Escuela Politécnica Superior de Jaén
Grado en Ingeniería Informática

Universidad de Jaén
Escuela Politécnica Superior
de Jaén

TRABAJO FIN DE GRADO

DESARROLLO DE UN PROTOTIPO DE VIDEOJUEGO PARA DISPOSITIVOS MÓVILES

Alumno

Víctor Sánchez Rivero

Tutores

Juan Roberto Jiménez Pérez
(Departamento de Informática)

José Negrillo Cárdenas
(Departamento de Informática)

Septiembre, 2019

Universidad de Jaén

Departamento de Informática

Don Juan Roberto Jiménez Pérez y Don José Negrillo Cárdenas, tutores del Trabajo Fin de Grado titulado: '**Desarrollo de un prototipo de videojuego para dispositivos móviles**', que presenta Don Víctor Sánchez Rivero, otorgan el visto bueno para su entrega y defensa en la Escuela Politécnica Superior de Jaén.

Jaén, Septiembre de 2019

El alumno: Los tutores:

Víctor Sánchez Rivero

Juan Roberto Jiménez Pérez

José Negrillo Cárdenas

Agradecimientos

Me gustaría agradecer a mis compañeros, profesores, amigos y familiar por el apoyo mostrado durante la realización de este proyecto.

En concreto a mis padres y hermana, por apoyarme para continuar adelante.

A mis amigos y probadores Juan Mira, Javi, Juan Mesa y Chemi, por animarme a llevar a cabo este trabajo y ayudarme con sus sugerencias y comentarios para mejorar el juego y detectar los errores.

Y a mis profesores, por ayudarme a lograr los conocimientos necesarios para completar este proyecto, y en especial a Jose y Roberto por guiarme y animarme a realizar este prototipo de videojuego.

FICHA DEL TRABAJO FIN DE TÍTULO	
Titulación	Grado en Ingeniería Informática
Modalidad	Proyecto de Ingeniería
Especialidad	Tecnologías de la Información
Mención	Sistemas Gráficos
Idioma	Español
Tipo	Específico
TFT en equipo	No
Fecha de asignación	09/04/2019
Descripción corta	Dado el auge que está experimentando el mercado de los videojuegos a nivel global, y especialmente, en el sector de los videojuegos para dispositivos móviles, en este proyecto se ha propuesto elaborar un prototipo de videojuego para dispositivos Android que exprima las capacidades específicas que ofrecen este tipo de dispositivos. En concreto, se ha utilizado el acelerómetro como eje del sistema de control para desarrollar un juego del género de plataformas de niveles cortos en los que el objetivo es hacer rodar una bola hasta la meta.

NORMAS APLICADAS EN ESTE DOCUMENTO	
LOCALES	
TFT-UJA:2017	Normativa de Trabajos Fin de Grado, Fin de Máster y otros Trabajos Fin de Título de la Universidad de Jaén (Normativa marco UJA aprobada en Consejo de Gobierno)
TFT-EPSJ:2017	Normativa sobre Trabajos Fin de Grado y Fin de Máster en la Escuela Politécnica Superior de Jaén (Normativa EPSJ aprobada en Junta de Escuela)
TFT-EPSJ	Criterios de evaluación y normas de estilo para TFG y TFM de la Escuela Politécnica Superior de Jaén
NACIONALES	
UNE 157001:2014	Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico
UNE 157801:2007	Criterios generales para la elaboración de proyectos de sistemas de información
INTERNACIONALES	
ISO 2145:1978	Documentación - Numeración de divisiones y subdivisiones en documentos escritos
APA 6ª edición	Estilo de referencias y citas de APA (American Psychological Association)

Contenido

Índice de ilustraciones

Índice de tablas

1 - INTRODUCCIÓN

En el presente documento se van a detallar las etapas de planificación, diseño y desarrollo que han dado lugar a la elaboración del prototipo de un videojuego para dispositivos móviles basado en el uso del acelerómetro como principal sistema de control. En los apartados 1 a 6 de este documento se presentarán las labores de planificación desempeñadas, incluyendo el establecimiento de los objetivos del trabajo, la idea general del juego, el software empleado, la metodología de desarrollo y la realización de un estudio sobre los métodos de interacción en dispositivos móviles. En los apartados 7 a 9 se detalla la etapa del diseño, profundizando en el diseño de la interfaz, las mecánicas y los niveles. El apartado 10 comprende el propio desarrollo del proyecto a lo largo de varias iteraciones. Tras esto, se comentan las posibilidades de comercialización y se extraen las conclusiones sobre el trabajo. Finalmente, el apartado 13 se corresponde con varios anexos en los que se muestran las encuestas recibidas de los probadores y los bocetos de los niveles.

La proliferación de los smartphones ha permitido llevar a los videojuegos más allá de las consolas y ordenadores, consiguiendo una expansión de la industria del videojuego a un público mucho más amplio. Esta expansión viene propiciada por la concesión de una mayor libertad para disfrutar de un videojuego en cualquier momento y la supresión del impedimento de tener que desembolsar una cantidad considerable de dinero en un hardware específico para jugar.

Esto ha ocasionado que los juegos para dispositivos móviles generen más ingresos que aquellos para las plataformas clásicas (consolas y ordenador), como se aprecia en la Ilustración 1.1, situándose en la imponente cifra de 70.300 millones de dólares de beneficio en 2018, continuando con una tendencia al alza que le ha llevado a incrementar en más de un 25.5% el importe recaudado en 2017. En concreto, el 80% de esa cifra (un 41% del total) se corresponde con la facturación en smartphones, y el otro 20% a tabletas [1].

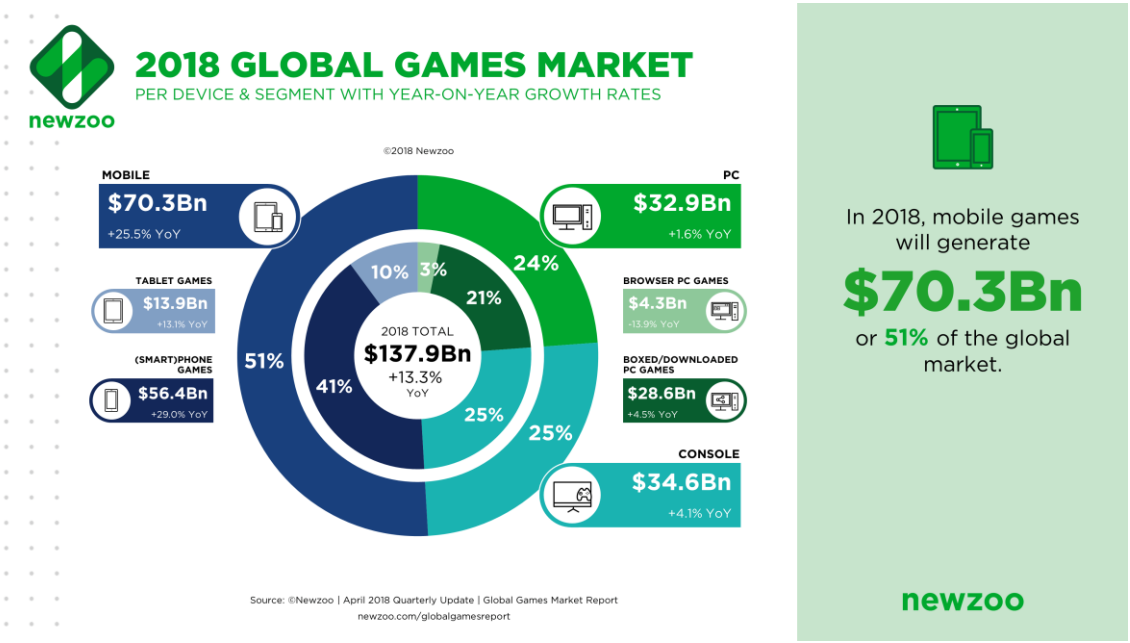


Ilustración 1.1 Mercado global de videojuegos en 2018 segmentado por dispositivo

De acuerdo con la Ilustración 1.2, el total de ingresos generado por la industria del videojuego para 2018 es de 137.900 millones de dólares, lo que supone un 13.3% más que lo facturado el año anterior. Y se espera que la tendencia siga en aumento en los próximos años, especialmente en el sector de los videojuegos para móviles (el 59% de la facturación de la industria en 2021).

Ilustración 1.2 Beneficios por dispositivo estimados del mercado global del videojuego

Categoría	Total Apps	Nota Media (0-5)	Apps más 50.000 descargas	Apps pago	Precio medio	Apps de baja calidad
Acción	27264	4.15	5485 (20%)	1497 (5%)	$3.50	1947 (5%)
Arcade	58861	4.35	4618 (8%)	2896 (5%)	$3.85	4193 (7%)
Aventura	24022	4.20	2983 (12%)	1921 (8%)	$4.89	2072 (9%)
Carreras	12117	4.11	3355 (28%)	440 (4%)	$3.60	452 (4%)
Cartas	8662	4.09	2139 (25%)	656 (8%)	$2.52	278 (3%)
Casino	7321	4.09	1576 (22%)	524 (7%)	$7.56	349 (5%)
Casual	50740	4.19	7965 (16%)	2466 (5%)	$5.32	3751 (7%)
Deportes	9541	4.03	2280 (24%)	608 (6%)	$4.61	522 (5%)
Educativo	29757	4.28	3023 (10%)	3503 (12%)	$2.97	2571 (9%)
Estrategia	8660	4.13	1907 (22%)	875 (10%)	$3.82	350 (4%)
Juegos de mesa	10506	4.17	1603 (15%)	705 (7%)	$3.45	631 (6%)
Juegos de rol	8081	4.08	2722 (34%)	933 (12%)	$4.16	275 (3%)
Música	3956	4.03	638 (16%)	308 (8%)	$2.52	598 (15%)
Palabras	11189	4.31	1568 (14%)	421 (4%)	$2.69	791 (7%)
Preguntas y respuestas	17015	4.22	1595 (9%)	376 (2%)	$7.86	1906 (11%)
Puzles	59119	4.27	6359 (11%)	4074 (7%)	$3.44	4744 (8%)
Simulación	24018	3.95	8441 (35%)	1232 (5%)	$6.97	1200 (5%)

Tabla 1.1 Tabla resumen de juegos por categorías en Google Play

Según la Tabla 1.1, *Arcade* es la segunda categoría con mayor número de juegos, por detrás de *Puzle* y seguido de *Casual*. *Arcade* es la categoría con menor

porcentaje de juegos que consigue alcanzar más de 50.000 descargas (8%) y sin embargo es la categoría con mejor puntuación media (4.35) [2]. Estos datos nos llevan a pensar que *Arcade* es una categoría muy saturada, en la que es difícil despuntar incluso con un buen juego.

