

## Controlador de Personaje en Primera Persona

Triana Casal Rafael Alberto

### Introducción:

Este es un controlador de personaje en primera persona hecho en Unity3D, con un código hecho en Visual Studio (Cabe aclarar que la versión que yo ocupo es Unity 2021.3.10f1 y *VisualStudio* 2022). En este documento tendrás una breve explicación acerca del código, assests utilizados y cualquier movimiento que haya hecho dentro del programa.

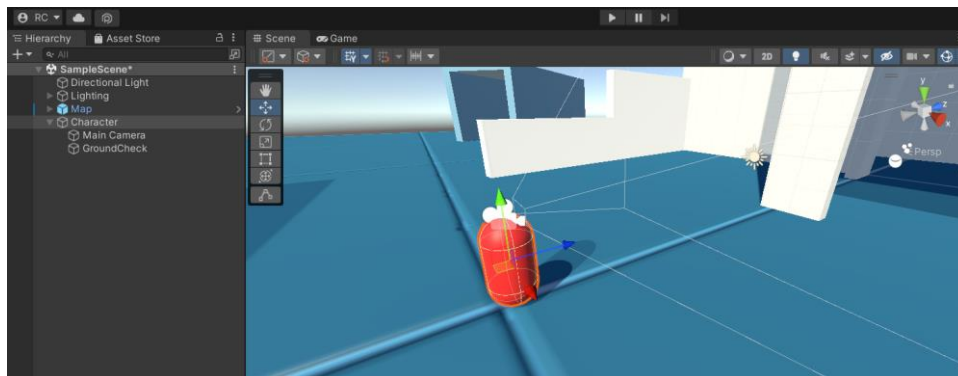
- *Numero 1: Elementos de Escena*

Aquí mostraré todo lo utilizado dentro de la escena y de donde ha sido obtenido.

### Personaje:

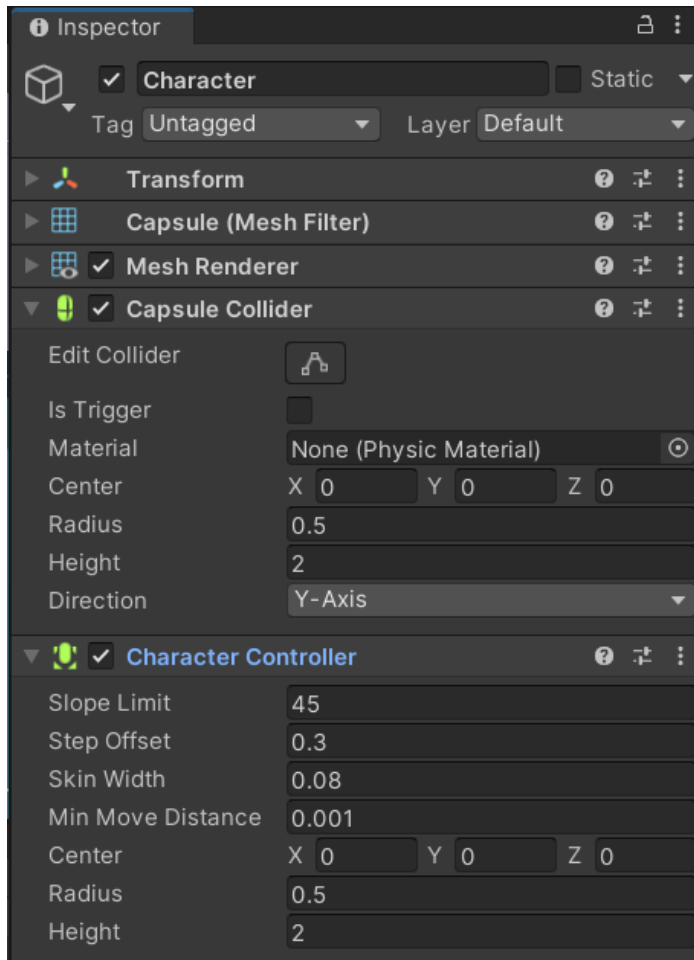
Para el personaje he utilizado una figura 3D básica del mismo Unity; el material utilizado es completamente opcional, sin embargo, si buscas asignarle uno tendrás que hacerlo desde el *click* derecho en *assets*, crear ---> material, para luego asignarlo a donde lo quieres.

