

**MP_0489. Programación
multimedia y dispositivos móviles
UF5. Desarrollo de juegos 2D y 3D**

5.1. Scripting

Índice

☰	Objetivos	3
☰	Crear el juego	4
☰	Crear y mover el jugador	10
☰	Resumen	24

Objetivos

Con esta unidad perseguimos los siguientes objetivos:

1

Aprender a realizar tareas con código en Unity.

2

Conocer el comportamiento básico de las clases.

3

Utilizar correctamente código para manipular *GameObjects*.

¡Ánimo y adelante!

Crear el juego

Gran parte del poder de Unity está en su rico lenguaje de *scripting*, C#. Podemos usarlo para manejar la entrada del usuario, manipular objetos en la escena, detectar colisiones o generar nuevos *GameObjects*.

A lo largo de este apartado **vamos a crear un pequeño juego con el que moveremos una bola en un espacio**. Comenzaremos descargando el siguiente archivo:



Una vez que hayamos abierto el proyecto, veremos que en la vista de *Escena* hay un pequeño espacio, que será el campo para el juego, una cámara y una luz.

Para una primera toma de contacto con el código, prepararemos la cámara para que siga al jugador cuando se mueva.

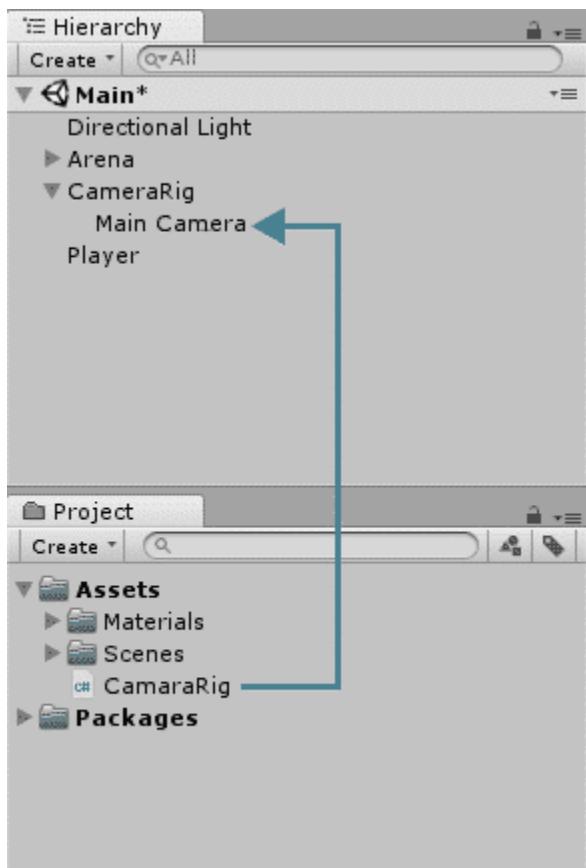
Crear el seguimiento de la cámara

1

Con la carpeta *Scripts* seleccionada, pulsamos el botón *Create* en el panel de *Proyecto* y seleccionamos *C # Script*.

2

Nombramos el nuevo script *CameraRig*. Por último, lo arrastramos sobre el objeto de la cámara principal.



3

Hacemos doble clic en el script *CamaraRig.cs*. Esto abrirá el editor de código con el script cargado. Unity viene con MonoDevelop o Visual Studio, dependiendo de la versión.

Esta es la clase por defecto que Unity genera para los nuevos scripts. Se deriva de la clase base *MonoBehaviour*, que se asegura de que este script se ejecute en el bucle del juego y tenga una funcionalidad adicional para reaccionar en ciertos eventos.

Unity llama a varios métodos en un orden predeterminado mientras se ejecuta el *script*. Estos son los más comunes:

- *Start()*: este método se llamará una vez, justo antes de que el *script* obtenga su primer *update*.
- *Update()*: este método se activará en cada fotograma, mientras el juego se ejecuta y el *script* está habilitado.
- *OnDestroy()*: este método se llama justo antes de que el *GameObject* al que se adjunte se destruya.
- *OnCollisionEnter()*: este método se llama cuando al *collider* o *rigidbody* se le adjunta este *script* para que toque a otro *collider* o *rigidbody*.

4

Creamos las siguientes variables dentro de la nueva clase *CameraRig*, justo encima del método *Start()*: *moveSpeed*, *target* y *rigTransform*.