En la categoría *Casual* hay menos juegos que en *Arcade* y el porcentaje de juegos que consiguen alcanzar más de 50.000 descargas asciende hasta el 16% por lo que sería recomendable usar la categoría *Casual* frente a la *Arcade*.

Existen un total de 2.600.569 aplicaciones para Android en Google Play a día 13 de abril de 2019, de las cuales sólo el 123.961 (4.8%) son de pago [3].

Los juegos que obtienen un mayor beneficio diario tanto en Google Play como en App Store son “Free to Play”, es decir, son juegos que no requieren realizar un desembolso inicial por la adquisición del producto, pero pueden requerir compras internas (micropagos o microtransacciones) para acelerar la progresión dentro del juego u obtener contenidos exclusivos, generalmente de carácter estético. Según los propios rankings de Google Play no se ha encontrado ningún juego de pago entre los 200 más populares y tan solo uno en el ranking de los juegos con mayores ingresos (*Minecraft* en el puesto 109) [4-6]. En el caso de la App Store, entre las 100 apps con mayores beneficios, 71 eran juegos, y únicamente uno de ellos era de pago (nuevamente *Minecraft*, en este caso en el puesto 50) [7].

Es por ello que se ha decidido que el juego a desarrollar en este TFG se comercializará bajo la categoría *Casual* y siguiendo la filosofía del juego “Free to Play” con micropagos que permitan suprimir esperas causadas por la recuperación de vidas, eliminación de publicidad o adquisición de elementos estéticos.

2 - OBJETIVOS

El objetivo principal de este TFG es diseñar un prototipo funcional de un videojuego para dispositivos móviles cuyo medio principal de control sea el acelerómetro y que posteriormente pueda ser completado con nuevos niveles y posibles refinamientos de ciertos aspectos de cara a su comercialización. Se pueden definir una serie de objetivos generales:

- Identificar el juego a desarrollar, destacando las características principales del mismo, así como el público objetivo.
- Definir las mecánicas de juego principales.
- Seleccionar el motor de juego más adecuado, así como otras herramientas complementarias que se requieran en el desarrollo o interacción con el juego específico.
- Analizar y diseñar la interfaz e interacción con el usuario, así como identificar los dispositivos de interacción más adecuados al tipo de videojuego que se propone.
- Analizar y diseñar el mundo sobre el que se desarrolla, los personajes, animaciones, efectos visuales y de sonido.
- Evaluar el prototipo.

3 - VISIÓN GENERAL DEL JUEGO Y CLASIFICACIÓN

Clasificación del juego:

- Título: Super Roll the Ball.
- Género: Plataformas, casual.
- Modos de juego: Vista cenital (movimientos relativos al mundo) y vista en tercera persona con la cámara en la espalda (movimientos relativos al personaje).
- Plataforma: Prototipo en dispositivos Android, fácilmente extensible a dispositivos IOS.
- Idioma: Textos en español, sin voces.
- Orientado a todo tipo de jugadores, pero especialmente a los jugadores casual. La organización del juego en niveles cortos, unido a la gran portabilidad de los dispositivos móviles, lo hacen un juego atractivo para dedicarle pequeñas sesiones de juego en ratos libres.
- Clasificación PEGI: PEGI 3, el juego es adecuado para todas las edades, ya que no contiene palabras malsonantes ni contenido, ya sea en forma de sonidos o imágenes, que pueda asustar a los pequeños, ni violencia [8].

El objetivo principal del juego es ir superando niveles y logrando en ellos el menor tiempo posible y la máxima puntuación de 3 estrellas por nivel. Para ello, cada nivel se deberá completar en un tiempo inferior a uno objetivo y recolectando una serie de coleccionables. El nivel se completa llegando hasta la meta, y si el personaje se cae se deberá empezar de nuevo el nivel.

La mecánica principal del juego consiste en desplazarse rodando por el escenario como consecuencia de la inclinación del dispositivo móvil (o en caso de que se seleccione así en el menú de opciones, mediante un botón deslizante). Esta mecánica principal es el núcleo del juego y se destinará una parte importante de los recursos en lograr que el desplazamiento sea lo suficientemente divertido por sí mismo. Los niveles se irán haciendo más complicados conforme se avance en el juego, estarán divididos en mundos, y cada mundo irá incorporando nuevas mecánicas más complejas.

Este juego está inspirado en las mecánicas de juego clásicos como *Super Monkey Ball* y *Marble Madness* [9-10]. A la vez que se intenta construir un juego innovador al trasladar estas mecánicas, respaldadas por un sistema de control basado en acelerómetro a dispositivos móviles.

4 - SOFTWARE UTILIZADO

4.1 - Elección del motor de juegos

Una decisión elemental a la hora de desarrollar un videojuego es seleccionar cuidadosamente el motor de videojuegos en el que va a desarrollarse. Los motores de videojuegos proveen de un motor específico para renderizar gráficos tanto en 2D como en 3D, un motor dedicado a la simulación de las leyes físicas como la gravedad, las colisiones o el rozamiento y un motor de sonido. Estos motores también aportan una serie de funcionalidades que ayudan a la animación, la creación de *scripts*, la inteligencia artificial, la gestión de las entradas de los usuarios o la gestión de memoria.

En la actualidad existen multitud de motores de videojuegos. Los hay propios, es decir, desarrollados y licenciados como software propietario de una compañía de videojuegos concreta para usarlo en sus propios juegos. Y también hay motores de juego que se ponen a disposición del público por medio de diferentes tipos de licencias, variando en función de si el motor es usado para fines personales o educativos, o con fines profesionales, en cuyo caso la licencia suele ser de pago con un coste variante en función del tamaño de la empresa o de los beneficios obtenidos con los productos creados utilizando dicho motor de juegos. Algunos de los motores de videojuegos más populares en la actualidad para desarrollar un videojuego son Unity, Unreal Engine o CryEngine, entre otros [11-13].

Unity es un motor de videojuegos creado por Unity Technologies y se perfila como una gran opción para los usuarios que se deseen iniciar con un motor de videojuegos debido a la enorme cantidad de tutoriales existentes sobre su funcionamiento, tanto oficiales como no oficiales. Otra de sus ventajas es la amplia comunidad de usuarios que posee, por lo que encontrar respuestas a los problemas que puedan aparecen suele ser sencillo consultando en sus foros. Estas amplias cifras de usuarios lo llevan a posicionarse como el motor empleado para crear la mayoría de los juegos del mundo en la actualidad y también provoca que reciba una gran cantidad de actualizaciones de manera frecuente para mantener el motor lo más actualizado posible. La versión actual (a día 26 de julio de 2019) es la 2019.1.12.

Este motor es muy versátil, ya que permite soporte multiplataforma de manera sencilla, sin necesidad de volver a compilar el proyecto para adaptarlo a la nueva

plataforma. Soporta la amplia mayoría de las plataformas existentes en la actualidad: IOS, Android, Windows, Mac Os, Linux, Nintendo Switch, PS4, Xbox One, WebGL, SmartTVs y múltiples dispositivos de realidad virtual y aumentada. También se permite la ampliación de las funciones ofrecidas por este motor mediante herramientas personalizadas que se pueden obtener de la Asset Store, su tienda de recursos [14]. Este motor no sólo se utiliza para el desarrollo de videojuegos, sino que es empleado en la industria automotriz, la animación en películas y cinemáticas e incluso en el sector de la arquitectura, ingeniería y construcción.

Unity también ofrece una serie de soluciones incorporadas para gestionar la monetización dentro de un juego y maximizar así los ingresos generados. Estas soluciones van desde herramientas que permiten monitorizar la actividad de los jugadores, hasta mecanismos para incorporar y gestionar compras desde dentro de la aplicación y publicidad dentro del juego. Respecto a los tipos de licencias ofrecidas se puede optar entre una versión “Personal” gratuita para principiantes si los ingresos o financiación recibidos no exceden los $100.000 anuales, una versión “Plus” para aficionados por $25/mes y una versión “Pro” para profesionales y estudios por $125/mes.

Una segunda opción sería Unreal Engine, creado por Epic Games [15], cuya última versión estable es la 4.22.1 (a día 26 de julio de 2019). Este motor de juegos proporciona a los desarrolladores las herramientas necesarias para crear sus juegos con un gran rendimiento sin sacrificar calidad visual, y todo ello sin necesidad de utilizar complementos o compras adicionales. Cuenta con soporte para multitud de plataformas: PS4, Xbox One, Nintendo Switch, Windows, Mac OS, Linux, HTML 5, IOS, Android y varios dispositivos de realidad virtual.

Al igual que Unity, cuenta con una amplia comunidad de desarrolladores y está respaldado por una empresa con gran experiencia y buen posicionamiento en el sector actualmente.

El uso de este motor es gratuito, así como el acceso a las herramientas, características, códigos fuente, actualizaciones regulares y contenidos de prueba, a cambio del 5% de los ingresos que genere el producto desarrollado una vez se haya comercializado.

Otra opción a tener en cuenta sería CryEngine, desarrollado por la empresa alemana Crytek [16] y popularizado principalmente por la saga Crysis, que revolucionó la industria con su primera entrega al conseguir alcanzar unas cotas de realismo

insospechadas para la época (2007) [17]. La versión actual de este motor es CryEngine 5, y coincidiendo con la salida de esta nueva versión se modificó su modelo de negocio ofreciendo la posibilidad de “pagar lo que quieras”, sólo siendo necesario pagar a Crytek el 5% en concepto de royalties si se superan los $5.000 de ingresos con el producto.

También cuenta con soporte para varias plataformas, como dispositivos de realidad virtual, Windows, Linux, PS4 y Xbox One; gran cantidad de videotutoriales y documentación.

Sin embargo, este motor gráfico está más orientado al desarrollo de videojuegos de alto presupuesto y generalmente “FPS” (Shooters en Primera Persona) de mundo abierto, por lo que hay otras alternativas más atractivas y sencillas de usar para desarrollar un juego de las características que se han planteado en este proyecto. La ausencia de soporte para juegos en dispositivos móviles lo descarta totalmente para este proyecto.

