Tutorial 2 – FPS Game – Jump!

This tutorial will show how to create a jump movement.

1. Open the scene created in Tutorial 1

We will be utilizing the previous script created in the aforementioned tutorial.

Ensure that you have a Rigidbody component on your player. If you haven't you can do this by selecting Add Component in the Inspector tab and select Rigidbody.

2. Editing the Script

Open the script called FPSController.

Two new floats need to be created in order to use later. Create the new float `jumpSpeed` in order to utilize for the jump only, as we want this to be different to the normal speed used for movement. Create the second one and call it `distToGround` and set to 0.5f in order to ensure the player is on the ground perfectly.

Finally, create a `Rigidbody` called `rb`. This will be used later when we play around with the velocity.

```
public class FPSController : MonoBehaviour {

      public float speed = 2f;
      public float sensitivity = 2f;
      CharacterController player;

      public GameObject eyes;

      float moveLR;
      float moveFB;

      float rotX;
      float rotY;


      public float jumpSpeed = 10f;

      public float distToGround = 0.5f; //

      Rigidbody rb; //used to call upon the rigidbody
```

Similarly, to the last tutorial, we need to store the `Rigidbody` as a variable, allowing us to change and manipulate that component.

```
void Start () {

      player = GetComponent<CharacterController> ();
      rb = GetComponent<Rigidbody> ();

`
```

Now moving into `Update`, create an if statement to allow the character to move along the y axis when the `space` bar is pressed, adding in the addition of `&&isGrounded` to ensure both requirements are met in order to jump only when the player is on the ground. In this if statement create a new `Vector3` called `jumpVelocity` so you will be able to change the movement of the y axis to the `jumpSpeed`, and in turn use this to make a new velocity for the `Rigidbody`.

By adding the new velocity to the standard velocity, the number will be greater therefore the character will move upward in the air against the gravitational pull.

Next, we will track if the player is grounded by using a `Raycast`, which casts a ray from the center of the box to see if it is touching another object. To create the variable `isGrounded`, we must attribute the origin point to `transform.Position`, then set the downwards cast direction using `Vector3` down. Finally, implement the `distToGround` to specify the maxDistance.

```
Debug.Log (isGrounded()); //Shows if the isGrounded is true or false in log.

if (Input.GetKey(KeyCode.Space)&&isGrounded()) {

      Vector3 jumpVelocity = new Vecotr3 (0f, jumpSpeed, 0f);
      rb.velocity = rb.velocity + jumpVelocity;

}

bool isGrounded() {
      return Physics.Raycast (transform.position, Vector3.down, distToGround);

}
```

3. Back to Unity

Now press play and jump away!