Las variables son **públicas**, lo que significa que **serán visibles en el panel de Inspector y se podrán ajustar desde él**. Las variables privadas no se podrán establecer a través del panel de *Inspector*.

```
public float moveSpeed;  
public GameObject target;  
  
private Transform rigTransform;
```

Así quedaría el código:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NewBehaviourScript : MonoBehaviour {
    public float moveSpeed;
    public GameObject target;

    private Transform rigTransform;

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
}
```

La variable *moveSpeed* será la velocidad con la que la cámara seguirá al *target*, que puede ser cualquier *GameObject* dentro de la escena.

5

Dentro del metodo *start()* añadimos lo siguiente:

```
rigTransform = this.transform.parent;
```

Así quedaría el código:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NewBehaviourScript : MonoBehaviour {
    public float moveSpeed;
    public GameObject target;

    private Transform rigTransform;

    // Use this for initialization
    void Start () {
        rigTransform = this.transform.parent;
    }

    // Update is called once per frame
    void Update () {

    }
}
```

Este código obtiene una referencia a la transformación del objeto de *Main Camara* en el panel de *Jerarquía*. **Cada objeto en una escena tiene un componente *Transform***, que describe la posición, la rotación y la escala de ese objeto.

6

Ahora, añadimos este código dentro de *start()*:

```
void FixedUpdate()
{
    if (target == null)
    {
        return;
    }

    rigTransform.position = Vector3.Lerp(rigTransform.position, target.transform.position,
        Time.deltaTime * moveSpeed);
}
```

Así quedaría el código:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NewBehaviourScript : MonoBehaviour {
    public float moveSpeed;
    public GameObject target;

    private Transform rigTransform;

    // Use this for initialization
    void Start () {
        rigTransform = this.transform.parent;
    }

    // Update is called once per frame
    void Update () {

    }

    void FixedUpdate()
    {
        if (target == null)
        {
            return;
        }