Para este proyecto se utilizará Unity por ser gratuito y por la gran cantidad de tutoriales y el apoyo ofrecido desde los foros de Unity, que se mantienen muy activos. También resulta atractivo el mercado de recursos que es la Asset Store, donde se puede encontrar gran cantidad de material, tanto gratuito como de pago. Una última ventaja que presenta Unity es que ha sido el software empleado en varias de las asignaturas del grado, por lo cual se han adquirido ciertos conocimientos que se desean ampliar con este proyecto al buscar un enfoque diferente frente a lo realizado hasta ahora.

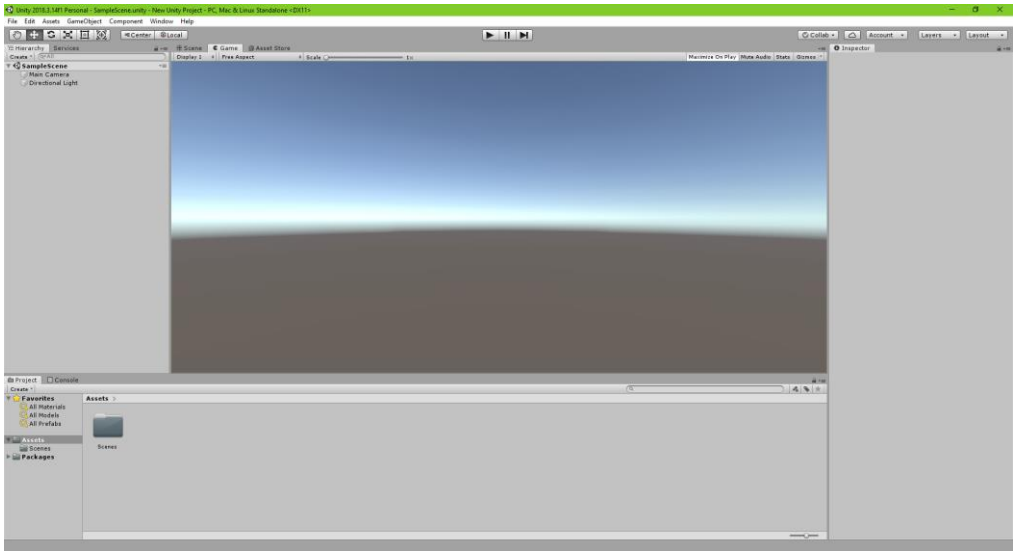


Ilustración 4.1 Unity 2018.3.14f1

4.2 - Adobe Photoshop

Adobe Photoshop es la referencia en cuanto al software de edición de imágenes. Para este proyecto se utilizará la versión CS6 [18]. Este software será utilizado para la edición de ciertas imágenes del framework utilizado para la interfaz, cuyo formato es exclusivo de Adobe Photoshop.

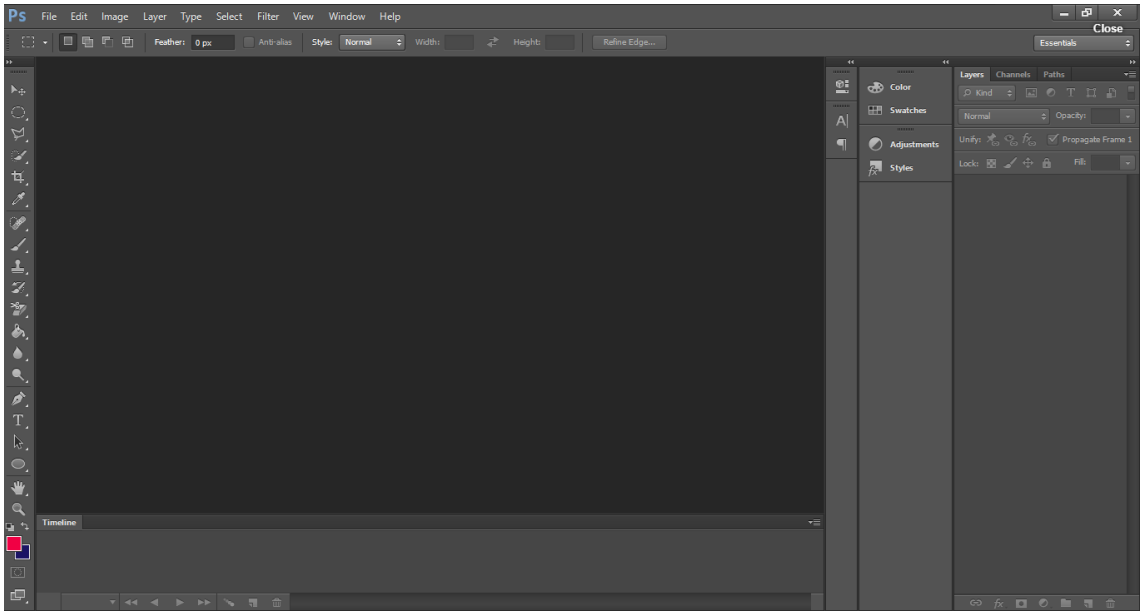


Ilustración 4.2 Adobe Photoshop CS6

4.3 - Paint 3D

Paint 3D es un editor de imágenes gratuito que incorpora la capacidad de trabajar en 3D [19]. Supone la evolución del clásico Paint, con nuevas herramientas y una interfaz renovada. Este programa será utilizado para editar el resto de las imágenes del juego (mediante redimensionado, recorte, etc.) y obtener las imágenes utilizadas en este documento.

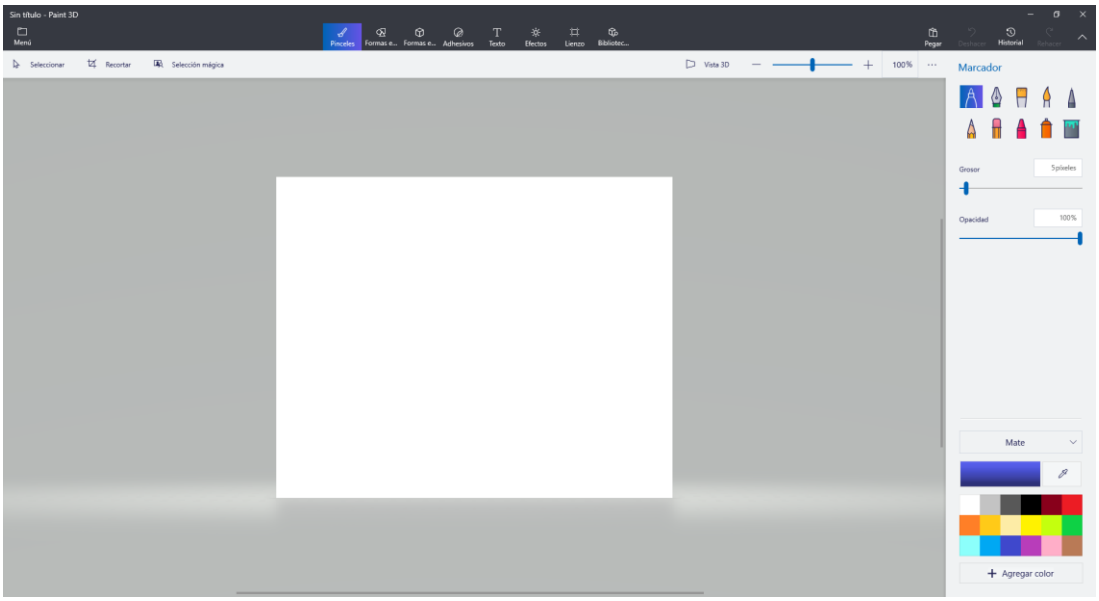


Ilustración 4.3 Paint 3D

4.4 - Microsoft Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado para Windows, Linux y macOS compatible con múltiples lenguajes de programación como C++, C#, Java, Python o PHP [20]. Visual Studio permite desarrollar tanto aplicaciones tradicionales como sitios y aplicaciones web.

Para este proyecto se utilizará la versión Community 2017, la cual es gratuita. Visual Studio incorpora un conjunto de herramientas para el desarrollo de videojuegos con Unity.

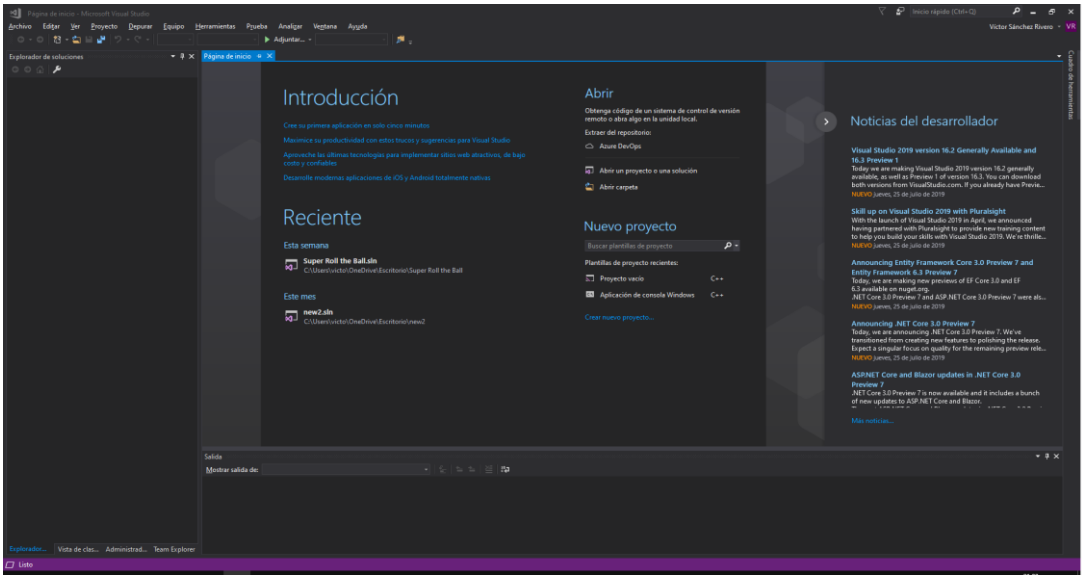


Ilustración 4.4 Microsoft Visual Studio Community 2017

4.5 - Microsoft Word

Microsoft Word es un programa para el procesamiento de textos propiedad de Microsoft y englobado en el paquete ofimático Microsoft Office [21]. Este programa será el utilizado para redactar la presente memoria.

