DAVID CANTÓN NADALES

CURSO PRÁCTICO CON UNITY 3D

ANAYA MULTIMEDIA

ÍNDICE DE CONTENIDOS

2. MANIPULACIÓN DE GAMEOBJECTS Y GESTIÓN DE *ASSETS* 37

3. INTRODUCCIÓN A C# Y VISUAL SCRIPTING 61

4. CREACIÓN Y PERSONALIZACIÓN DEL TERRENO Y EL JUGADOR 89

5. COLISIONES Y SALUD 111

6. CREAR UN NPC CON CHATGPT 127

7. DOMINANDO LOS *SHADERS* Y EFECTOS VISUALES AVANZADOS 141

INTRODUCCIÓN

Unity es una de las herramientas más populares y accesibles para el desarrollo de videojuegos, permitiendo a desarrolladores de todos los niveles llevar sus ideas a la realidad. Este libro no solo te enseña los fundamentos, sino que también profundiza en técnicas avanzadas como animaciones, optimización de rendimiento, integración de realidad aumentada y el uso de librerías esenciales para enriquecer tus proyectos. A través de ejercicios prácticos y explicaciones claras, aprenderás a aprovechar al máximo las posibilidades de Unity, tanto si eres nuevo en el desarrollo de videojuegos como si ya tienes experiencia en la creación de proyectos.

Destinatarios de este libro

Este libro está dirigido a personas de todas las edades y niveles de experiencia que quieran aprender a desarrollar videojuegos en Unity. Gracias a la inclusión de Visual Scripting, es una herramienta ideal para niños, adolescentes y principiantes que deseen explorar el mundo del desarrollo de videojuegos sin necesidad de conocimientos previos en programación. También está diseñado para desarrolladores intermedios y avanzados que busquen mejorar sus habilidades y explorar áreas más técnicas como la optimización, la realidad aumentada y el uso de librerías de terceros. Padres, educadores y entusiastas del diseño encontrarán en este libro una guía clara y completa para enseñar o aprender a crear videojuegos de forma efectiva.

Organización del libro

Este libro se divide en 14 capítulos, cada uno centrado en un aspecto esencial y progresivo del desarrollo de videojuegos con Unity. Los primeros capítulos introducen los fundamentos del motor, como su interfaz, la manipulación de objetos y la programación básica en C#, incluyendo Visual Scripting como una alternativa intuitiva para principiantes y niños.

En los capítulos intermedios, se profundiza en áreas clave como la inteligencia artificial con ChatGPT, los efectos visuales avanzados, la integración de sonido y música y el diseño de interfaces de usuario. Además, se abordan animaciones con herramientas como Animator, Cinemachine y Timeline, así como la implementación de realidad aumentada utilizando ARFoundation.

Los capítulos finales están orientados a la optimización y la publicación, con temas como la monitorización del rendimiento, la generación de *builds* y el uso de herramientas y librerías de terceros gratuitas y *premium* para maximizar la calidad y productividad de los proyectos.

Cada capítulo combina teoría con ejercicios prácticos y ejemplos claros, lo que proporciona un enfoque integral para desarrollar videojuegos en Unity, desde los conceptos básicos hasta técnicas avanzadas.

Convenios empleados

En este libro se han empleado los siguientes estilos para facilitar la lectura y comprensión del contenido:

- **Nombres de botones, herramientas y comandos de Unity:** Aparecen en Arial, por ejemplo, el botón Play.
- **Combinaciones de teclas:** Se presentan en negrita, como **Control-S** para guardar.
- **Códigos y *scripts*:** Aparecen en letra monoespacial, como en el ejemplo:

```
Debug.Log("Hola Mundo");
```

Ejemplos del libro

Todos los proyectos y ejemplos utilizados en este libro están disponibles para su descarga en la página web de Anaya Multimedia en `https://anayamultimedia.es`, en la opción Selecciona complemento que encontrará en la ficha correspondiente a este libro. También podrá en encontrarlos en el espacio del autor en Github en la siguiente dirección: `https://github.com/davidcantonnadales/Ejemplos-Anaya-Curso-Practico-Con-Unity`. Podrás acceder a todos los ejemplos para estudiar el código, experimentar con los proyectos o adaptarlos a tus necesidades.

