Unity

Programación de aplicaciones multiplataforma

Raúl Narváez Pérers
2ºDAM, I.E.S. Torre de los Guzmanes, 03/02/2025

Índice

Introducción

Unity es un motor gráfico que incluye su propia interfaz de desarrollo para gestionar y trabajar en la producción de videojuegos. Mediante el uso de estructuras llamadas escenas, podemos gestionar desde escenarios y personajes, hasta pantallas completas y menús que pueden ser interactuables para el usuario final

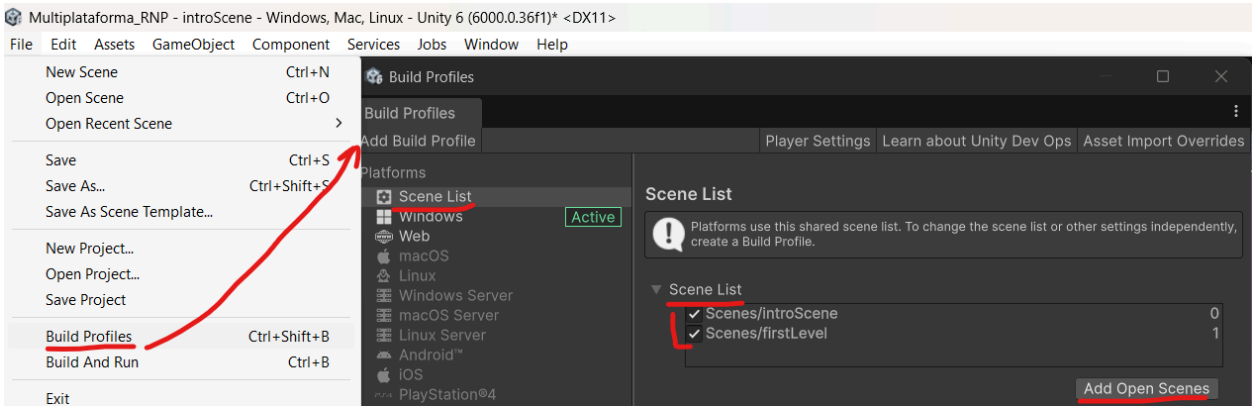
En Unity es frecuente el uso de scripts, que dan funcionalidad y comportamientos personalizados a los objetos y elementos gestionados por el motor. Estos scripts serán una parte importante de nuestro desarrollo, y debemos de tener en cuenta que se codifican en el lenguaje de programación C#.

Temática

La temática del videojuego creado es un plataformas sencillo, con obstáculos, música y un final de fase definido. El desafío es el control de salto, el cálculo de la caída y la evasión de elementos perniciosos para el personaje principal.

Escenas

En nuestro proyecto tenemos dos escenas, una que hospeda la página de inicio y otra en la que se desenvuelven las mecánicas principales. La gestión de escenas es importante para la carga secuencial correcta de las partes del juego. Se pueden listar desde el botón “File” de la barra de herramientas superior, que emitirá un menú desplegable del que deberemos seleccionar la opción “Build profiles”. Esto nos permitirá acceder al menú de perfiles de construcción y al listado de escenas. Podemos añadir escenas a la lista, ordenarlas por secuencia y activar y desactivar las mismas. Este proceso genera un id numérico para cada escena que comienza en 0, esto es muy útil para gestionar sus usos sin necesidad de llamar a las escenas por sus nombres.



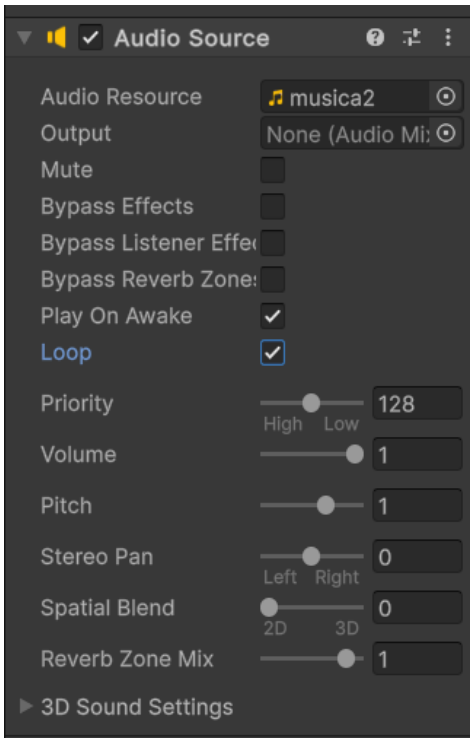
Escena 0: Pantalla de inicio

Tenemos una serie de objetos en el gestor de jerarquía en la barra izquierda. Estos objetos configuran la pantalla de inicio que da paso al juego principal y también permite salir de la aplicación.