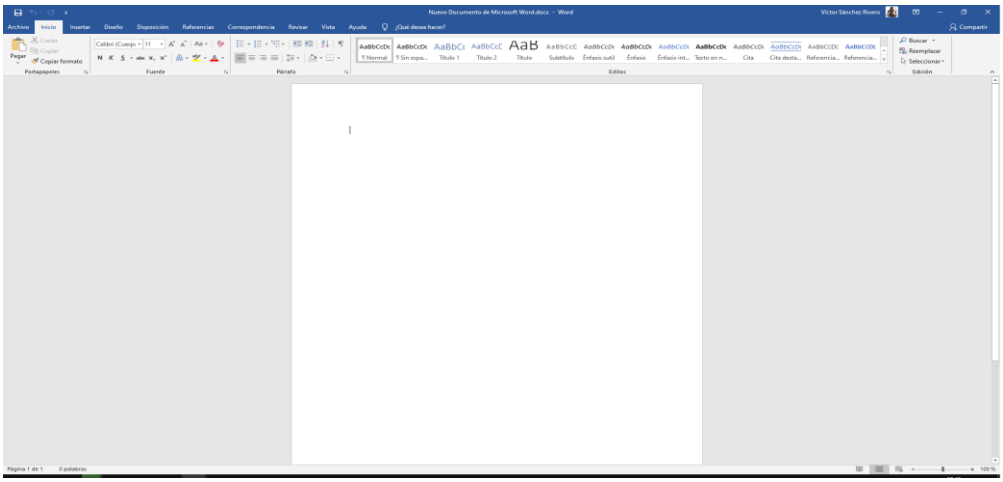


Ilustración 4.5 Microsoft Word 2016

4.6 - Audacity

Este es un software gratuito, de código abierto y multiplataforma para la edición y grabación de audio [22]. Este programa será utilizado para recortar pistas de audio de terceros y adecuarlas a las necesidades de este proyecto, así como para convertir estas pistas a diferentes formatos.

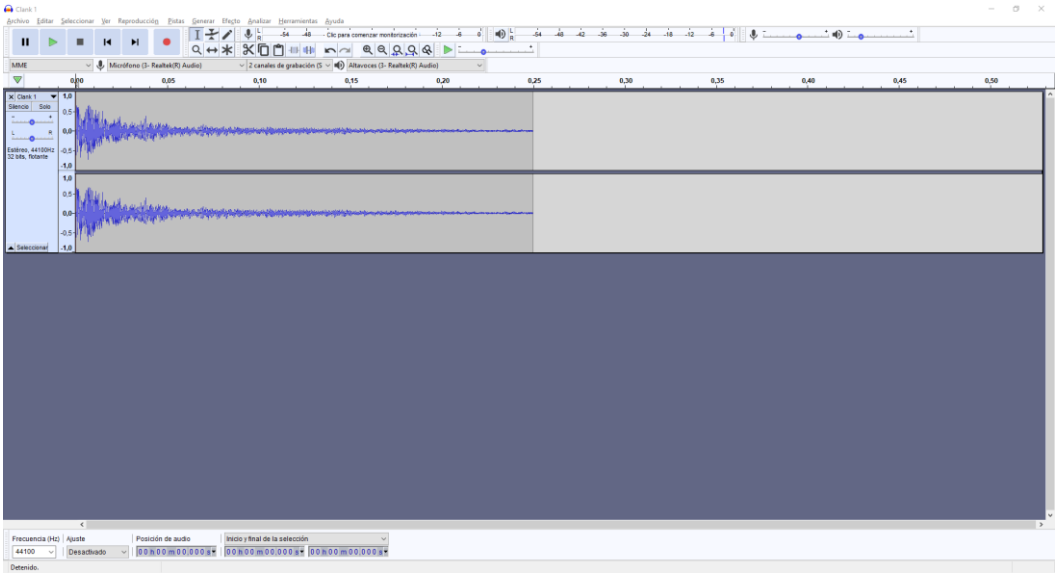


Ilustración 4.6 Audacity

5 - INTERACCIÓN EN DISPOSITIVOS MÓVILES

Las principales diferencias entre construir un juego para ordenador o hacerlo para un dispositivo móvil (en este caso Android) residen en la forma de interactuar con el juego. En un juego para ordenador convencional, las entradas vienen definidas por el uso de un ratón y un teclado, o en su defecto un mando. Un dispositivo móvil no utiliza ni ratón ni botones para interactuar, en su lugar utiliza principalmente pulsaciones sobre la pantalla táctil o sensores específicos como el acelerómetro, el giroscopio, la brújula o el GPS. Por tanto, lo primero que se ha de tener en cuanta cuando se vaya a desarrollar un juego para Android es el sistema de control.

En Unity es posible simular las pulsaciones en una pantalla táctil como entradas de ratón. Pulsar con un dedo sobre la pantalla del dispositivo móvil será interpretado por Unity como un clic izquierdo del ratón en dicha posición. Además, si se está arrastrando el dedo por la pantalla, Unity proporciona su posición exacta, al igual que lo haría con un ratón, con *Input.mousePosition*.

Sin embargo, existen ciertas diferencias cruciales que hacen que esta interpretación de las pulsaciones sobre una pantalla táctil como entradas del ratón no sean viables para sistemas de interacción más complejos. Mediante la pulsación en una pantalla táctil sólo se podrá utilizar el botón izquierdo del ratón, y no el derecho o la rueda. Con este método tampoco se podrán utilizar varios dedos a la vez en la pantalla táctil, pues no tendría equivalencia en el ratón. Y también presentaría problemas con el desplazamiento del ratón sin tener ningún botón pulsado, o desde el otro punto de vista, en una pantalla táctil es posible pasar de un extremo al otro (mediante pulsaciones) sin recorrer las posiciones intermedias, cosa que no ocurre con un ratón.

Como solución a estos inconvenientes existe otra forma de leer las entradas del usuario en un dispositivo móvil y es mediante el uso de *Input.Touch*. Durante cada fotograma se dispondrá de acceso a *Input.touches*, donde se almacenarán todas las pulsaciones (en la forma de *Input.Touch*) que se estén produciendo simultáneamente en este momento. Cada pulsación tiene varios atributos que permiten extraer información sobre la pulsación:

- *fingerId*: es un índice único para la pulsación.
- *position*: la posición en pantalla de la pulsación.

- *deltaPosition*: la variación en posición respecto al último fotograma.
- *deltaTime*: el tiempo que ha pasado desde la última variación en la pulsación.
- *tapCount*: permite saber el número de pulsaciones rápidas (equivalente al doble clic), pero no lo proporcionan los dispositivos Android, por lo que en un dispositivo Android siempre será 1.
- *phase*: el estado de la pulsación.

*Ph*ase proporciona información sobre el estado de la pulsación y así poder reaccionar de forma adecuada ante una pulsación o un arrastre, por ejemplo. El atributo *phase* puede tomar los valores:

- *Began*: si el dedo acaba de tocar la pantalla.
- *Moved*: un dedo que ya estaba sobre la pantalla se ha movido.
- *Stationary*: un dedo que ya estaba sobre la pantalla no se ha movido en el último fotograma.
- *Ended*: un dedo se levanta de la pantalla.
- *Cancelled*: el dispositivo deja de reconocer la pulsación, por ejemplo, si se emplean más de cinco dedos.

Estos dos últimos estados suponen el final de un evento de pulsación.

Cabe destacar que en los dispositivos Android no existe un convenio sobre el número máximo de dedos que puede manejar simultáneamente el dispositivo, y puede variar entre dos y cinco dependiendo de la antigüedad y gama del mismo.

Por otra parte, también hay que enfrentarse al problema de la pérdida de botones, no se dispone de mando o teclado para jugar al juego. La solución habitual a este problema es crear unos botones transparentes en el canvas que realicen las acciones necesarias en el juego, de forma similar a lo mostrado en la Ilustración 5.1.

Ilustración 5.1 Ejemplo botones táctiles

Por último, hay que tener en cuenta las nuevas formas de interacción que nos presentan los dispositivos móviles (acelerómetro, giroscopio, etc.). Estos controles pueden suponer una forma menos precisa o fácil para el usuario, pero en cambio pueden otorgar una experiencia más inmersiva. En Unity se puede extraer el valor del acelerómetro del dispositivo mediante *Input.acceleration*, con lo que se obtiene un Vector3 que indicará la aceleración que se ha producido en cada uno de los ejes en medidas de Fuerza g, que equivale a la aceleración por la gravedad (9.80665 m/s^2). Dichos ejes siguen el esquema de la Ilustración 5.2.

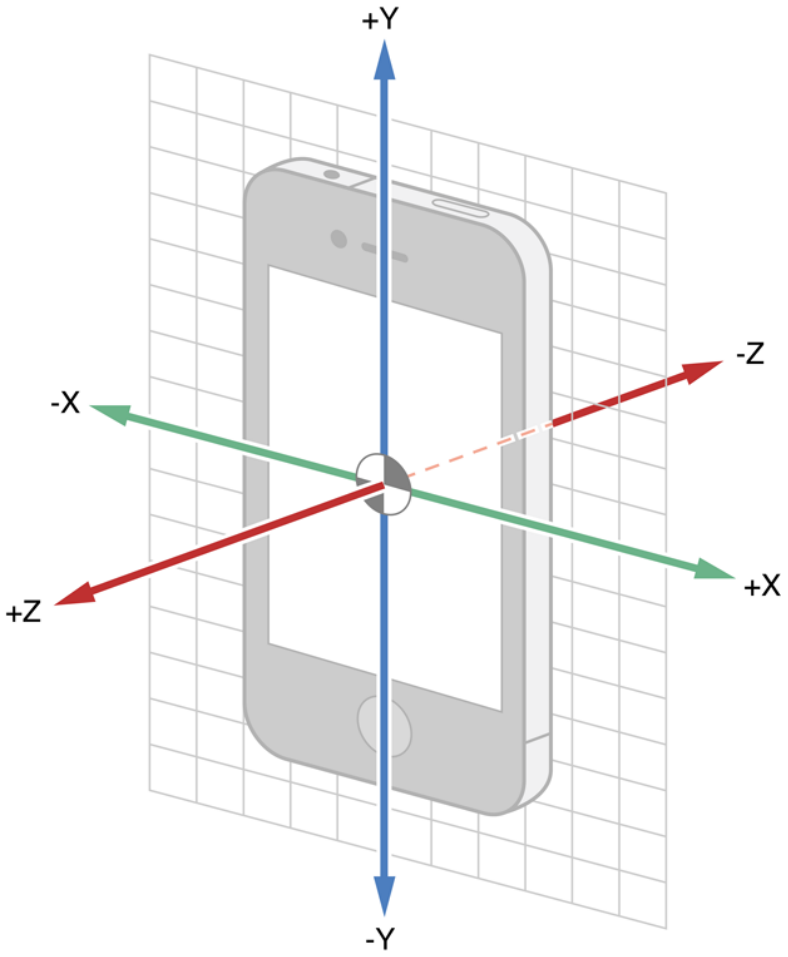


Ilustración 5.2 Ejes acelerómetro

Suele ser habitual aplicar filtros de paso bajo a la señal del acelerómetro para suavizar el ruido ocasionado por pequeñas variaciones involuntarias [23].