De lo primero que se hará será el colocar la *main camera* a donde tu creas propio que serían los ojos de tu personaje



## **Propiedades del personaje:**

A la píldora que ha sido introducida a nuestra escena, se tendrán que agregar los siguientes componentes.



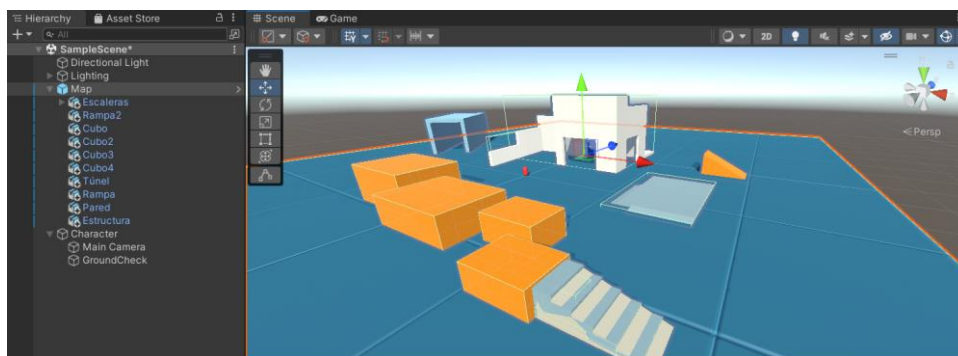
El *Character Controller* será algo que nos ayudará desde un inicio para poder crear esta herramienta que ya es propia de Unity, al igual que la *Capsule Collider*.

## **Mapa:**

Para el mapa he utilizado un paquete gratuito de Unity que se puede descargar directamente desde el apartado de *window* ----> *Asset Store* y te dará la opción de descargarlo para poder utilizar algunas figuras o materiales que Unity no cuenta de base. Sin embargo, se podrá trabajar con todas las opciones que Unity ya nos da de sus objetos bases.



Este es el paquete en específico utilizado, donde solo he descargado la parte de escena, sin código.

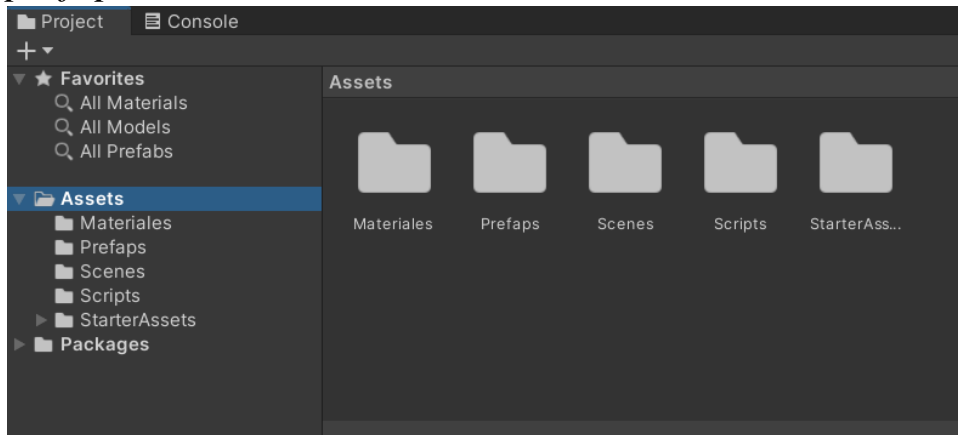


Todo elemento del mapa esta con una *Box Collider* donde he modificado el mismo *collider* a mi gusto. Esto es para poder darles propiedades físicas a los objetos y nuestro personaje no pueda traspasarlos.