3

INTRODUCCIÓN A C# Y VISUAL SCRIPTING

Introducción

Unity viene equipado con una amplia gama de herramientas que ayudan a resolver problemas comunes en el desarrollo de videojuegos, como hemos visto hasta ahora. No obstante, cada juego tiene características únicas que requieren soluciones específicas que Unity no puede resolver con sus componentes predefinidos. Por esta razón, es necesario utilizar *scripts*. En este capítulo, exploraremos las dos opciones principales de *scripting* en Unity: C# y la programación visual (Visual Scripting). Evaluaremos sus ventajas y desventajas, y discutiremos los conocimientos básicos necesarios para comenzar a desarrollar juegos con ambas herramientas. A partir de aquí, aprenderemos a implementar todos nuestros *scripts* utilizando estas dos opciones; una vez los conozcas, podrás utilizar el sistema que más te guste.

Exploraremos los siguientes conceptos:

- Introducción a la programación con *scripts*.
- Creación de *scripts*.
- Uso de eventos e instrucciones.
- Trucos y consejos de C# para principiantes.

Introducción a la programación con *scripts*

En este capítulo, aprenderemos a desarrollar nuestros propios componentes en Unity, comprendiendo la estructura fundamental de un *script* y cómo ejecutar acciones y exponer propiedades tanto con C# como con la programación visual. En

lugar de desarrollar código de juego real, nos enfocaremos en crear *scripts* de ejemplo que establecerán las bases para el próximo capítulo. Comencemos examinando las opciones de *scripting* en Unity.

Opciones de *scripting* en Unity

Unity nos proporciona herramientas para ampliar sus capacidades mediante la codificación. Una de las formas principales de hacerlo es utilizando C#, un lenguaje de programación versátil y potente. Además de C#, Unity ofrece la opción de programación visual, que permite crear *scripts* utilizando una interfaz gráfica de nodos, como se ve en la figura 3.1.

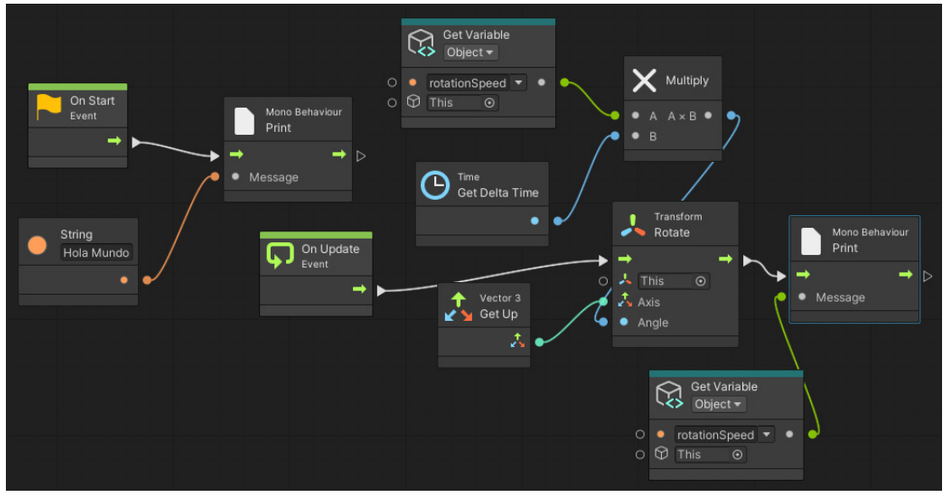

Figura 3.1. Ejemplo de un gráfico de Visual Scripting.

Esto facilita la creación de *scripts* sin necesidad de escribir código, simplemente arrastrando y soltando nodos que representan acciones encadenables.

Ambas opciones pueden lograr resultados similares, pero son adecuadas para diferentes propósitos. Por lo general, la lógica central de un juego se implementa en C# debido a su eficiencia y capacidad de manejar grandes volúmenes de código de manera eficiente. No obstante, los *scripts* visuales son ideales para que los artistas y diseñadores de juegos, que pueden no tener experiencia en programación, realicen ajustes menores y experimenten con el equilibrio del juego o efectos visuales.