6 - PLANIFICACIÓN

La ejecución de este Trabajo de Fin de Grado está programada para ocupar 300 horas, que serán concentradas durante 3 meses de trabajo, en los que se irá desarrollando el videojuego al mismo tiempo que se elabora la presente documentación.

6.1 - Metodología de desarrollo

Para el desarrollo de este proyecto se ha considerado que la metodología más adecuada será una incremental, pues de esta forma es posible desarrollar el producto final del proyecto (el videojuego) refinando en cada iteración aspectos desarrollados anteriormente en base a los resultados de las pruebas realizadas y las valoraciones de diversos probadores, incluido yo mismo (autor del proyecto), los tutores y voluntarios que harán las funciones de probadores o usuarios de pruebas de un producto software aún en desarrollo. Con esto se consigue que las primeras iteraciones (correspondientes con los aspectos más cruciales del proyecto, como son las mecánicas de juego) puedan ser revisadas en múltiples ocasiones con el objetivo de ir refinándolas. Las posteriores iteraciones se construyen a partir de las ya existentes y suelen implicar la corrección o modificación de características de la última iteración.

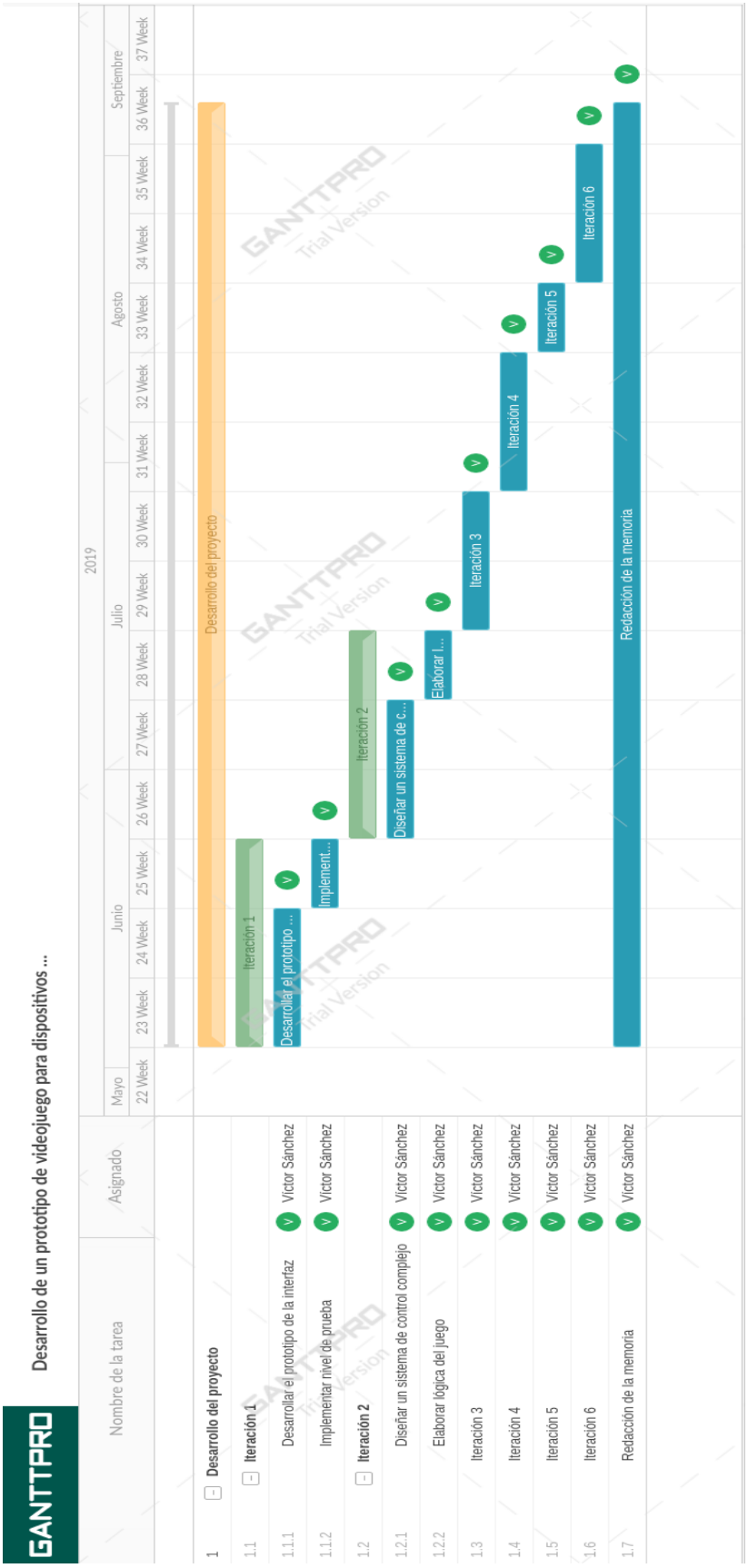
Para este proyecto se ha considerado una división en seis iteraciones:

1. Elaboración de un prototipo de interfaz funcional y un nivel de prueba con mecánicas de movimiento básicas.
2. Elaboración de un sistema de control complejo y la lógica del juego.
3. Ajuste del sistema de control en base a los resultados de las pruebas de los probadores y diseño definitivo de la estética de la interfaz.
4. Diseño e implementación de los niveles del primer mundo.
5. Mejoras gráficas varias, adición de música y efectos de sonido y corrección de posibles errores.
6. Diseño e implementación de los niveles del segundo mundo.

Está previsto realizar entregas del software a los probadores en tres ocasiones a lo largo del desarrollo: tras la segunda iteración para evaluar la funcionalidad de la

interfaz y el sistema de control del juego, tras la cuarta iteración para evaluar la interfaz definitiva y un conjunto de niveles finales y por última vez tras la sexta iteración, que sería la versión final del producto, salvo por la realización de posibles correcciones en base a sus informes.

En la Ilustración 6.1 se presenta el diagrama de Gantt del proyecto.


GANTTPRO
Desarrollo de un prototipo de videojuego para dispositivos ...
Nombre de la tarea
Asignado
2019
Mayo
Junio
Julio
Agosto
Septiembre
22 Week
23 Week
24 Week
25 Week
26 Week
27 Week
28 Week
29 Week
30 Week
31 Week
32 Week
33 Week
34 Week
35 Week
36 Week
37 Week
1 Desarrollo del proyecto
1.1 Iteración 1
1.1.1 Desarrollar el prototipo de la interfaz
1.1.2 Implementar nivel de prueba
1.2 Iteración 2
1.2.1 Diseñar un sistema de control complejo
1.2.2 Elaborar lógica del juego
1.3 Iteración 3
1.4 Iteración 4
1.5 Iteración 5
1.6 Iteración 6
1.7 Redacción de la memoria
Víctor Sánchez
Desarrollo del proyecto
Iteración 1
Desarrollar el prototipo ...
Implement...
Iteración 2
Diseñar un sistema de c...
Elaborar l...
Iteración 3
Iteración 4
Iteración 5
Iteración 6
Redacción de la memoria
GANTTPRO Trial Version

6.2 - Estimación de recursos y costes

Para hacer una correcta estimación de costes será necesario tener en cuenta todos los recursos empleados, incluyendo recursos software, recursos materiales y recursos humanos. Se empezarán detallando los costes ocasionados por el software y otros recursos digitales.

Dado que estamos tratando con costes ficticios, se asumirá que estos serán cubiertos por una empresa que no se dedica exclusivamente a este proyecto, sino que puede desarrollar simultáneamente hasta cuatro proyectos, por lo que los gastos derivados de los recursos software (al igual que los propios recursos) pueden repartirse entre todos los proyectos.

Se considerarán como costes de recursos software los generados por cualquier programa que se haya utilizado durante algún punto del desarrollo del proyecto. Muchos de estos programas requieren un abono de carácter mensual y son usados exclusivamente para este proyecto, por lo que no se produce amortización. A continuación, se detallará el coste de los recursos software empleados:

Adobe Photoshop tiene un coste de 36,29 €/mes. Dado que este software para edición de imágenes únicamente ha sido utilizado para editar ciertas imágenes con un formato que no soportaba Paint 3D (el software para edición de imágenes principal en este proyecto), el coste total supone únicamente 36,29€.

Paint 3D es un software gratuito, por lo que no implica gasto adicional, al igual que Audacity. Tampoco Visual Studio, que es un entorno de desarrollo gratuito con el que se ha realizado toda la programación.

Unity será usado bajo la licencia “Personal”, por lo que en principio no supondrá coste alguno, sólo supondría un coste si los ingresos obtenidos por el producto superan los $10.000. Pero en este punto no es posible conocer los ingresos que se obtendrán.

Microsoft Word debe adquirirse como parte del paquete Microsoft Office, el cual tiene un coste mensual de 8,80€ por usuario para las empresas. Dado que este proyecto será realizado por una sola persona, el coste mensual es de 8,80€ durante todo el proceso de desarrollo del proyecto, lo que supone un total de 26,40€. El coste total a causa de software se recoge en la Tabla 6.1.