        rigTransform.position = Vector3.Lerp(rigTransform.position, target.transform.position,
        Time.deltaTime * moveSpeed);
    }
}
```

Vector3.Lerp() toma dos puntos en el espacio y uno flotante en el rango de [0, 1] , que describe un punto a lo largo de los dos puntos finales. El punto final izquierdo es 0, y el punto final derecho es 1. Pasar 0.5 a *Lerp()* devolvería un punto situado exactamente entre ambos puntos finales.



Esto **mueve el *rigTransform*** más cerca de la posición de destino **con un poco de aceleración**. En resumen: **la cámara va a seguir al jugador.**

Crear y mover el jugador

Y ahora, vamos a crear nuestro jugador para que la cámara pueda seguirlo.

Además, **escribiremos algunas líneas más complejas** para poder moverlo.

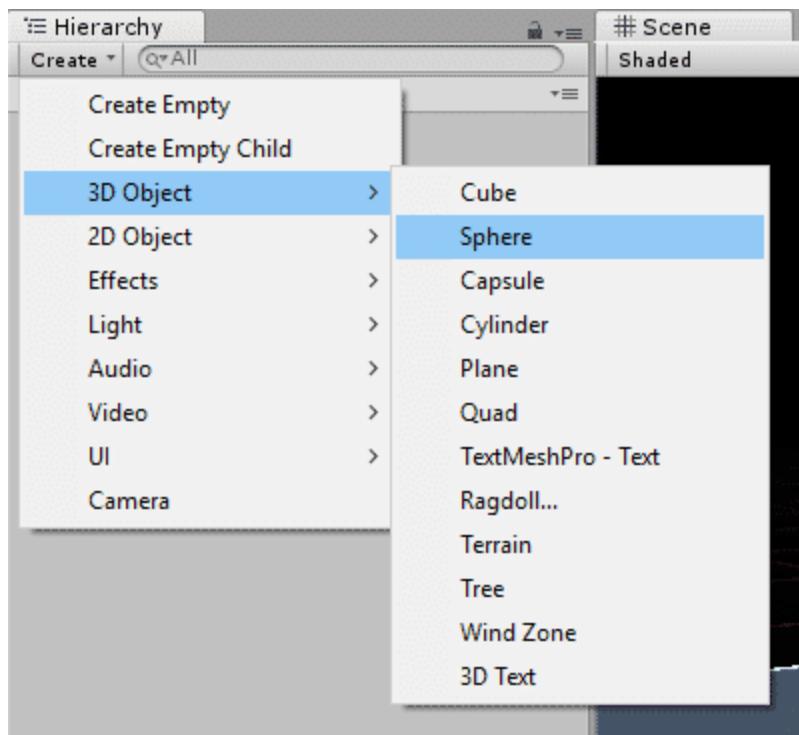
Crear nuestro jugador

1

En el panel de *Jerarquía*, pulsamos el botón *Create* y seleccionamos *Sphere* de la sección 3D.

2

Lo nombramos *Player* y, en el panel de *Inspector*, modificamos la posición con estos valores: (X:0, Y:0.5, Z:0).



Recuerda que Unity usa un sistema de componentes para construir los *GameObjects*. Esto significa que **todos los GameObjects tienen componentes que se pueden adjuntar para darles un comportamiento y propiedades**.

Recordamos **algunos de estos componentes**, como, por ejemplo, los que se muestran en nuestro *Player*:

- **Transform**: todos los *GameObject* vienen con este componente. Mantiene la posición, rotación y escala de un *GameObject*.
- **Sphere Collider**: un colisionador en forma de esfera que puede usarse para detectar colisiones.
- **Mesh Filter**: datos de malla que se utilizan para mostrar un modelo 3D.

Responder a colisiones

El *Player* deberá responder a **colisiones con otros objetos en la escena**.

1

Para que esto suceda, **seleccionamos Player** en el panel de *Jerarquía* y pulsamos en el botón *Add component* en el panel de *Inspector*.

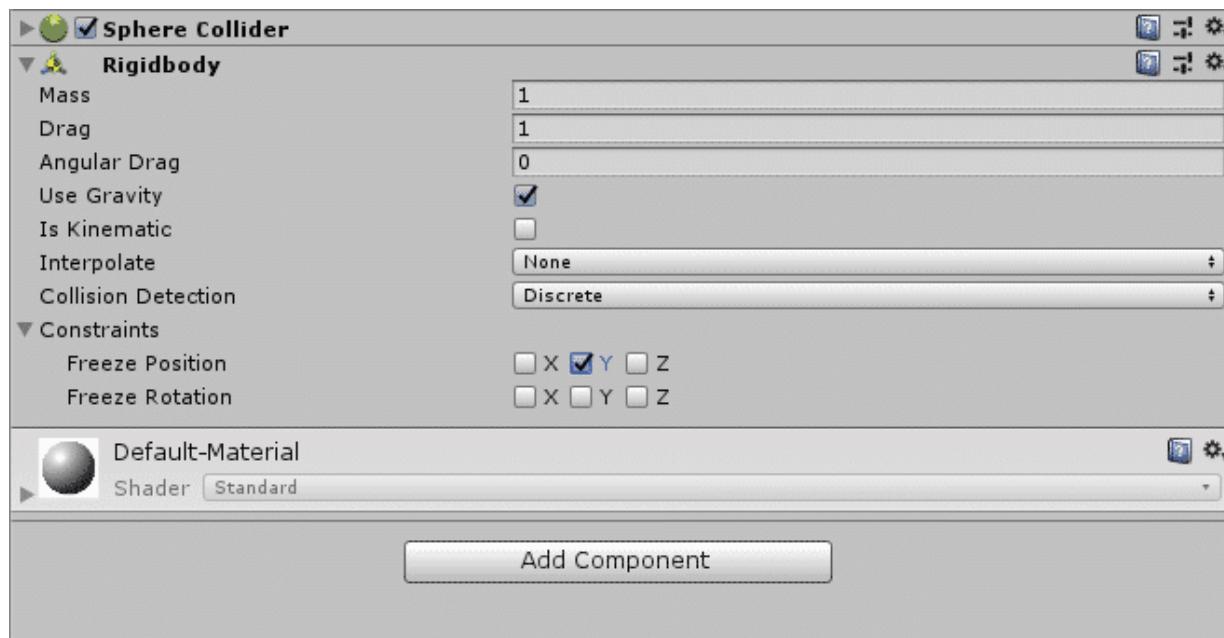
2

Seleccionamos Physics>Rigidbody en el menú emergente; esto nos agregará un componente de *Rigidbody* al *Player* para que podamos utilizar la física de Unity.

3

Vamos a ajustar los valores de *Rigidbody*. Establecemos *Drag* a 1, *Angular Drag* a 0 y marcamos la Y junto a *Freeze Position*.

De este modo, nos aseguramos de que el *Player* no podrá moverse hacia arriba ni hacia abajo, ni tendrá amortiguación al girar.



Crear el movimiento

Ahora que el *Player* está listo, vamos a **crear el código que tomará información del teclado y moverá al jugador.**

1

En el panel de *Proyecto*, hacemos **clic en Create y seleccionamos Folder**. Asignamos el nombre "Scripts" a la nueva carpeta.

2

Dentro de la carpeta "Scripts", hacemos clic en el botón *Create*, seleccionamos C # Script y nombramos al nuevo script *PlayerMovement*.

3

Al hacer **doble clic sobre el script** se nos abrirá en el editor de código.

4

Agregamos estas dos líneas antes del método *Start()*:

```
public float acceleration;  
public float maxSpeed;
```

Así quedará nuestro código:

```
using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
  
public class Player : MonoBehaviour {  
    public float acceleration;  
    public float maxSpeed;  
  
    // Use this for initialization  
    void Start () {  
    }  
  
    // Update is called once per frame  
    void Update () {  
    }  
}
```



- *acceleration* describe cuánto aumenta la velocidad del *Player* con el tiempo.
- *maxSpeed* es el "límite de velocidad".

5

Justo debajo, declararemos las siguientes variables:

```
private Rigidbody rigidBody;  
private KeyCode[] inputKeys;  
private Vector3[] directionsForKeys;
```

Así quedará nuestro código:

```
using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
  
public class Player : MonoBehaviour {  
    public float acceleration;  
    public float maxSpeed;  
  
    private Rigidbody rigidBody;  
    private KeyCode[] inputKeys;  
    private Vector3[] directionsForKeys;  
  
    // Use this for initialization  
    void Start () {  
    }  
  
    // Update is called once per frame  
    void Update () {  
    }  
}
```



- *rigidBody* mantendrá una referencia al componente *Rigidbody* del *Player*.
- *inputKeys* es una matriz de claves para detectar entradas de teclado.
- *Vector3* contiene una matriz de variables *Vector3*, con los datos de movimiento.

6

Ahora, añadimos lo siguiente al método *start()*:

```
void Start () {
    inputKeys = new KeyCode[] { KeyCode.W, KeyCode.A, KeyCode.S, KeyCode.D };
    directionsForKeys = new Vector3[] { Vector3.forward, Vector3.left, Vector3.back, Vector3.right };
    rigidBody = GetComponent<Rigidbody>();
}
```

Así quedará nuestro código:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Player : MonoBehaviour {
    public float acceleration;
    public float maxSpeed;

    private Rigidbody rigidBody;
    private KeyCode[] inputKeys;
    private Vector3[] directionsForKeys;

    // Use this for initialization
    void Start()
    {
        inputKeys = new KeyCode[] { KeyCode.W, KeyCode.A, KeyCode.S, KeyCode.D };
        directionsForKeys = new Vector3[] { Vector3.forward, Vector3.left, Vector3.back, Vector3.right };
        rigidBody = GetComponent<Rigidbody>();
    }

    // Update is called once per frame
    void Update ()
    {
    }
}
```

Este fragmento de código **enlaza las direcciones correspondientes para cada tecla**. Por ejemplo, al presionar `W` se moverá el objeto hacia delante.

La última línea obtiene una referencia al componente *Rigidbody* y lo guarda en la variable *rigidBody* para su uso posterior.

Mover el *player* desde el teclado

Para mover *Player*, tendremos que hacerlo desde el teclado y siguiendo estos pasos.

1

Cambiamos el nombre de *Update()* a *FixedUpdate()* y añadimos el siguiente código:

```
// 1
void FixedUpdate () {
    for (int i = 0; i < inputKeys.Length; i++){
        var key = inputKeys[i];

        // 2
        if(Input.GetKey(key)) {
            // 3
            Vector3 movement = directionsForKeys[i] * acceleration * Time.deltaTime;
        }
    }
}
```

Así quedará nuestro código:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Player : MonoBehaviour {
    public float acceleration;
    public float maxSpeed;

    private Rigidbody rigidBody;
    private KeyCode[] inputKeys;
    private Vector3[] directionsForKeys;

    // Use this for initialization
}
```

```
void Start()
{
    inputKeys = new KeyCode[] { KeyCode.W, KeyCode.A, KeyCode.S, KeyCode.D
};
    directionsForKeys = new Vector3[] { Vector3.forward, Vector3.left,
Vector3.back, Vector3.right };
    rigidBody = GetComponent<Rigidbody>();
}