Además, la programación visual es perfecta para niños que desean iniciarse en el desarrollo de videojuegos y para personas que no saben programar, ya que permite comprender los conceptos básicos del *scripting* sin necesidad de escribir código complejo.

Un caso práctico es cuando los diseñadores de juegos prototipan ideas usando *scripts* visuales, que luego los programadores convierten en *scripts* de C# una vez aprobadas. Además, los programadores pueden crear nodos personalizados en C# para ser utilizados por los diseñadores en la programación visual.

La combinación de estas herramientas varía entre los equipos de desarrollo. Aunque en los próximos capítulos nos centraremos principalmente en C#, también presentaremos las versiones equivalentes en programación visual de los *scripts* que desarrollaremos. Así, podrás decidir cuándo es más conveniente utilizar una u otra opción, adaptándose a la estructura y necesidades de tu equipo.

Creación de *scripts*

Para empezar a desarrollar un comportamiento específico en Unity, el primer paso es crear recursos de *script*. Estos recursos son archivos que contienen la lógica que define el comportamiento de nuestros componentes. Tanto C# como la programación visual en Unity requieren sus propios tipos de activos para llevar a cabo esta tarea. Vamos a profundizar en cómo crear estos recursos utilizando ambas herramientas.

Este libro supone que ya tienes algún conocimiento previo de programación. Sin embargo, en esta primera sección, abordaremos la estructura básica de un *script* para asegurarnos de que tengas una base sólida antes de avanzar hacia la codificación de comportamientos más complejos en los próximos capítulos. Incluso si ya estás familiarizado con C#, es recomendable no saltarse esta sección, ya que discutiremos las estructuras de código específicas de Unity.

Temas que abordaremos en esta sección:

- Configuración inicial.
- Creación de un *script* en C#.
- Creación de un gráfico de Visual Scripting.

Antes de adentrarnos en la creación de *scripts*, es crucial asegurarnos de que nuestro entorno de desarrollo esté configurado adecuadamente. Este paso preliminar nos garantiza que tendremos todas las herramientas necesarias para escribir, depurar y gestionar nuestros *scripts* de manera eficiente. En este libro, utilizaremos exclusivamente Visual Studio como nuestro entorno de desarrollo integrado (IDE). Visual Studio ofrece características avanzadas como coloreado de sintaxis, autocompletado y capacidades de depuración, lo que facilita enormemente el proceso de desarrollo.

5

COLISIONES Y SALUD

Introducción

La física es un componente clave en los videojuegos cuando se trata de simular comportamientos del mundo real, como el movimiento de objetos y las interacciones al chocar, ya sea entre jugadores y paredes o entre balas y enemigos. Sin embargo, controlar la física puede ser un desafío debido a la multitud de reacciones posibles tras una colisión. Por eso, es crucial aprender a configurar nuestro juego de manera que las colisiones sean lo más precisas posible, permitiendo que el movimiento y las interacciones en el juego se sientan naturales y fluidos. Esto nos ayudará a crear una experiencia de juego dinámica y convincente, donde las reacciones físicas reflejen con precisión la intención del diseño.

Exploraremos los siguientes conceptos:

- Configuración de la física.
- Detección de colisiones.

Primero, nos enfocaremos en configurar correctamente la física, un paso esencial para que nuestros *scripts* puedan detectar colisiones entre objetos, utilizando nuevos eventos de Unity que aprenderemos en el proceso. Esto es fundamental para asegurarnos de que nuestras balas impacten a los enemigos y les causen el daño esperado. Después, exploraremos las diferencias entre mover objetos utilizando Transform, como hemos hecho hasta ahora, y Rigidbody, evaluando los beneficios y limitaciones de cada método. A partir de estos conceptos, experimentaremos con distintas maneras de mover a nuestro jugador, dándote la libertad de elegir la que mejor se adapte a tu estilo de juego. Comencemos con la configuración de la física.

Configuración de la física

El sistema de física de Unity está diseñado para cubrir una amplia gama de aplicaciones dentro del juego, por lo que configurarlo correctamente es esencial para lograr los resultados que buscamos. En esta sección, abordaremos los siguientes aspectos clave de la configuración de la física:

- Ajuste de formas.
- Tipos de objetos físicos.
- Detección de colisiones.