Main Camera

Main Camera representa el campo de visión al que el espectador va a tener acceso al iniciar la aplicación. En la barra derecha, o inspector, podemos ver qué elementos se han añadido a la cámara. Entre ellos se han añadido un complemento **Audio Source** que implementa en su atributo Audio Resource un archivo mp3 importado directamente desde el directorio Assets/music. Esto permite que el audio esté vinculado a la cámara, de modo que la música, extradiegética en el ambiente del videojuego, siempre suene de fondo de manera estable e ininterrumpida. La opción Loop está activada para que no encuentre fin la canción por mucho que esperemos en esta escena.

Además de esto tenemos añadido un script llamado Start Script, implementado desde el directorio Assets/scripts. Este script configura las funcionalidades de cargar la siguiente escena en la lista (StartGame()), la número 1, y cerrar el juego (QuitGame()). Estas funciones no van a ser usadas por la cámara pero van a ser importadas por otro objeto de unity que si las usan.




```
0 references
public class StartScript : MonoBehaviour
{
    0 references
    public void StartGame()
    {
        // Carga la escena del juego
        SceneManager.LoadSceneAsync(1);
    }

    0 references
    public void QuitGame()
    {
        // Sale del juego
        Application.Quit();
    }
}
```

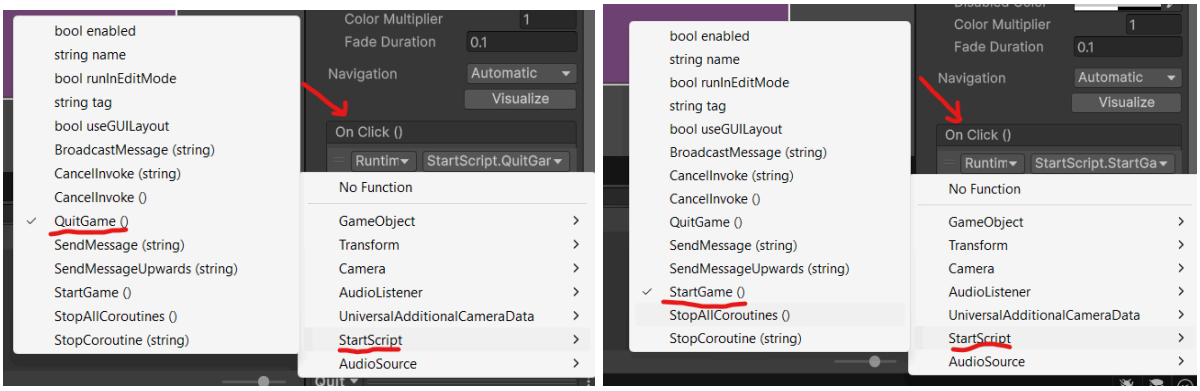
Canvas

Canvas es un tipo de objeto plano en Unity que permite la colocación de elementos visuales sobre él. Es importante que en el inspector seleccionemos que el canvas tenga el **Render mode** en Screen Space para que tome toda la pantalla, que seleccionemos **Pixel perfect** para que se ajuste al píxel de precisión y por último le pasamos como objetos en **Render Camera** el objeto Main Camera. Esto permitirá que el canvas y todos los elementos que contiene estén siempre centrados y enfocados en la visión del usuario. El canvas contiene, en este caso, un elemento **Panel** que a su vez sostiene tres elementos.

