Game Design & Development

Game Programming (AME\_5\_GPG\_1920)

CW1\_25

Bobby Edu 3709393

Four Component Tutorials

Character Movement & First-Person Camera (Mouse)

This component for character movement constructs: Movement, camera movement and slope and steps mobility (this means the script allows it to climb steps and slopes).

A screenshot of a cell phone

Description automatically generated

In order to reconstruct this, follow these steps;

1. Create a game 3D game object (any 3D object your choice, you can name it whatever you choose)

2. Add the component called “character controller”

A screenshot of a cell phone

Description automatically generated3. You can configure the height, radius, centre etc, but I recommend you use the same settings I selected

The character the controller also acts as a collider.

A screenshot of a cell phone

Description automatically generated4. Now go to the hierarchy, right click and insert a camera. Only do this if you don’t have a camera in your unity scene, if you already a ‘main camera’ drag it under your 3D game object so it becomes its parent, this means the main camera is now located in the 3D game object.

4b. On the main camera, reset the transform and drag the camera just around the top of the object, this is done so the camera doesn’t clip through when the object jumps e.g. a ceiling in game.

A picture containing green

Description automatically generated

5. Now to start including scripts and logic. On the main camera, create a new component/scripts called “MouseLook” (you can call it anything you want, just along the lines of camera movement so you’re organised) double click it so it opens the visual studio.

A screenshot of a social media post

Description automatically generated6.

6B. This code manages the mouse sensitivity/speed. “100f” is the speed, you can change this to your preference.

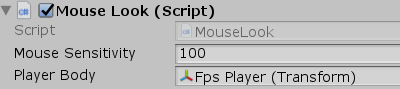
6D. Type “public Transform player ody;” This references the main camera to the whole 3D object.

6C. Now type in “\* mouseSensitivity \* Time.deltaTime;” in both the Y & X line of code. This allows the camera to be rotated around the entire body of the object.

6A. Type “float mouseX = Input.GetAxis("Mouse X") & float mouseY = Input.GetAxis("Mouse Y")” this gathers information about mouse movement.

6E. Type in “playerBody.Rotate(Vector3.up \* mouseX);” after this, go back to unity and link up the 3D game object to the script. Doing this enables camera movement for the X axis.

6F. Type in the code “float xRotation = 0f;” followed by “xRotation -= mouseY;” in then, “transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);”, after add “xRotation = Mathf.Clamp(xRotation, -90f, 90f);”, this allows the camera to rotate but never rotate and look behind the player (3D game object).



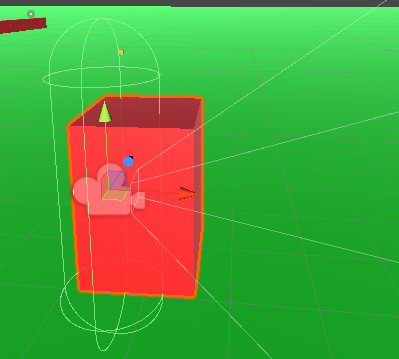
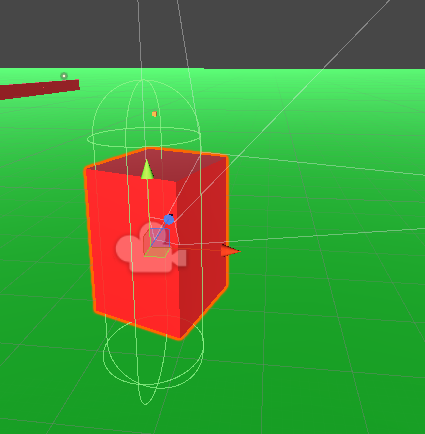
void Start ()

    {

        Cursor.lockState = CursorLockMode.Locked;

    }

6G. Now add the single line of code “Cursor.lockState = CursorLockMode.Locked;” in the void start menu, this will render the cursor hidden so when looking around in game the cursor won’t be visible. Now go back to unity (make sure you save the script!) press play, you will now see that the camera is able to move around the X & Y axis in game!



7. Now for Movement, add a new script on the player/3D game object. Under the void update menu in the script, type in “float x = Input.GetAxis("Horizontal");” then “float z = Input.GetAxis("Vertical");” to gain input.