Comenzaremos explorando los distintos tipos de colisionadores que ofrece Unity y cómo configurarlos para capturar diversas reacciones físicas, como colisiones y disparadores. Por último, veremos cómo evitar colisiones no deseadas entre objetos específicos, como cuando las balas del jugador impactan en su propio personaje.

Ajuste de formas

En el capítulo 2, vimos que los objetos suelen tener dos formas: la visual, que es la malla 3D, y la física, que corresponde al colisionador utilizado por el sistema de física para calcular colisiones. La idea es que puedas tener un modelo visual detallado envuelto en una forma física simplificada para mejorar el rendimiento.

Unity proporciona varios tipos de colisionadores, y aquí repasaremos los más comunes, comenzando con los tipos primitivos: Box, Sphere y Capsule. Estas formas son las más eficientes en términos de rendimiento para detectar colisiones, ya que las interacciones entre ellas se calculan mediante fórmulas matemáticas, a diferencia de colisionadores como el Mesh Collider, que permite usar cualquier malla como cuerpo físico del objeto, aunque con un mayor costo de rendimiento y algunas limitaciones. La recomendación es usar formas primitivas para representar tus objetos, o combinaciones de ellas; por ejemplo, un árbol podría modelarse con dos colisionadores Box, uno para el tronco y otro para la copa. Por ejemplo, el que vemos en la figura 5.1.

Sin embargo, en algunos casos no es necesario complicar tanto las cosas. Por ejemplo, si solo queremos que el jugador colisione con el tronco, ya que nunca alcanzará la copa, un colisionador Box que abarque todo el tronco puede ser suficiente, este tipo de decisiones ayudará a mejorar el rendimiento. Además, algunas formas complejas no pueden representarse adecuadamente ni siquiera combinando colisionadores primitivos, como en el caso de rampas o pirámides. En estas situaciones, la única solución sería utilizar un Mesh Collider, que usa una malla 3D para las colisiones.

Para profundizar en el uso de colisionadores en Unity, exploraremos una escena diseñada específicamente para ilustrar cómo se configuran y aplican en diferentes objetos del entorno. Esta práctica nos permitirá observar de manera directa

cómo los colisionadores interactúan con otros elementos en un entorno de juego, proporcionando una comprensión más sólida de su funcionalidad y su implementación.

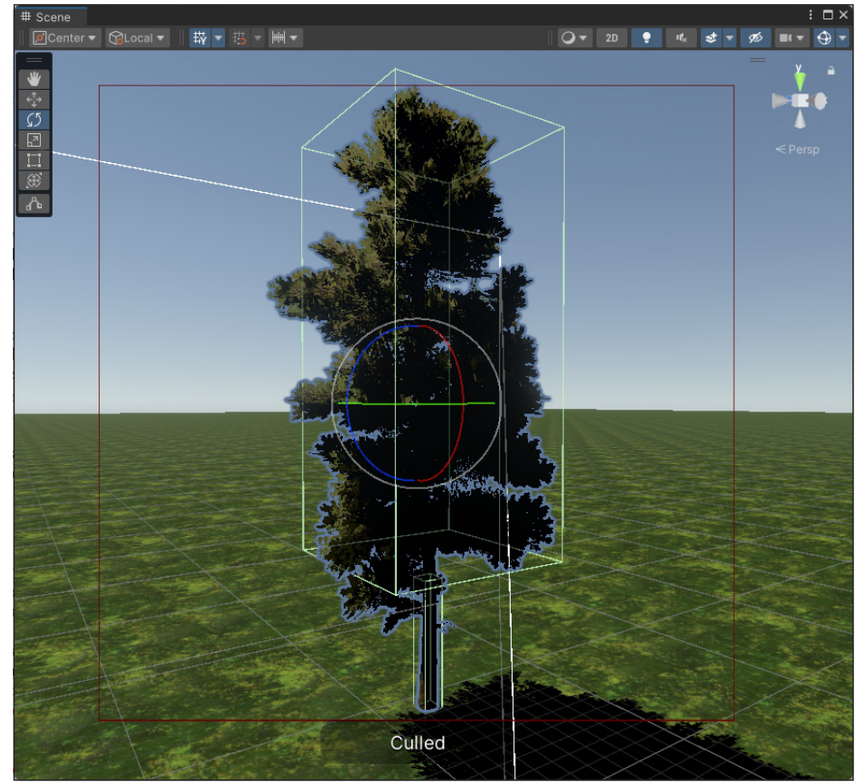

Figura 5.1. Ejemplo de composición de colisionadores Box.