- Botón Play: Elemento accionable con características de color, forma y texto modificable. En este objeto tiene un apartado denominado **On click()**, esto implica que ejecuta una acción cuando se pulsa sobre él. Para asignar la acción debemos arrastrar el objeto Main Camera a este atributo y

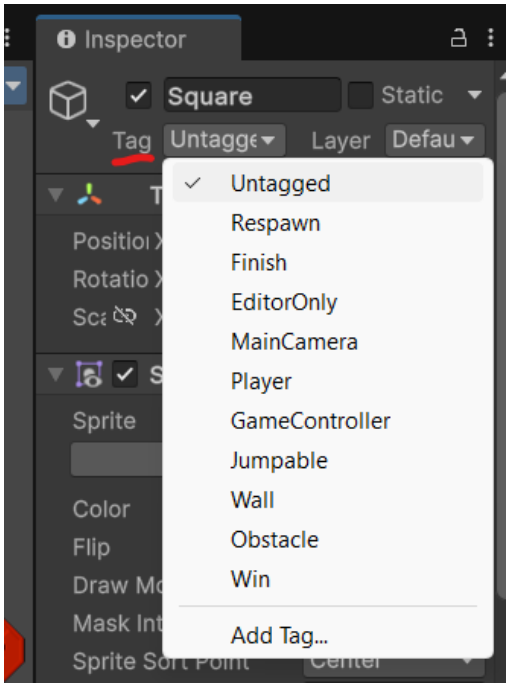
seleccionar el script Start Script y la función StartGame(). A partir de este momento, este botón iniciará la siguiente escena.

- Botón Quit: Exactamente igual que el anterior pero con otro color y texto, así como carga la función QuitGame().
- Text: Un cartel de texto que permite adecuar la fuente, color y tamaño de la letra.



Escena 1: Primer nivel

En esta escena podemos ver una Jerarquía más cargada. Aquí hay más objetos con mayor carga lógica y que representan y ejercen más funcionalidades que simplemente botones estáticos.Es importante, antes de continuar que especifiquemos que son los Tags.

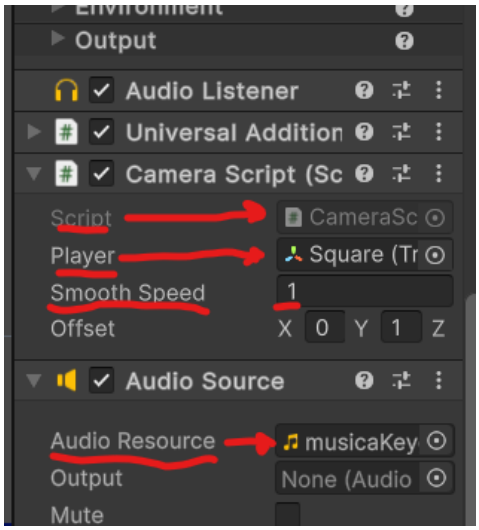


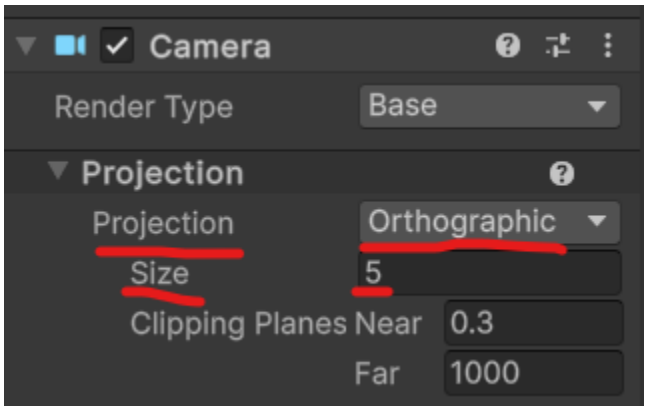
Las Tags son etiquetas, elementos discriminadores que nos permiten generar grupos de objetos que van a compartir un determinado efecto que decidamos designar. Un tag puede aplicarse a elementos y objetos diferentes y de diferentes orígenes. Esto será especialmente útil cuando describamos cómo interaccionan los objetos entre sí en la lógica de los scripts. Se pueden acceder a ellos en la parte superior del inspector de objetos.

Main Camera

Esta cámara es igual que la anterior, con la diferencia de que el script que tiene asociado, así como el audio de la música extradiegética son distintos. El Script define unos valores que establecen un objeto jugador como objetivo, del que sacan los valores de su posición en todo momento y los usan para reorientarse. Esto produce una función de seguimiento de la cámara al jugador principal, pero puede ser bastante áspero en su movimiento, por lo que debemos asignar en el inspector el valor de **Smooth speed** a 1.