Programa	Objetivo	Coste
Adobe Photoshop	Diseño de elementos de la interfaz	36,29€
Paint 3D	Edición de imágenes y diseño de elementos de la interfaz	0,00€
Audacity	Edición de audio	0,00€
Visual Studio	Programación	0,00€
Unity	Motor gráfico	0,00€
Microsoft Word	Procesamiento de textos	26,40€
Coste		**62,96€**

Tabla 6.1 Costes de software

También deberán ser tenidos en consideración los costes generados por los recursos materiales empleados. Para este proyecto han sido necesarios varios dispositivos:

Un dispositivo móvil de gama media, cuyo coste en el momento de su adquisición fue de 239,95€, que será utilizado para realizar las pruebas del juego.

Un ordenador portátil, valorado en aproximadamente 1400€, que será utilizado para labores de desarrollo del proyecto.

Los equipos informáticos sí son amortizables, pues ni su adquisición, ni su uso recaerá exclusivamente sobre este proyecto, sino que pueden reutilizarse para otros fines en otros proyectos de la empresa. Dada la nueva Ley de Impuesto de Sociedades, se establecen unos periodos y porcentajes de amortización de 8 años para equipos electrónicos que pueden ser amortizables en un mínimo de 4 años con un porcentaje de amortización lineal máximo del 25% [24]. Se tomará, por tanto, como vida útil de los dispositivos 4 años, lo que supone un total de 35.040 horas.

Si tenemos en cuesta esta amortización obtenemos que el coste amortizado del dispositivo móvil es de 15,00€ y el del ordenador portátil es de 87,50€, tal como se aprecia en la tabla 6.2.

Equipo	Objetivo	Coste
Móvil gama media	Dispositivo destino de la aplicación	15,00€
Ordenador portátil	Desarrollo del proyecto y redacción de la documentación	87,50€
Coste total		**102,50€**

Tabla 6.2 Costes de equipos

Finalmente, se tendrán en cuenta los costes de personal. En este proyecto trabajará solamente una persona, el alumno, que asumirá diversos roles durante todo el proceso de desarrollo del proyecto. Entre estos roles se incluyen los de diseñador (encargado de realizar el diseño del videojuego), programador (encargado de la implementación del videojuego), supervisor de pruebas (encargado de la experimentación y las pruebas de usuario) y jefe de proyecto (encargado de supervisar todo el proyecto y elaborar la documentación necesaria).

Los sueldos base de los trabajadores se extraerán del *Convenio colectivo estatal de empresas de consultoría y estudios de mercado y de la opinión pública*, según se reflejó en el Boletín Oficial del Estado el 4 de abril de 2009, siendo este el último convenio colectivo estatal que hay disponible [25].

En la empresa ficticia que hemos supuesto cada rol lo asumirá una persona diferente, y que no necesariamente se dedican exclusivamente a este proyecto, sino que también pueden trabajar en los otros 3 proyectos en los que está trabajando la empresa. De esta forma, sus sueldos se amortizarán en cada proyecto en los que colaboren de acuerdo a la dedicación con la que trabajen en cada uno.

Además, es necesario incorporar al sueldo base de los trabajadores una cuantía extra en concepto de impuestos por la Seguridad Social que se corresponde con el 28,30% del sueldo mensual de cada empleado [26]. Los costes totales vienen detallados en la Tabla 6.3:

Puesto	Sueldo base	Retención mensual	Meses	Dedicación al proyecto	Total
Diseñador	1.539,69€	435,73€	2	25%	987,71€
Programador	1.103,04€	312,16€	2	100%	2.830,40€
Administrador de test	1.104,04€	312,44€	2	50%	1.416,48€
Jefe de proyecto	1.167,29€	330,34€	3	25%	1.123,22€
Coste total					**6.357,81€**

Tabla 6.3 Costes de personal estimados

Una vez calculados los gastos directos también es importante realizar una estimación de los gastos indirectos que se producirán durante el desarrollo del proyecto. Entre esos gastos se incluyen los gastos de funcionamiento de la empresa (luz, agua, conexión a internet, alquiler de la superficie, etc.) y todos los gastos imprevistos que no se han podido tener en cuenta durante la planificación y elaboración de los presupuestos. De acuerdo a la política de la empresa, estos gastos se estimarán como el 25% del presupuesto inicial del proyecto.

En la Tabla 6.4 se resumen todos los costes estimados del proyecto.

Recurso	Coste
Recursos software	62,69€
Recursos materiales	102,50€
Recursos humanos	6.357,81€
Costes indirectos (25%)	1.630,75€
Coste total	8.153,75€

Tabla 6.4 Costes totales

En conclusión, el presupuesto inicial estimado para este proyecto asciende a 8.153,75€.

7 - DISEÑO DE LA INTERFAZ DE USUARIO

En cualquier sistema interactivo la interfaz es un elemento clave, pues es el medio mediante el cual se comunican el usuario y el sistema, propiciando el intercambio de información necesaria para el correcto funcionamiento de la aplicación. Una buena interfaz debe ser simple e intuitiva, ceder a los usuarios el control en todo momento, ser consistente tanto consigo misma como con el conocimiento previo del usuario y proporcionar versatilidad y personalización ofreciendo al usuario alternativas para poder interactuar con la aplicación de la forma que más cómoda le resulte.

Teniendo estos principios en cuenta, y que el dispositivo en el que se realizará la interacción será un dispositivo móvil, se han diseñado unos botones lo suficientemente grandes para que su uso sea agradable, haciendo uso de metáforas para representar su funcionalidad e incorporando dos sistemas de interacción entre los que escoger para el juego. Antes de la codificación se han realizado los bocetos correspondientes a cada una de las vistas de la interfaz, que se plasmarán en este documento juntos con las vistas finales, así como el "workflow" que se presenta en la Ilustración 7.1.

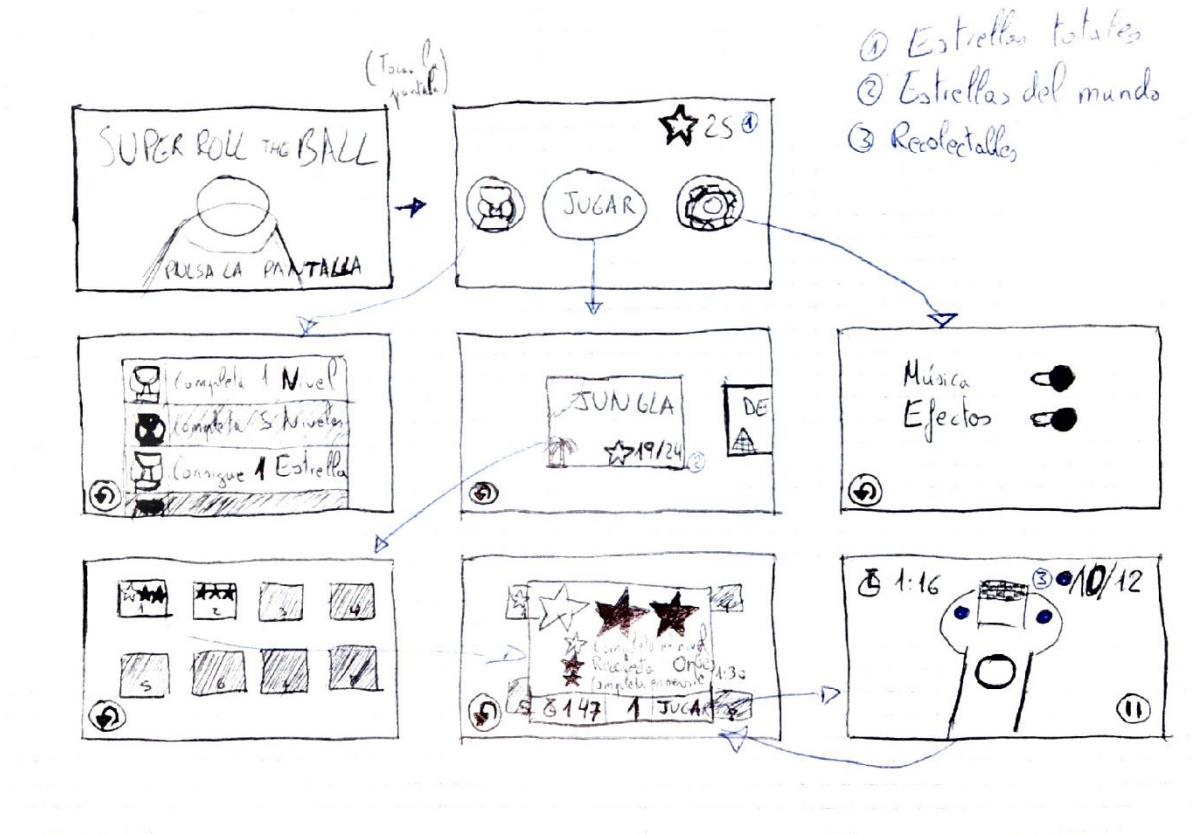


Ilustración 7.1 Workflow

7.1 - Pantalla de título

Una pantalla de inicio a modo de presentación del juego, con el título y una imagen atractiva del juego. Es lo primero que se encuentra un usuario al acceder al juego, siendo este momento, además, el único en el que esta pantalla estará disponible.