Escena de ejemplo

En lugar de añadir colisionadores manualmente a tus objetos, una excelente manera de familiarizarte con la variedad de colisionadores y su aplicación en Unity es explorando la escena de ejemplo proporcionada en el paquete StarterAssets. Esta escena, en la figura 5.2, ubicada en StarterAssets>ThirdPersonController>Scenes>Playground, está repleta de objetos que ya cuentan con diferentes tipos de colisionadores configurados de manera óptima. Al estudiar esta escena, podrás observar cómo Unity maneja las colisiones y cómo se configuran los colisionadores en diversos contextos dentro de un entorno de juego.

9

INTRODUCCIÓN A LA INTERFAZ DE USUARIO

Introducción

La interfaz de usuario (UI) es una de las piezas clave en el desarrollo de videojuegos, ya que actúa como el puente entre el jugador y el mundo del juego. A través de una UI bien diseñada, se logra que la experiencia de juego sea intuitiva, accesible y atractiva. En este capítulo, nos adentraremos en la creación de interfaces usando UI Toolkit, una de las herramientas más recientes de Unity para el desarrollo de UI. Aunque en capítulos anteriores hemos trabajado con el sistema tradicional de uGUI, UI Toolkit ofrece una forma más moderna y flexible de construir interfaces dinámicas, orientada a la adaptabilidad y al rendimiento.

Aprenderemos cómo crear y gestionar interfaces utilizando UI Documents y el editor UI Builder, exploraremos cómo aplicar estilos reutilizables a través de USS (*Unity Style Sheets*) y veremos cómo hacer que nuestras interfaces se adapten a diferentes resoluciones de pantalla. Además, compararemos las capacidades de UI Toolkit con las de uGUI, para que puedas elegir la mejor herramienta para tus proyectos según tus necesidades. Al finalizar el capítulo, tendrás una comprensión sólida de cómo usar UI Toolkit para construir UI robustas y atractivas, preparándote para las futuras versiones de Unity.

En este capítulo cubriremos los siguientes temas:

- Por qué usar UI Toolkit.
- Crear interfaces básicas con UI Toolkit.
- Aplicar estilos con USS.
- Ajustes de posición y escalabilidad.
- Añadir interactividad a la UI.

Ahora que hemos entendido la importancia de UI Toolkit y por qué es relevante aprenderlo, es el momento de profundizar en su implementación práctica. A lo largo de este capítulo, exploraremos cómo utilizar esta herramienta para crear interfaces adaptativas y modernas que puedan elevar la experiencia de usuario en nuestros proyectos de Unity. Veremos cómo definir, organizar y diseñar elementos visuales de manera eficiente, aprovechando las capacidades de UI Toolkit para crear una interfaz que no solo se vea bien, sino que también funcione de manera óptima en diferentes dispositivos y resoluciones.

Pasemos a desglosar los conceptos clave y a realizar ejemplos prácticos que te permitirán dominar esta nueva forma de crear UI en Unity.

Por qué usar UI Toolkit

UI Toolkit es la nueva herramienta de Unity diseñada para crear interfaces de usuario (UI) de manera más eficiente y moderna. Aunque el sistema uGUI ha sido durante mucho tiempo el estándar en Unity, la llegada de UI Toolkit introduce una forma de trabajo que está inspirada en tecnologías web, como HTML y CSS, haciendo que la transición para desarrolladores web sea más sencilla. UI Toolkit se destaca por su enfoque en la flexibilidad y la escalabilidad de las interfaces, permitiendo crear UI que se adapten mejor a diferentes resoluciones y dispositivos. Además, su diseño modular facilita el mantenimiento y la extensión de la UI de los proyectos.