También se ha añadido la capacidad de alejar la cámara un 25% a través de un atributo float adicional (1.25f) que se aplica a la propiedad ortográfica size de la cámara en el método LateUpdate() si se pulsa la letra control.





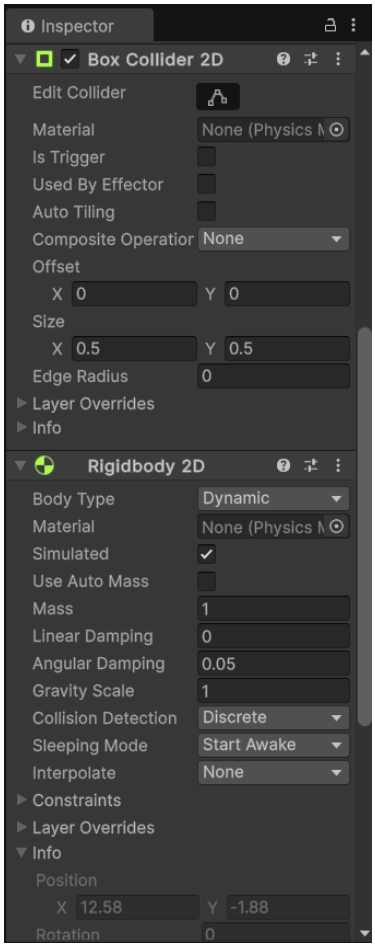
Global light 2D

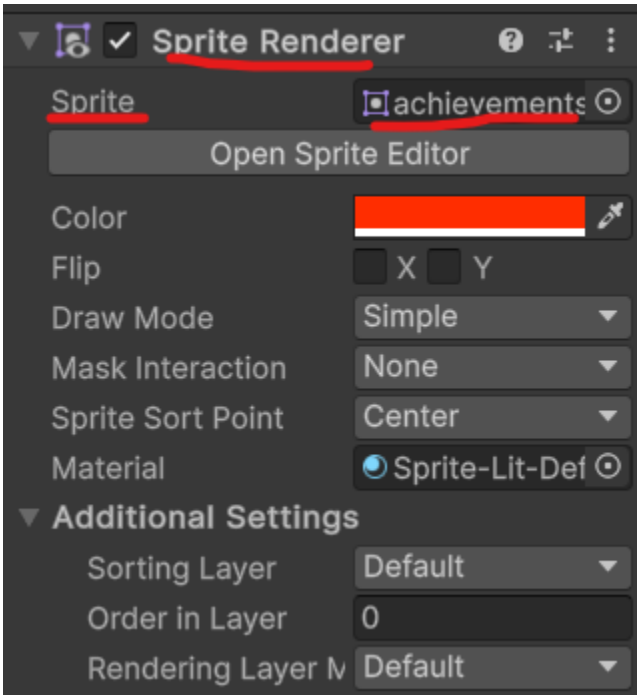
Iluminación general, predeterminada de la escena al generarse. No ha sido modificada.

Square, Goal y Obstacle

Estos tres objetos son diferentes implementaciones del mismo objeto, el objeto 2d Square, pero se les han asignado diferentes tags, diferentes sprites y diferentes tamaños. Sólo el objeto que se comporta como el jugador principal, Square, tiene asignado un Script.

Los tags que reciben los objetos obstacles se les denomina Obstacle y el objeto Goal tiene el tag Win. Todos los objetos mencionados, necesitan un **Box Collider 2D** para implementar las colisiones, el cual puede modificarse desde el inspector especialmente la característica size. También implementan un **RigidBody 2D** que les permiten tener físicas de gravedad, modificables desde el inspector, pero en este caso no han sido alterados. Todos tienen un **Sprite Renderer** al que se les han asignado un sprite desde el directorio Assets/sprites.





El objeto Square, que representa al jugador, tiene además un script que se llama Player Movement. Este script define los atributos básicos que necesita el objeto square para que su rigidBody2D pueda realizar las funciones de saltar, moverse en el salto y escalar paredes.