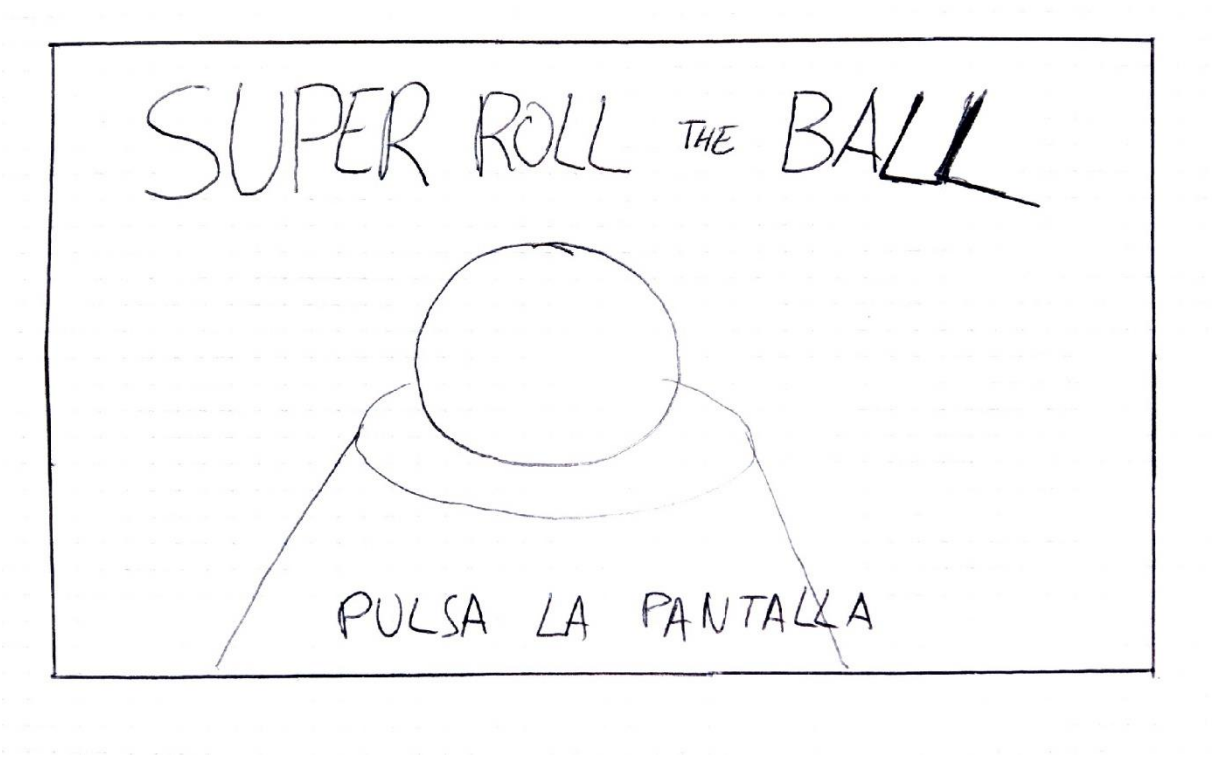


Ilustración 7.2 Boceto de la pantalla de título

7.2 - Menú principal

En el menú principal del juego se puede ver el número total de estrellas que se han obtenido y se presentarán tres botones para acceder a la pantalla de logros, a la pantalla de selección de mundos y al menú de opciones. El primer botón, que permite acceder a la pantalla de logros estará representado por el icono de un trofeo. El segundo botón, que conduce a la pantalla de selección de mundos estará representado por el icono de Play. Y el tercer botón, que redirige hasta el menú de opciones estará representado por un engranaje.

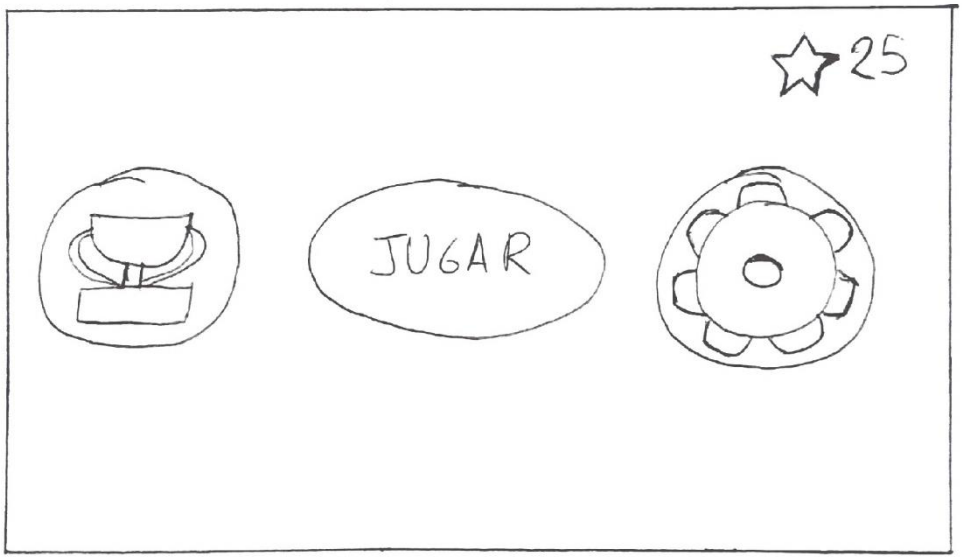


Ilustración 7.3 Boceto del menu principal

7.3 - Pantalla de logros

En esta pantalla el usuario puede visualizar los logros que ha obtenido, así como aquellos pendientes de obtención, junto con los requisitos necesarios para desbloquearlos. Esta pantalla contará con un desplazamiento vertical del panel de logros puesto que el número total de logros es superior al que puede visualizarse simultáneamente en pantalla. Los logros obtenidos se mostrarán con el fondo en blanco y el trofeo en color, mientras que los que no se hayan obtenido se visualizarán con el fondo gris y la silueta en negro del trofeo.

Ilustración 7.4 Boceto de la pantalla de logros

7.4 - Menú de opciones

En el menú de opciones se presentará la opción de modificar ciertos ajustes del juego, como el esquema de control que se quiere emplear y activar y desactivar el sonido o los efectos sonoros del juego.

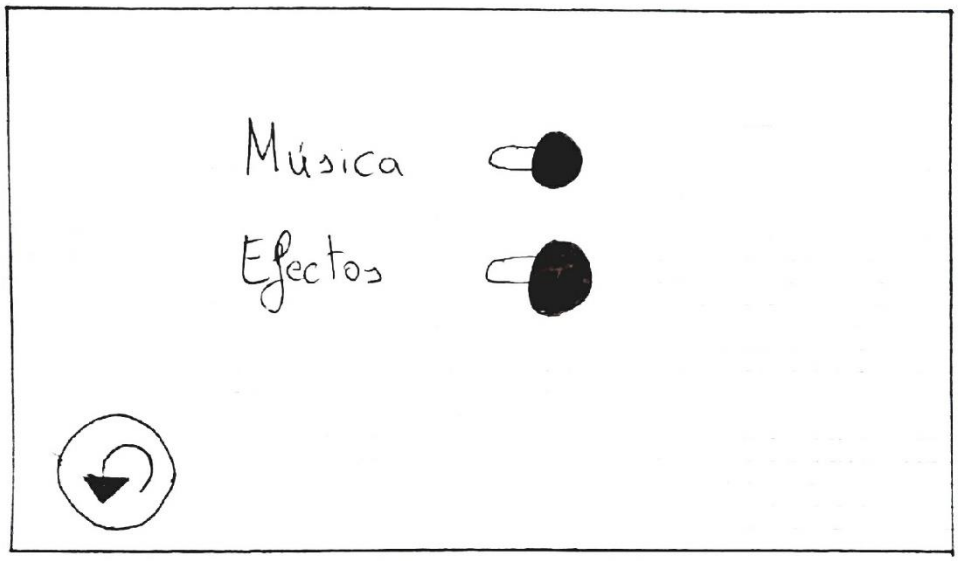


Ilustración 7.5 Boceto del menú de opciones

7.5 - Pantalla de selección de mundo

El objetivo de esta pantalla es el de seleccionar el mundo a cuyos niveles se quieren jugar. Los mundos que aún no se hayan desbloqueado no permitirán que se interactúe con ellos y, además, presentarán un candado negro superpuesto sobre ellos para atestiguarlo.

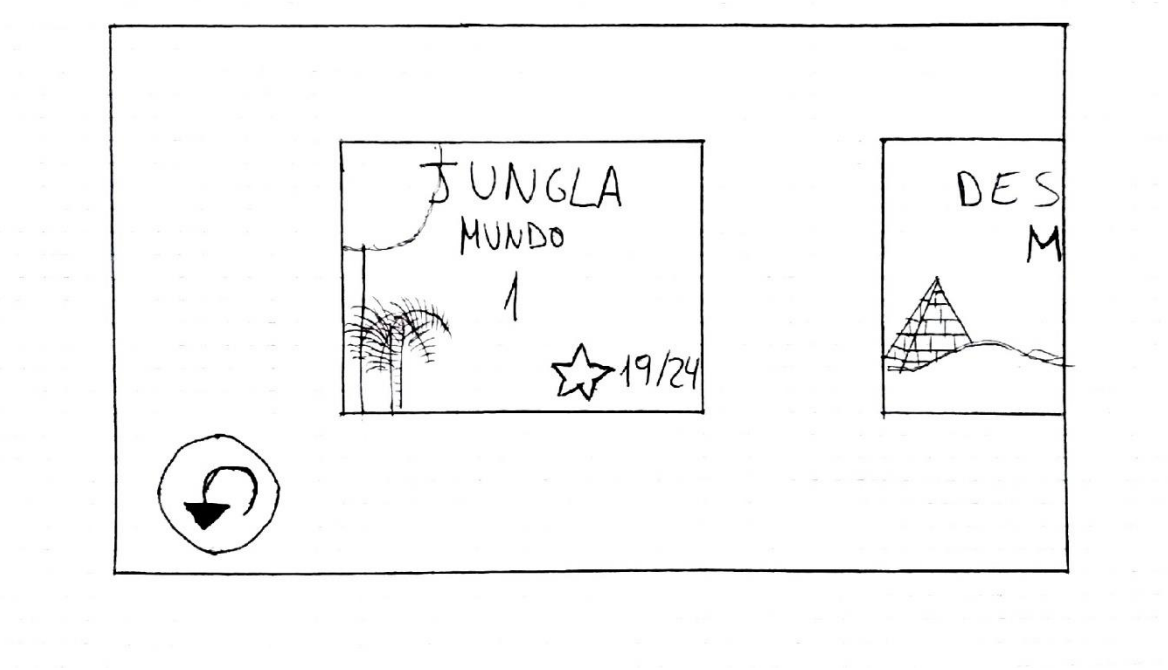


Ilustración 7.6 Boceto de la pantalla de selección de mundo

7.6 - Pantalla de selección de nivel

Cada mundo consiste en 8 niveles, y para desbloquear un nivel se tendrá que haber completado previamente el nivel anterior. Cada nivel muestra las estrellas que se han conseguido en él y el número del nivel. Los niveles bloqueados tendrán un icono de un candado sobre el botón.