A medida que Unity continúa evolucionando, es importante que los desarrolladores conozcan tanto el sistema uGUI como UI Toolkit. Esto asegura que, en el futuro, puedan trabajar con proyectos que utilicen ambas tecnologías y, eventualmente, aprovechen las mejoras y optimizaciones que UI Toolkit ofrece para interfaces más complejas y adaptativas.

Evolución hacia UI Toolkit y diferencias con uGUI

UI Toolkit representa un cambio significativo en la forma de crear interfaces en Unity. Está diseñado para aprovechar la experiencia de los desarrolladores familiarizados con la creación de interfaces web, utilizando UXML para definir la estructura de la UI y USS (*Unity Style Sheets*) para el estilo, de manera similar a HTML y CSS. Esto lo convierte en una herramienta poderosa para diseñar UI que sean tanto estéticamente agradables como altamente funcionales.

A diferencia de uGUI, que se basa en GameObjects y componentes, UI Toolkit introduce un enfoque basado en documentos. Esto significa que las interfaces se pueden definir y gestionar como archivos independientes, facilitando la modularidad y la reutilización de componentes. Además, UI Toolkit permite una separación más clara entre la lógica y la presentación, lo que puede resultar en un código más limpio y fácil de mantener.

¿QUÉ ES UGUI?

uGUI, también conocido como Unity GUI, es el sistema de interfaz de usuario integrado en Unity desde la versión 4.6. Se basa en el uso de un Canvas que actúa como una especie de "lienzo" donde se colocan los diferentes elementos de la interfaz, como botones, textos, imágenes y paneles. Cada uno de estos elementos se trata como un GameObject, lo que permite integrarlos de manera natural en la jerarquía de objetos de la escena. Esto facilita su manipulación, animación y control a través de *scripts* en C#.
uGUI es muy versátil y ha sido el estándar durante muchos años para la creación de interfaces de usuario en Unity. Además, su diseño visual en el editor de Unity permite a los desarrolladores y diseñadores construir y ajustar la UI de manera intuitiva, lo que lo convierte en una herramienta accesible tanto para principiantes como para desarrolladores más avanzados. Sin embargo, debido a que cada elemento de la UI es un GameObject, puede consumir más recursos en proyectos con interfaces muy complejas.

Ventajas de UI Toolkit para proyectos futuros

Adoptar UI Toolkit hoy puede ser una ventaja para estar preparados para futuras versiones de Unity, donde se espera que se convierta en el estándar para la creación de interfaces. Aunque aún no tiene paridad completa de características con uGUI, UI Toolkit ya ofrece mejoras en términos de rendimiento y facilidad de diseño para ciertos tipos de aplicaciones, como herramientas editoriales y UI más complejas que requieren adaptabilidad a diferentes dispositivos.

Escalabilidad y adaptabilidad

Una de las principales ventajas de UI Toolkit es su capacidad para crear interfaces adaptables a distintas resoluciones de pantalla, algo que es cada vez más importante en un mundo de dispositivos variados. Gracias a sus herramientas de diseño *responsive*, los desarrolladores pueden crear interfaces que se ajusten automáticamente al tamaño de la pantalla, mejorando la experiencia de usuario en móviles, *tablets* y monitores de alta resolución.

¿CUÁNDO UTILIZAR UI TOOLKIT?

Aunque uGUI sigue siendo una opción viable para proyectos que ya están en desarrollo, UI Toolkit es ideal para proyectos nuevos donde la flexibilidad y la eficiencia en el diseño de UI son primordiales. Además, es una excelente opción para desarrolladores que desean

14

LIBRERÍAS DE TERCEROS GRATUITAS E INDISPENSABLES

Hemos llegado al último capítulo de este libro, un espacio dedicado a potenciar tu flujo de trabajo con librerías y herramientas de terceros que han demostrado ser indispensables en el desarrollo de videojuegos con Unity. Estas herramientas no solo facilitan la creación de proyectos de alta calidad, sino que también optimizan procesos que, de otra forma, podrían consumir mucho tiempo y recursos.

En el vasto ecosistema de Unity, la comunidad de desarrolladores ha creado soluciones innovadoras que abarcan desde la gestión de datos hasta la animación, el diseño visual y la optimización. Este capítulo incluye herramientas que han sido las más comentadas por los programadores de videojuegos a lo largo de 2024, basándonos en consultas realizadas en foros y comunidades. Es importante destacar que no tengo ninguna relación comercial con los creadores de los *assets* de pago mencionados.

