButtonDown("Jump") && isGrounded)
50              //A check for KB input and if the character is grounded or not
51          {
52              velocity.y = Mathf.Sqrt(jumpHeight * -2f * gravity);
53              //Maths to figure out how to change the y velocity to simulate a jump
54          }
55
56
57          velocity.y += gravity * Time.deltaTime;
58          //incrementing gravity
59          controller.Move(velocity * Time.deltaTime);
60          //incrementing velocity
61      }
62  }
63
```