- Número 2: Código (Scripts):

Para el código presentaré la segunda versión que he utilizado, la cual ha resultado más funcional, completa y concreta. Hago esta aclaración porque dentro del paquete tendrá tres códigos distintos, pero solo estará activo aquel con el nombre de “*Controller2*”. Sin embargo, dentro del código tendrás agregado los comentarios explicando las líneas de código.

*Como recomendación haz una carpeta específica para cada elemento que utilizarás, una para código, otra para materiales, prefaps, etc.*



## **Movimiento del objeto:**

Para el movimiento del personaje estas serán las variables a utilizar:

```
public class Controller2 : MonoBehaviour
{
    CharacterController characterController;

    //Toda variable utilizada para control del personaje
    [Header("Personaje")] //Esto solo es estética para el script
    public float walkspeed= 10.0f;
    //velocidad de caminar
    public float runspeed= 20.0f;
    //velocidad al correr
    public float jump = 8.0f;
    //impulso de salto
    public float gravity = 20.0f;
    //gravedad con la que se trabajará

    30
    31     private Vector3 move = Vector3.zero;
    32     //Es un Vector3 con los tres ejes en 3, por eso el .zero
    22
```

Como se puede observar cada línea de código tiene su comentario respectivo para la aclaración de funcionalidad del código.

*public float walkspeed= 10.0f;*

*public float runspeed= 20.0f;*

*public float jump = 8.0f;*

```
public float gravity = 20.0f;
```

Las propiedades declaradas a las variables podrán ser al gusto de quien las programe al igual que sus nombres.

- En el apartado de *Start* solo se ocupará lo siguiente:

```
void Start()
{
    characterController = GetComponent<CharacterController>();
    //Esto busca el component de ChracterController
}
```

Esto llamará el componente

```
characterController = GetComponent<CharacterController>();
```

- En el apartado *update* es dónde se hará la mayor parte de la programación para el movimiento de nuestro personaje

```
if (characterController.isGrounded)
{
    move = new Vector3(Input.GetAxis("Horizontal"), 0.0f, Input.GetAxis("Vertical")); //Mov. de derecha a izquierda

    if (Input.GetKey(KeyCode.LeftShift)) //Al cumplir con preionar shift correra la condición de correr
        move = transform.TransformDirection(move) * runspeed;
    else
        move = transform.TransformDirection(move) * walkspeed;

    if (Input.GetKey(KeyCode.Space)) //Salto con tecla Space
        move.y = jump;
}

move.y -= gravity * Time.deltaTime;
characterController.Move(move * Time.deltaTime);
```

Todo este apartado es específicamente para el movimiento del personaje y nada más.

```
if (characterController.isGrounded)
```

```
{
```

```
    move = new Vector3(Input.GetAxis("Horizontal"), 0.0f,
Input.GetAxis("Vertical"));
```

```
    if (Input.GetKey(KeyCode.LeftShift))
```

```
        move = transform.TransformDirection(move) * runspeed;
```

```
    else
```

```

        move = transform.TransformDirection(move) * walkspeed;

        if (Input.GetKey(KeyCode.Space)) //Salto con tecla Space

            move.y = jump;

    }

    move.y -= gravity * Time.deltaTime;

    characterController.Move(move * Time.deltaTime);
    
```

## **Movimiento de cámara:**

Para el movimiento de cámara estas serán las variables a utilizar:

```

19
20 //Variables para la cámara
21 [Header("Cámara")]
22 public Camera cam; //Manda a llamar la cámara que usaremos
23 //Velocidad de mov. Sensibilidad vaya
24 public float mHorizontal = 3.0f;
25 public float mVertical = 2.0f;
26 //Límites de rotación
27 public float minRotation = -65.0f;
28 public float maxRotation = 60.0f;
29 float h_mouse, v_mouse;
    
```

Las propiedades declaradas a las variables podrán ser al gusto de quien las programe al igual que sus nombres.

```

public Camera cam;

public float mHorizontal = 3.0f;

public float mVertical = 2.0f;

public float minRotation = -65.0f;

public float maxRotation = 60.0f;

float h_mouse, v_mouse;
    
```

- Para el apartado de *Start* tenemos la opción de esconder nuestro puntero de la siguiente manera:

```

38 Cursor.lockState = CursorLockMode.Locked;
    
```

Todo lo que se coloca en *start* será activo durante el tiempo que el programa corra.