Al integrar estas herramientas en tus proyectos, no solo mejorarás la eficiencia del desarrollo, sino que también desbloquearás nuevas posibilidades creativas. En este capítulo, exploraremos una selección de librerías gratuitas y *premium*, ampliamente utilizadas y recomendadas por la comunidad. Cada recurso será presentado con su descripción, capturas representativas y un enlace directo para que puedas incorporarlo fácilmente en tu proyecto. Además, discutiremos cómo estas herramientas pueden transformar aspectos específicos del desarrollo, desde la localización hasta la creación de efectos visuales avanzados.

Sin más preámbulos, comencemos a descubrir estas joyas del desarrollo con Unity, herramientas que no solo enriquecerán tus proyectos actuales, sino que también sentarán una base sólida para tus futuros juegos.

Firebase para Unity

Firebase es una plataforma de desarrollo integral creada por Google que ofrece una colección de herramientas y servicios para facilitar el desarrollo de aplicaciones y videojuegos. En Unity, Firebase permite integrar funcionalidades avanzadas que mejoran tanto la experiencia del usuario como la gestión del proyecto.

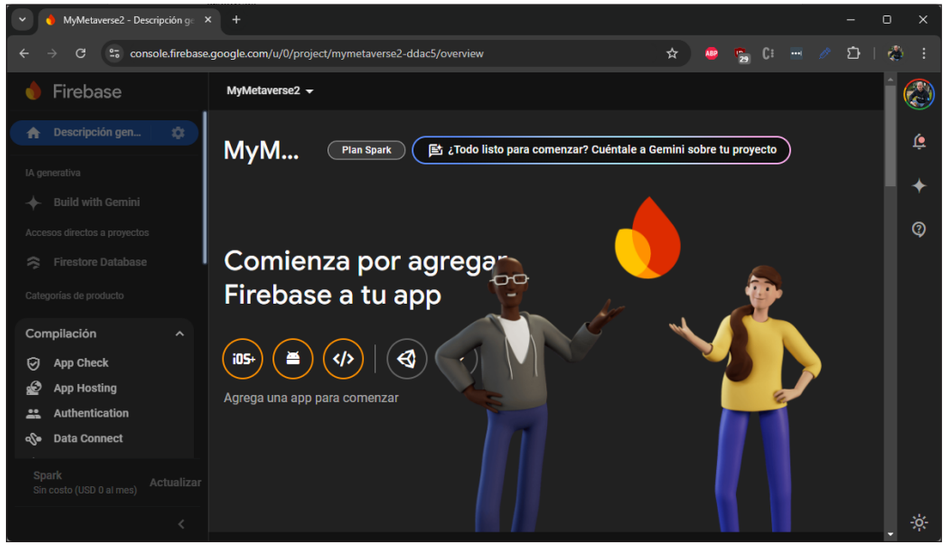

Figura 14.1. *Dashboard* Firebase.

¿Qué puedes hacer con Firebase en Unity?

1. **Autenticación:** Firebase Authentication proporciona una solución fácil para gestionar el inicio de sesión de usuarios. Puedes integrar autenticación por correo electrónico, Google, Facebook y más, permitiendo a tus usuarios acceder a tus juegos sin complicaciones.
2. **Base de datos en tiempo real:** Firebase Realtime Database permite almacenar y sincronizar datos entre tus usuarios en tiempo real. Esto es ideal para juegos multijugador o sistemas de puntuación global.
3. **FireStore:** Si necesitas una base de datos escalable y con más capacidades de consulta, Firestore es una gran opción. Ofrece una estructura más flexible y potente que Realtime Database.
4. **Analíticas avanzadas:** Firebase Analytics proporciona métricas detalladas sobre el comportamiento del usuario en tu juego. Desde eventos básicos como inicio de sesión hasta personalización avanzada de objetivos, puedes tomar decisiones informadas para mejorar la experiencia de juego.