// Update is called once per frame
// 1
void FixedUpdate()
{
    for (int i = 0; i < inputKeys.Length; i++)
    {
        var key = inputKeys[i];

        // 2
        if (Input.GetKey(key))
        {
            // 3
            Vector3 movement = directionsForKeys[i] * acceleration * Time.-deltaTime;
        }
    }
}
```

Cosas importantes a tener en cuenta:

- **FixedUpdate()** es independiente de la velocidad de fotogramas y debe utilizarse cuando se trabaja con *Rigidbodies*. En lugar de ejecutarse lo más rápido posible, este método se activará en un intervalo constante.
- Este bucle **comprueba si se presionó alguna de las teclas**.
- Obtenemos la tecla presionada, y la multiplicamos por la aceleración y la cantidad de segundos que nos llevó completar el último fotograma. Esto produce un **vector de dirección** (velocidad en los ejes X, Y y Z) que utilizaremos para mover el objeto *Player*.

Velocidad y frecuencia de los fotogramas

Mientras el juego se ejecuta, la frecuencia de fotogramas (o fotogramas por segundo) variará dependiendo del hardware. Esto puede hacer que las cosas sucedan demasiado rápido, en máquinas potentes, y demasiado lento en las más débiles, causando un comportamiento no deseado.



La regla general es que cuando realizamos una acción en cada fotograma (fijo) debemos multiplicar por *Time.deltaTime*.

1

Agregamos el siguiente método a continuación de *FixedUpdate()*:

```
void movePlayer(Vector3 movement) {
    if(rigidBody.velocity.magnitude * acceleration > maxSpeed) {
        rigidBody.AddForce(movement * -1);
    } else {
        rigidBody.AddForce(movement);
    }
}
```

Así quedará nuestro código:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Player : MonoBehaviour {
    public float acceleration;
    public float maxSpeed;

    private Rigidbody rigidBody;
    private KeyCode[] inputKeys;
    private Vector3[] directionsForKeys;
```

```
// Use this for initialization
void Start()
{
    inputKeys = new KeyCode[] { KeyCode.W, KeyCode.A, KeyCode.S, KeyCode.D };
    directionsForKeys = new Vector3[] { Vector3.forward, Vector3.left,
Vector3.back, Vector3.right };
    rigidBody = GetComponent<Rigidbody>();
}

// Update is called once per frame
// 1
void FixedUpdate()
{
    for (int i = 0; i < inputKeys.Length; i++)
    {
        var key = inputKeys[i];

        // 2
        if (Input.GetKey(key))
        {
            // 3
            Vector3 movement = directionsForKeys[i] * acceleration *
Time.deltaTime;
        }
    }
}

void movePlayer(Vector3 movement)
{
    if (rigidBody.velocity.magnitude * acceleration > maxSpeed)
    {
        rigidBody.AddForce(movement * -1);
    }
    else
    {
        rigidBody.AddForce(movement);
    }
}

}
```

El método anterior aplica fuerza a *Rigidbody*, haciendo que se mueva. Si la velocidad actual excede la velocidad *maxSpeed*, la fuerza va en la dirección opuesta para ralentizar al *Player*, y así limitar su velocidad máxima.

2

En *FixedUpdate()* agregamos la siguiente línea:

```
movePlayer(movement);
```

Así quedará nuestro código:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Player : MonoBehaviour {
    public float acceleration;
    public float maxSpeed;

    private Rigidbody rigidBody;
    private KeyCode[] inputKeys;
    private Vector3[] directionsForKeys;

    // Use this for initialization
    void Start()
    {
        inputKeys = new KeyCode[] { KeyCode.W, KeyCode.A, KeyCode.S, KeyCode.D };
        directionsForKeys = new Vector3[] { Vector3.forward, Vector3.left, Vector3.back, Vector3.right };
        rigidBody = GetComponent<Rigidbody>();
    }

    // Update is called once per frame
    // 1
    void FixedUpdate()
    {
        for (int i = 0; i < inputKeys.Length; i++)
        {
            var key = inputKeys[i];

            // 2
            if (Input.GetKey(key))
            {
                // 3
                Vector3 movement = directionsForKeys[i] * acceleration * Time.deltaTime;
                movePlayer(movement);
            }
        }
    }

    void movePlayer(Vector3 movement)
```

```
{  
    if (rigidBody.velocity.magnitude * acceleration > maxSpeed)  
    {  
        rigidBody.AddForce(movement * -1);  
    }  
    else  
    {  
        rigidBody.AddForce(movement);  
    }  
  