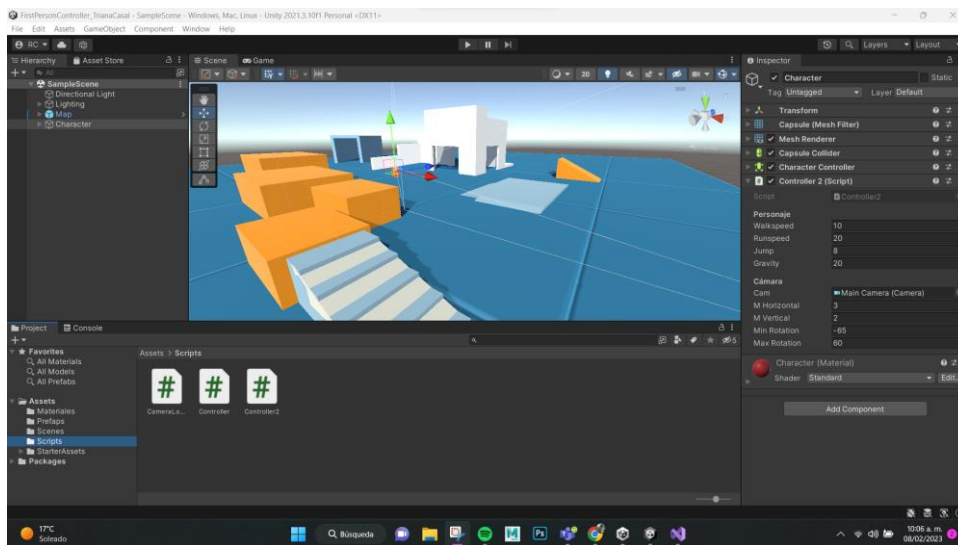
`Cursor.lockState = CursorLockMode.Locked;`

- Para el apartado *update* se hará lo siguiente:



```
h_mouse = mHorizontal * Input.GetAxis("Mouse X");  
v_mouse += mVertical * Input.GetAxis("Mouse Y");  
  
v_mouse = Mathf.Clamp(v_mouse, minRotation, maxRotation);  
//Aquí pondremos un límite para que no rote la cámara de manera infinita  
cam.transform.localEulerAngles = new Vector3(-v_mouse, 0, 0);  
//Esto es para poder rotar la cámara, dejando el vmouse como negativo para control coloquial
```

### Número 3: Asignación

Como último paso a dar se le asignará nuestro código a nuestro personaje, arrastrando nuestro script hacia los componentes de nuestro personaje.



Una vez asignado si has puesto los *Header*, se abrirá un apartado que estará vacío llamado “Cam” en el cual deberás de asignar la *Main Camera* de nuestra *Hierarchy*.

Personaje	
Walkspeed	10
Runspeed	20
Jump	8
Gravity	20
Cámara	
Cam	 Main Camera (Camera) 
M Horizontal	3
M Vertical	2
Min Rotation	-65
Max Rotation	60

Todos estos valores dependerán de los asignados dentro del mismo código.

### **Recomendaciones:**

- Recuerda que muchas veces dependerá de la versión de Unity o Visual Studio que uno utilice.
- Siempre asegúrate de una correcta escritura de los comandos utilizados, variables y valores lógicos.
- Prioriza el orden dentro de tus proyectos, esto para una mejor presentación y una segura optimización del tiempo.
- No olvides cerrar las líneas de código con: `;` / `()` / `{ }` / `[]` respectivamente.
- Recuerda guardar tus progresos siempre que los actualices o hagas un cambio, ya sea desde el menú de archivo o con el *shortcut*: `ctrl + s`, ya sea el código como la escena.