Dentro de la función que se inicializa automáticamente al iniciar la escena (void start()) se declara la posición inicial del Square y se obtiene como variable el atributo RigidBody2D desde Unity.

Durante la función actualizadora (void update()), que se ejecuta a cada fracción de tiempo, se le asocian funciones de comprobación de input de las teclas direccionales, que ejecutan movimientos en base a los atributos asociados al principio y a otras variables como comprobaciones de si el objeto square está tocando el suelo. Esto se consigue calculando si el collider de Square está en contacto con el collider de cualquier otro objeto que tenga el tag "Ground". De esta manera se puede decidir que si está tocando ground, puede saltar, y si no está tocando ground puede moverse lateralmente. De esta forma definimos un movimiento basado en saltos.

El escalado de paredes por saltos funciona de manera similar, pero calculado por proximidad determinada, muy pequeña, a objetos con el tag "Wall". De esta manera, cuando estamos cerca de un objeto wall podemos saltar verticalmente.

Para calcular que se está cerca de un objeto tageado Wall, se define una bool IsNearWall(), este método se usa dentro de void update() para que se calcule constantemente la proximidad a la pared.

Por último, tenemos OnColissionEnter2D(Collision2D collision) Una función que detecta colisiones entre box colliders y ejecuta según qué lógica dependiendo de los tags de los objetos con los que choque el objeto square:

- Si el objeto tiene el tag Jumpable: resetea la capacidad de poder saltar del objeto, a través de un atributo booleano llamado isGrounded.
- Si el objeto tiene el tag Obstacle: devuelve el objeto square a la posición de inicio que se toma al comenzar la escena. Esto simula el reinicio de nivel al ser destruido por un obstáculo.
- Si el objeto tiene el tag Win: El juego termina lanzando un SceneManager que te envía de nuevo a la escena inicial.

Ground y Wall

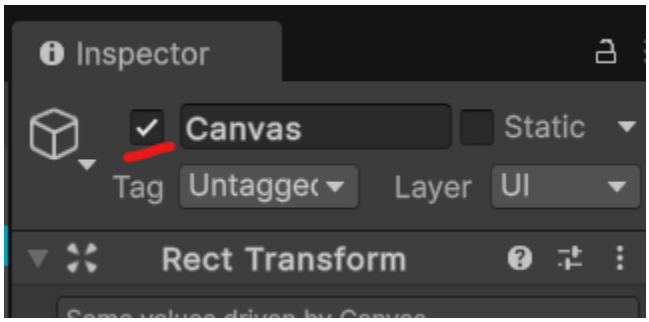
Son objetos sencillos 2D a los que se les ha añadido un BoxCollider 2D y un tag Ground y Wall respectivamente. Sólo su forma y posición han sido alterados.

Canvas

Se ha realizado un objeto Canvas, con un panel, como en la página inicial, pero se ha modificado para que tenga un solo botón y un texto que determina que es el menú de pausa. El botón envía a la página inicial si se pulsa, mediante el mismo método que los botones de la página inicial, cargando una función dentro del script de la cámara que se invoca desde canvas. El canvas, contenido dentro del objeto Main Camera, se ha ajustado como el de la página inicial para que quede centrado con la cámara, pero se le ha puesto una ligera transparencia y está desactivado de base, sólo se activa mediante la tecla espacio. Este comportamiento se ha asignado mediante la adición del componente script al canvas.

El otro canvas es más sencillo, transparente y sólo tiene un objeto Text dentro que se ve manipulado por el Script asociado a él, Puntuación. Este Script genera una serie de valores como el contenido del Text y el valor de la hora actual. Con ellos calcula un número en base a doscientos y pierde valor por segundos transcurridos. Mediante la función void update(), se asigna este valor calculado al valor del texto y

podemos mostrar por pantalla la puntuación. A diferencias del canvas anterior, este siempre está activado y no puede eliminarse mediante pulsaciones de teclas, puesto que no está definido en su script.



Conclusión

Con estos simples pasos y un poquito de programación en C#, se ha realizado un juego simple pero funcional y completo. La utilización de motores prefabricados para la generación de videojuegos puede ser algo compleja, pero aprender las herramientas y estructuras predeterminadas propias del motor te permite en poco tiempo adquirir velocidad y precisión en la creación de videojuegos de una manera rápida y estandarizada.