Tutorial 2: 3D Player Movement

Preparation:

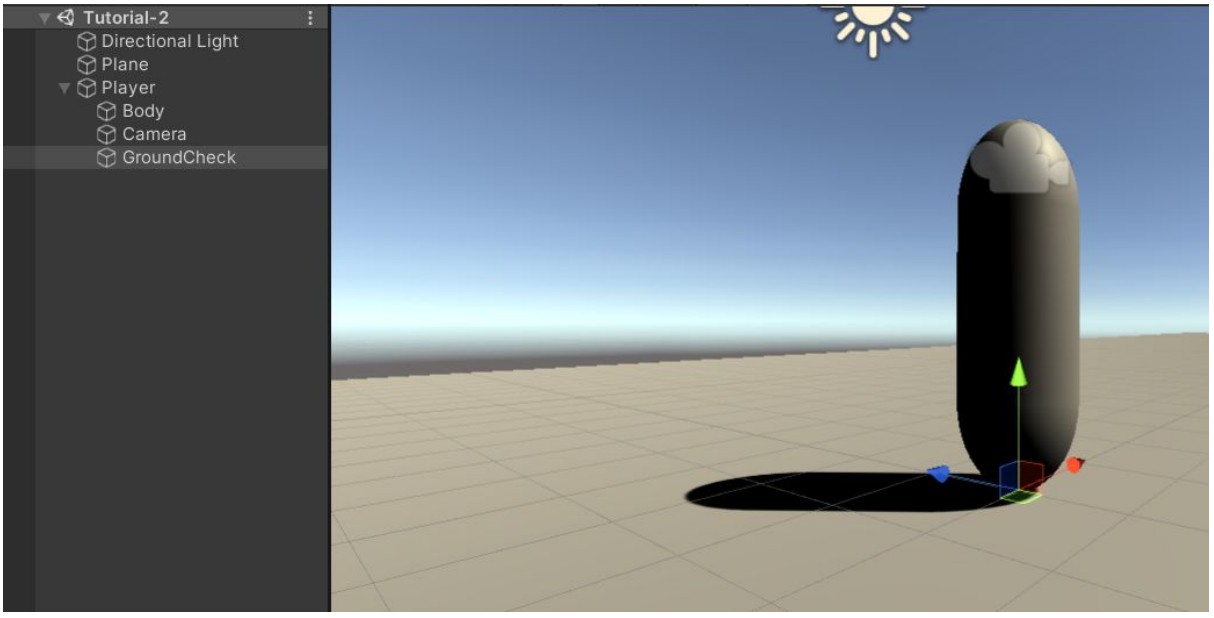
For making a 3D Player Movement that includes jumping we will need:

- An object that represents the player with a character controller
- A ground

You can check Tutorial 1 to see this step in more detail.

Code:

First of all, we need to set all the variables that we are going to need. So let's start with the variables for the ground collision check. Before writing anything, create an empty GameObject inside the Player and situate it in the bottom of the object that represents the Player. It will serve as the ground checker.



Now, we will need to connect the position of the object object (transform) to the script, so we will need to write:

```
public Transform groundCheck;
```

We also need the maximum distance at which the object needs to be to check the collision, a LayerMask variable to identify the ground and a bool to check if there is a collision.

```
public float groundDistance = 0.4f;
public LayerMask groundMask;
private bool isGrounded;
```

Then we need to connect the Character Controller with the script and the speed of movement.

```
public CharacterController controller;
public float speed = 12f;
```

For the jump, we only need to set the height.

```
public float jumpHeight = 3f;
```

And for the gravity we need the proper value of the acceleration and a variable for the velocity.

```
public float gravity = -9.81f;
private Vector3 velocity;
```

The variables should look something like this:

```
//Ground collision check variables
public Transform groundCheck;
public float groundDistance = 0.4f;
public LayerMask groundMask;
private bool isGrounded;

//Movement variables
public CharacterController controller;
public float speed = 12f;

//Jump variables
public float jumpHeight = 3f;

//Gravity variables
public float gravity = -9.81f;
private Vector3 velocity;
```

Next, we need the code inside the function Update.

To avoid problems, we need to start with the ground checker, for that, we will create a sphere around the empty GameObject that we created earlier that is at the bottom of the Player. This sphere will check if it's touching any object with the groundMask, and turn the answer into a bool.

```
isGrounded = Physics.CheckSphere(groundCheck.position, groundDistance, groundMask);
```

Then, to avoid more problems later on, we need to reset the velocity everytime the Player touches the ground.

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

After that, we can continue with the normal movement, assigning new variables to the input of the horizontal and vertical axis and then creating a Vector3 with those variables to use with the character controller.

```
float x = Input.GetAxis("Horizontal");
float z = Input.GetAxis("Vertical");
Vector3 movement = transform.right * x + transform.forward * z;
controller.Move(movement * speed * Time.deltaTime);
```

For the jump we only need to check the input of the “Jump” button and if the Player is grounded. If both are true, we’ll need to add velocity to the Player according to the velocity formula.

```
if (Input.GetButtonDown("Jump") && isGrounded)
{
  velocity.y += Mathf.Sqrt(jumpHeight * -2f * gravity);
}
```

And then, after all those calculus, we need to apply the gravity using the controller.

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

The final code inside Update should look like this:

```
// Update is called once per frame
void Update()
{
    //Ground collision check
    isGrounded = Physics.CheckSphere(groundCheck.position, groundDistance, groundMask);
    if (isGrounded && velocity.y < 0)
    {
        velocity.y = -2f;
    }

    //Movement
    float x = Input.GetAxis("Horizontal");
    float z = Input.GetAxis("Vertical");
    Vector3 movement = transform.right * x + transform.forward * z;
    controller.Move(movement * speed * Time.deltaTime);

    //Jump
    if (Input.GetButtonDown("Jump") && isGrounded)
    {
        velocity.y += Mathf.Sqrt(jumpHeight * -2f * gravity);
    }

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

After writing the code, make sure to have all the necessary elements connected to the script.