5. **Mensajería *push*:** Con Firebase Cloud Messaging (FCM), puedes enviar notificaciones personalizadas a tus usuarios, mantenerlos comprometidos con tu juego o informarles sobre actualizaciones y eventos.
6. **Almacenamiento en la nube:** Firebase Storage permite alojar y servir archivos, como imágenes o vídeos, de manera rápida y eficiente. Ideal para proyectos que dependen de contenido dinámico.

¿Por qué usar Firebase en Unity?

Firebase simplifica procesos complejos como la gestión de usuarios y la sincronización en tiempo real. Además, su escalabilidad y facilidad de uso la convierten en una herramienta ideal tanto para desarrolladores independientes como para estudios de desarrollo profesional. Con Firebase, puedes centrarte en la experiencia del jugador mientras la plataforma maneja aspectos críticos de *backend* y analíticas.

Descarga e información: `https://firebase.google.com`.

Vuforia Engine

Vuforia Engine es una de las plataformas más destacadas en el desarrollo de aplicaciones de realidad aumentada (RA). Con su capacidad para transformar objetos físicos en experiencias digitales interactivas, se ha convertido en una herramienta esencial para desarrolladores que desean incorporar RA en sus proyectos. Su integración fluida con Unity permite aprovechar todo su potencial, desde el reconocimiento de imágenes hasta el seguimiento avanzado de objetos, ofreciendo una solución robusta tanto para principiantes como para expertos.

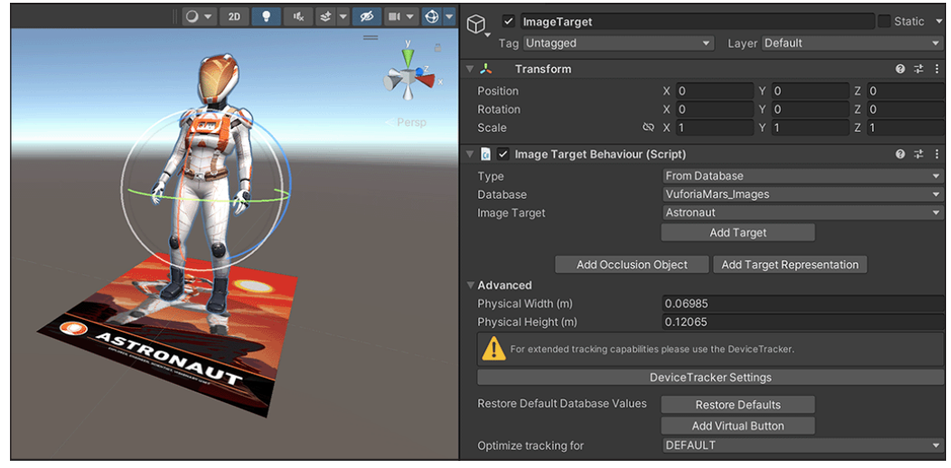

Figura 14.2. Imagen promocional de Vuforia Engine.

EL PRIMER LIBRO EN ESPAÑOL QUE TE GUÍA EN EL DESARROLLO DE VIDEOJUEGOS CON UNITY, DESDE LOS FUNDAMENTOS HASTA LA CREACIÓN DE *BUILDS* EJECUTABLES.

Dirigido tanto a principiantes como a desarrolladores con experiencia, este libro te guía desde el uso del Unity Editor hasta la creación de videojuegos completos y optimizados. Aprenderás a manipular GameObjects, programar en C# y Visual Scripting, diseñar terrenos, personajes jugables y NPCs, incluyendo la implementación de inteligencia artificial avanzada con ChatGPT. Además, explorarás cómo integrar música, sonidos, animaciones y cinemáticas, y a optimizar el rendimiento para dispositivos reales.

Incluye una introducción práctica a la realidad aumentada con ARFoundation, técnicas para generar *builds* ejecutables y una selección de librerías gratuitas para enriquecer tus proyectos. Con ejemplos prácticos, ejercicios y acceso al código fuente en GitHub, este libro es la herramienta perfecta para llevar tus ideas al siguiente nivel en el desarrollo de videojuegos con Unity.

ISBN 978-84-415-5163-3
2315272
9 788441 551633

ANAYA MULTIMEDIA

www.anayamultimedia.es