}  
  
}
```

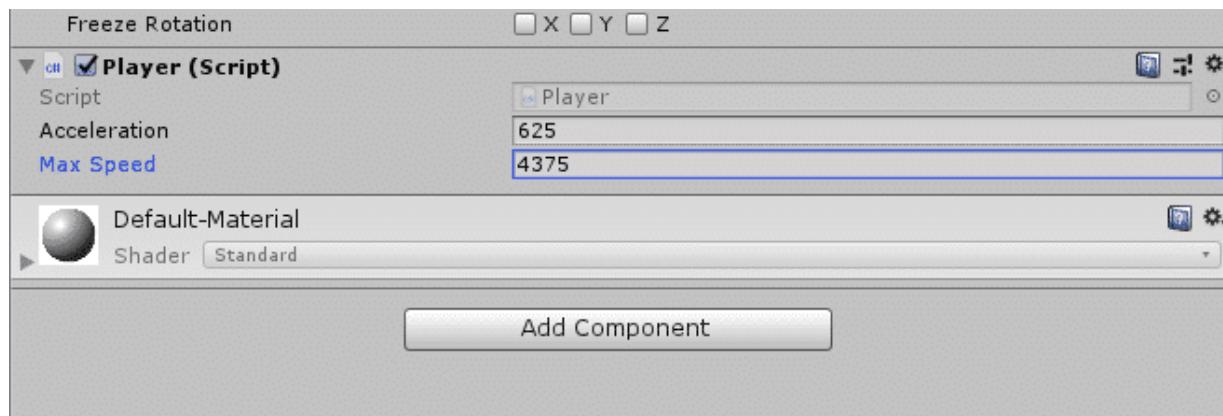
3

Guardamos el *script* y volvemos al editor de Unity. En el panel de *Proyecto*, arrastramos el *script PlayerMovement* al *Player*, dentro del panel de *Jerarquía*.

Al agregar el *script* al *GameObject* se crea un componente, lo que significa que se ejecutará todo el código en el *GameObject* al que se adjunta.

4

En el panel de *Inspector* establecemos la *Acceleration* en 625 y la *Max Speed* en 4375:



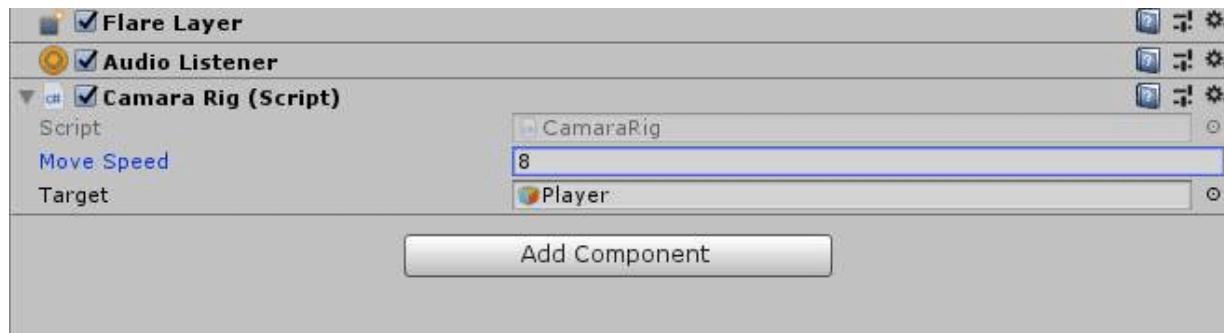
5

Ejecutamos el juego y ya podemos mover el *Player* con las teclas WASD.

Pero puede suceder que en algún momento nos salgamos de plano. En ese caso utilizaremos el *script* que habíamos creado anteriormente.

6

En el panel de *Jerarquía* seleccionamos la *Main Camera* y en el panel de *Inspector*, establecemos la *Move Speed* en 8 y el *Target* en *Player*.



Resumen

Hemos terminado la lección, repasemos los puntos más importantes que hemos tratado.

- A lo largo de esta unidad hemos conocido y aprendido a trabajar con el **editor de código de Unity**.
- También hemos aprendido a **añadir nuevos scripts** a nuestro proyecto.
- Hemos visto **cómo podemos mover la cámara mediante código**, y **cómo programarla para que siga a un objeto**.
- Y, finalmente, hemos aprendido a **manejar y mover un GameObject con el teclado**.



PROEDUCA