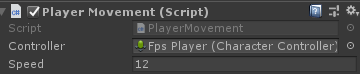
float x = Input.GetAxis("Horizontal");

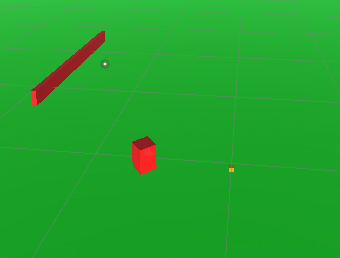
float z = Input.GetAxis("Vertical");

8. For movement directions type “Vector3 move = transform.right \* x + transform.forward \* z;”, in order to actually move on command you will need a reference to the character controller, so type “public CharacterController controller;” under public class menu, it would be ideal to control the speed of movement so type “public float speed = 12f;” under the CharacterController code. “12f” speed is the default speed; you can modify it.

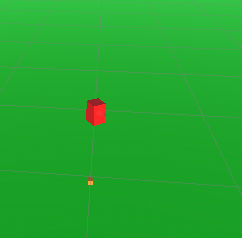
public CharacterController controller;

public float speed = 12f;

9. To finish things on the basic movement type “controller.Move(move \* speed \* Time.deltaTime);” under the Vector3 line of code. When you go back to unity you will now see reference the character controller

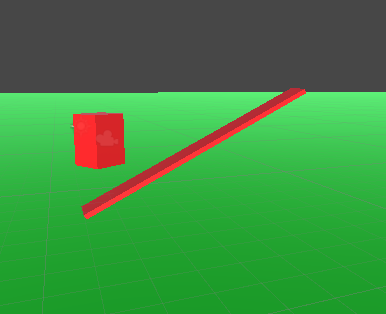
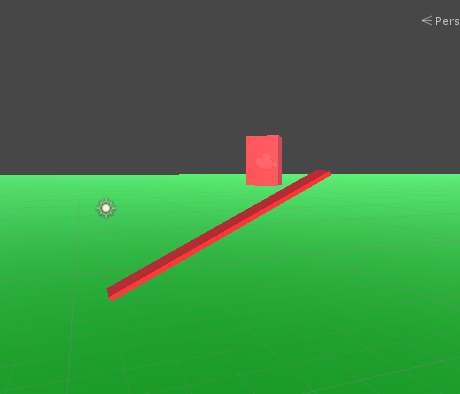


As you can see the 3D game object is able to move able the scene. Controls are automatically binded to the WASD keys.



10. Now to add steps and slope mobility, this is pretty fast, all you have to do is change the “Step Offset” in the Character Controller Menu from 0.3 to 0.7. this is the final step!



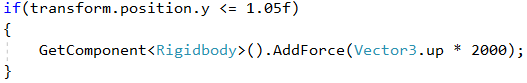


The game object can now climb up steps and slopes

Jump

This component enables a jump action which can be integrated to countless unity3d projects. These are the steps to recreate;

1. Under the update functions, type “if(Input.GetKeyDown(KeyCode.Space))” this assigns the spacebar key to the action (you can change this), so pressing spacebar in game with cause your player to jump.
2. For finishing touches continue with “GetComponent<Rigidbody>().AddForce(Vector3.up \* 2000);”and then “if(transform.position.y <= 1.05f)” this code modifies the speed/force of how high you jump, if you want to make changes, only alter the numerical portion of the code.



2A. Make sure you add the “RigidBody” component to your player, the action will only function because of this.

A screenshot of a cell phone

Description automatically generated

Now if you go back to your project and test it, the player should now elevate upwards and drop (just like a jump motion) when pressing the spacebar key!

A picture containing sky, grass, outdoor

Description automatically generatedA close up of a green field

Description automatically generated

After pressing

Before pressing

Crouching

This final component allows a 3D object to perform a crouch action, this is commonly featured in FPS projects & games, coding this is quite simple;

1. Create a new script and call it “Crouch”, then assign it to your game object.



1. Now under “public class” in the script, type in “CharacterController characterCollider;”, then in the start menu type “characterCollidergameObject.GetComponent<CharacterController>();”, this will enable access to the character controller in your game object.

A screenshot of a cell phone

Description automatically generated

2A. Height can be modified; you can use this to visualize the game object when crouched e.g. half of the height = crouch height.

1. Next in the update function type “if (Input.GetKey(KeyCode.C))” this assigns the key “C” to the action which is crouch (using the key C is optional, you can change this for preference).
2. Now follow it up with “characterCollider.height = 1.8f;” this sets the crouch height, the numerical number represents the height (you don’t need to have the same, it can be changed by altering the number), then add “else { characterCollider.height = 3.8f;” when the “C” key isn’t pressed the height will return back to normal! (this can also be modified) the crouching component is now complete.

A picture containing grass

Description automatically generatedA picture containing grass, green

Description automatically generated

Crouching

Standing

A screenshot of a social media post

Description automatically generatedFinal Script should look similar to this;