

11. Scripts

Programación de scripts

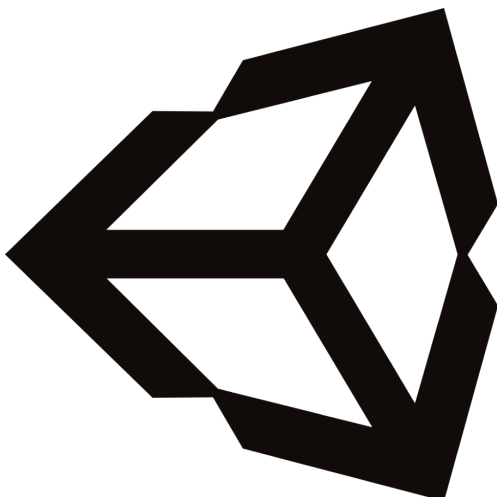
Los scripts son pequeños programas que controlan el comportamiento de los objetos y la dinámica del juego en Unity.

Estos se crean utilizando el lenguaje C# y se guardan en archivos con extensión .CS. Para editarlos se utiliza Visual Studio Code, y se guardan dentro de la carpeta "assets". Dentro de esta carpeta, creamos una nueva carpeta llamada "scripts", donde se guardarán todos nuestros programas.

Para crear un script, hacemos clic derecho en la carpeta y elegimos la opción "create C# script".s

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  [UnityScript | 0 references]
6  public class Players : MonoBehaviour
7  {
8      public bool player1;
9      public float speed = 3;
10
11
12
13  [Unity Message | 0 references]
14  void Start()
15  {
16      }
17
18  // Update is called once per frame
19  [Unity Message | 0 references]
20  void Update()
21  {
22      }
23  }
```

Asignar scripts a objetos



unity

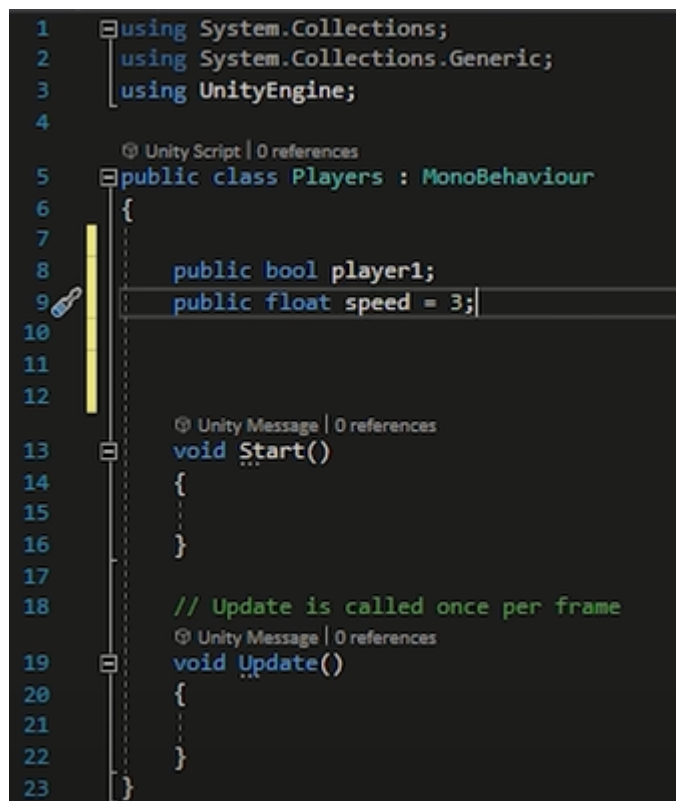
Asignar scripts a objetos

Para que un script tenga efecto, hay que asociarlo a uno o más objetos.

Para ello los scripts se arrastran y sueltan sobre los objetos que queremos que los utilicen.

Cada vez que volvamos a Unity después de modificar nuestros scripts, se recargará el proyecto para incluir los cambios.

En el caso de haber errores de programación, deberemos primero subsanarlos.



```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  @ Unity Script | 0 references
6  public class Players : MonoBehaviour
7  {
8      public bool player1;
9      public float speed = 3;
10
11
12
13  @ Unity Message | 0 references
14  void Start()
15  {
16      }
17
18  // Update is called once per frame
19  @ Unity Message | 0 references
20  void Update()
21  {
22      }
23 }
```

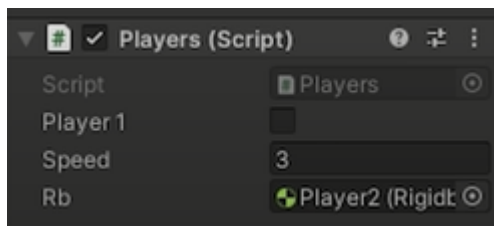
Variables

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Players : MonoBehaviour
6  {
7
8      public bool player1;
9      public float speed = 3;
10
11
12
13      void Start()
14      {
15          //
16      }
17
18      // Update is called once per frame
19      void Update()
20      {
21          //
22      }
23
24 }

```

Las variables creadas son accesibles desde fuera del programa para cambiar parámetros



Estructura de scripts

Los scripts en Unity tienen una estructura básica compuesta por dos partes principales: la parte de declaración de variables, y las funciones.

La parte de **declaración de variables** es donde se definen los campos, variables y propiedades que se usarán en el script.

La segunda parte es la **sección de funciones**, donde se escribe el código que controla el comportamiento de objetos en el juego.

- La función `Start()` se llama al comienzo del juego (una vez) y generalmente se usa para inicializar variables y configurar el estado inicial del objeto.
- La función `Update()` se llama una vez por frame y se usa para actualizar el estado del objeto.

Además de estas dos funciones, podemos crear todas las funciones que queramos para controlar el comportamiento de un objeto, desde eventos de entrada (como cuando un usuario presiona una tecla) hasta eventos de salida (como cuando un objeto sale del juego). Estas funciones se pueden llamar en el script para ejecutar el código deseado.

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Players : MonoBehaviour
6  {
7
8      public bool player1;
9      public float speed = 3;
10
11
12
13  void Start()
14  {
15      }
16
17
18  // Update is called once per frame
19  void Update()
20  {
21
22
23  }

```

Errores en programación

Los errores de programación más comunes en Unity son errores de sintaxis. Estos se producen cuando el programador escribe algo de forma incorrecta, por ejemplo, olvidarse de poner ; al final de las líneas o cerrar un }, o escribiendo mal mayúsculas o minúsculas.

Si hay errores de sintaxis, Unity no podrá ejecutar el juego correctamente, por lo que el programador debe solucionar los errores antes de poder continuar.

Algunas de las formas más comunes de solucionar estos errores son comprobar el código con cuidado, revisar la documentación para asegurarse de que está escribiendo cada línea correctamente

```

[13:02:37] Assets\Scripts\Boat.cs(10,31): error CS1002: ; expected
[13:02:37] Assets\Scripts\Player.cs(457,2): error CS1513: } expected
[13:02:37] Assets\Scripts\Water.cs(5,1): error CS0116: A namespace cannot directly contain members such as fields or methods

```