# How to create a simple movement script for Unity 3D

Step 1 - Create a new scene in unity and name it what ever you wish.

Step 2 – In that scene add a capsule and name it ‘Player’.

Step 3 – On the player object create a script called ‘FPSController’ and once done open the script to be able to edit it.

Step 4 – once we are in the script we need to add some variables, these will be placed below the line ‘ public class FPSController : MonoBehaviour{ ‘

public float speed = 2f;

public float sensitivity = 2f;

CharacterController player;

public float jumpForce = 4f;

public float walkSpeed = 2f;

public float sprintSpeed = 8f;

public float realSpeed;

private bool hasJumped;

float moveFB;

float moveLR;

float rotX;

float rotY;

float vertVelocity;

Step 5 – Next inside the void start we will need to provide the script with what the player is we do this by typing ‘ player = GetComponent<CharacterController> (); ‘ Below this if you do not want the mouse cursor to be visible you can add a simple line which is ‘ Cursor.visible = false;

Step 6 – After this we will be moving on using the variables in the update section, This small chunk of code will be what will move the player object in each direction the Vertical and horizontal are something unity already has inplace and so will allow the WASD and mouse buttons to be used for movement without having to type each individual one out.

moveFB = Input.GetAxis ("Vertical") \* speed;

moveLR = Input.GetAxis ("Horizontal") \* speed;

rotX = Input.GetAxis ("Mouse X") \* sensitivity;

rotY -= Input.GetAxis ("Mouse Y") \* sensitivity;

rotY = Mathf.Clamp (rotY, -60f, 60f);

Vector3 movement = new Vector3 (moveLR, vertVelocity,moveFB);

transform.Rotate (0,rotX,0);

//eyes.transform.Rotate (-rotY, 0, 0);

movement = transform.rotation \* movement;

player.Move (movement \* Time.deltaTime);

if (Input.GetButtonDown("Jump"))

{

hasJumped = true;

}

ApplyGravity();

{

realSpeed = Input.GetKey(KeyCode.LeftShift) ? sprintSpeed : walkSpeed;

}

void FixedUpdate()

{

transform.Translate(realSpeed \* Input.GetAxis("Horizontal") \* Time.deltaTime, 0f, realSpeed \* Input.GetAxis("Vertical") \* Time.deltaTime);

}

Step 7 – When jumping we want the player to fall back to the ground and so we will need a Gravity function. The below section of code will allow gravity to be acted on the object when the player has jumped to bring them back to the ground, it does this smoothly and so it does not instantly drop the player to the ground.

private void ApplyGravity()

{

if (player.isGrounded == true)

{

if (hasJumped == false)

{

vertVelocity = Physics.gravity.y;

}

else

{

vertVelocity = jumpForce;

}

}

else

{

vertVelocity += Physics.gravity.y \* Time.deltaTime;

vertVelocity = Mathf.Clamp(vertVelocity, -50f, jumpForce);

hasJumped = false;

}

}

Once done with that save the script and make sure it is on the object you wish to control. Make sure to add a character controller component to the object for it to work.