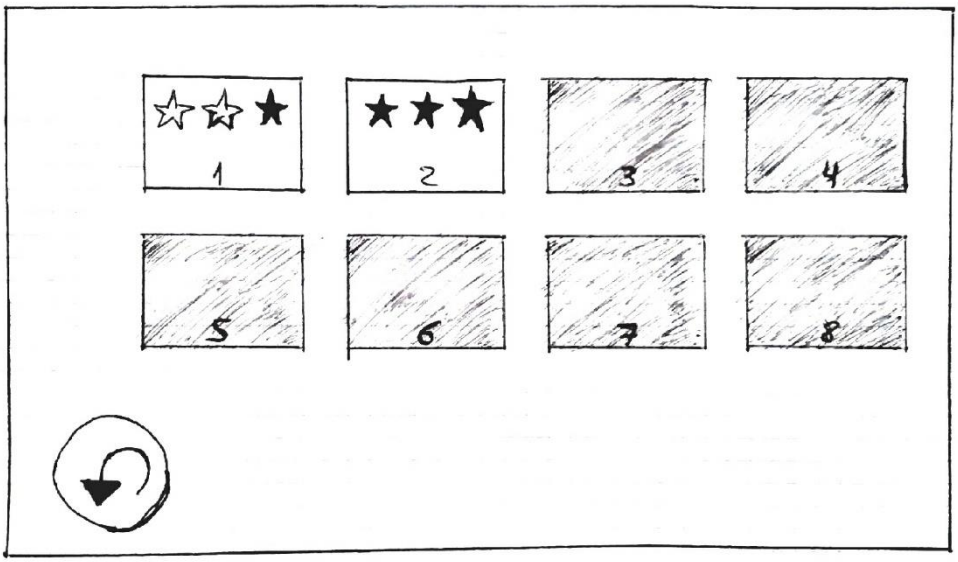


Ilustración 7.7 Boceto de la pantalla de selección de nivel

7.7 - Ventana de descripción de nivel

En esta ventana emergente se muestra toda la información del nivel seleccionado: los requisitos de obtención de cada una de las estrellas correspondientes a ese nivel, las estrellas ya obtenidas, el récord de tiempo obtenido en ese nivel, el número del nivel y un botón que permite jugar el nivel.

Ilustración 7.8 Boceto de la ventana de descripción de nivel

7.8 - Interfaz dentro del juego

Durante la partida se le mostrará al usuario el tiempo que ha pasado desde el inicio del nivel y el número de recolectables que se han recogido en el nivel y un botón para pausar el juego y acceder al menú de pausa. Durante el transcurso del juego se mostrarán algunos mensajes en el centro de la pantalla, indicando eventos como una cuenta atrás antes del inicio de un nivel o mensajes indicando que se ha finalizado la partida, ya sea a causa de una victoria o de una derrota.

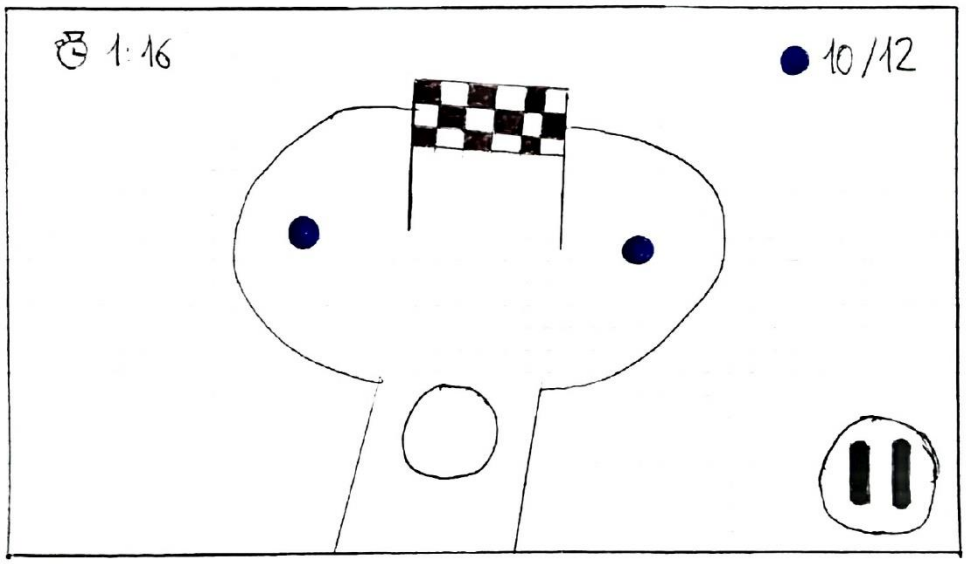


Ilustración 7.9 Boceto de la interfaz dentro del juego

8 - MECÁNICAS DE JUEGO

8.1 - Objetivos del Jugador

El objetivo del jugador es lograr el mayor número de estrellas posibles. Para ello deberá de completar cada uno de los niveles del juego y cumplir los objetivos especificados para cada una de las estrellas del nivel. Todos los niveles contarán con un total de 3 estrellas que el jugador podrá conseguir. La primera de ellas se obtendrá simplemente por completar dicho nivel, la segunda de ellas implicará recolectar todos los recolectables que se encuentran esparcidos por el nivel en un único intento, y la tercera de las estrellas se conseguirá si el jugador es capaz de completar el nivel en un tiempo inferior al especificado. El número de recolectables existentes en el nivel será siempre 3 y el tiempo a batir será específico de cada nivel.

No es necesario reunir los requisitos de las tres estrellas en un mismo intento para completar el nivel con 3 estrellas, sino que estas son acumulables entre intentos, es decir, en un intento el jugador puede plantearse el objetivo de completar el nivel y recolectar los objetos y en otro intento plantearse batir el récord de tiempo sin necesidad de atrapar todos los recolectables, consiguiendo así las 3 estrellas para ese nivel.

Inicialmente el jugador sólo tendrá desbloqueado el primer nivel, pero a medida que vaya completando niveles se desbloquearán los niveles siguientes.

8.2 - Criterios de Éxito y Fracaso

El jugador deberá llegar hasta la meta del nivel para completarlo, atravesando los obstáculos que se le presenten y evitando precipitarse al vacío. La caída al vacío supondrá la derrota del jugador en el nivel y si desea completarlo deberá empezar otra vez desde el inicio del nivel.

8.3 - Mecánica Principal

La mecánica principal de este juego es el movimiento de la bola. Este movimiento es especial, no se trata de un desplazamiento del personaje al uso, sino que para desplazar la bola el jugador realmente modifica la orientación de toda la escena, inclinándola para que la bola ruede en la dirección de la pendiente. El jugador,

por tanto, tendrá la capacidad de alterar en cada fotograma la orientación de la escena en la dirección especificada por el sistema de control que esté usando en ese momento (ya sea el control por giroscopio establecido por defecto o un control por botón deslizante en pantalla).

Estas direcciones obtenidas del medio de entrada que se esté empleando, no son absolutas (relativas al mundo), sino que son relativas a la cámara, lo que supone que inclinar el dispositivo móvil hacia delante (o deslizar hacia arriba el botón táctil) siempre repercutirá en la obtención de una mayor pendiente en la dirección en la que está mirando, y la inclinación del dispositivo móvil hacia el jugador (o deslizar el botón táctil hacia abajo) supondrá la reducción de dicha pendiente, o incluso la generación de una pendiente en la dirección contraria a la de la cámara. Estas direcciones son ponderadas por el tiempo transcurrido respecto al último fotograma, lo que garantiza un correcto funcionamiento del juego indistintamente de la capacidad del dispositivo que lo ejecute.

A medida que se completen los niveles básicos, los nuevos niveles incorporarán otras mecánicas que mantengan interesante el juego. Estas nuevas mecánicas incluyen el uso de plataformas inclinadas (rampas), plataformas móviles y obstáculos que impulsen a la bola en la dirección opuesta a la del impacto.

8.4 - Mundo y Personajes

El mundo del juego está ambientado en una jungla sobre las nubes. En mundos posteriores, se podría modificar la temática para conseguir mayor variedad en los escenarios, como, por ejemplo, mundos de hielo o de desierto.

En cuanto a los personajes presentes en el juego, sólo podemos mencionar al personaje protagonista, ya que es el único personaje en todo el juego. Este personaje es un conejo que está atrapado dentro de una bola de cristal. Los recolectables tienen forma de zanahoria para que tenga sentido que el personaje quiera conseguirlos. De cara a la versión final del juego, podría incluirse una tienda con otros personajes que sustituyan al conejo, como un mono o un pingüino.

9 - DISEÑO DE NIVELES

El juego estará compuesto por niveles independientes. Los niveles serán cortos, ya que niveles largos no son apropiados para un juego casual, especialmente en dispositivos móviles, donde las sesiones de juego son por lo general más cortas.

Los niveles estarán distribuidos en mundos. Los niveles pertenecientes a un mundo contarán con una temática concreta, que se verá reflejada tanto a nivel estético como a nivel jugable, con cada mundo incorporando nuevas mecánicas.

Cada nivel podrá tener uno de los dos sistemas de control que se han definido. En general, el sistema de control con cámara a la espalda será utilizado en niveles más rectos y el sistema de control con vista cenital será utilizado en niveles más laberínticos, con más obstáculos y giros más bruscos. Se irán alternando niveles directos y laberínticos a lo largo del juego, con niveles directos en los niveles impares y laberínticos en los pares.

Por lo general se colocarán barreras en las zonas próximas a la ubicación inicial del jugador para prevenir caídas accidentales en los primeros compases de un nivel. Por otro lado, los recolectables suponen una mecánica de riesgo-recompensa. Estos se colocarán en tramos opcionales de los niveles o en zonas menos seguras. Esto permitirá avanzar a los jugadores menos experimentados sin tener que exponerse a los desafíos adicionales que suponen los recolectables y a la vez les proporcionará a los jugadores más experimentados un mayor desafío si quieren conseguir las tres estrellas en cada nivel.

Para el diseño de los niveles se realizarán primero bocetos sobre papel para plasmar las ideas y establecer la distribución general del nivel. En el anexo “Bocetos de los niveles” se detallan los bocetos de los niveles desarrollados, así como una breve descripción de los desafíos a los que se enfrentaría el jugador en cada uno de ellos.

10 - DESARROLLO DEL PROYECTO

10.1 - Primera iteración

Durante esta iteración se ha elaborado la interfaz de acuerdo con los diseños mostrados en el apartado “Diseño de la interfaz de usuario”.

10.1.1 - Canvas

El canvas del juego utilizará el modo de renderizado en el espacio de la pantalla, concretamente mediante superposición (*Screen Space - Overlay*). Este modo permite que la interfaz se ajuste a la resolución de la pantalla de forma automática, y es independiente de la posición de la cámara. Se ha decidido incluir el canvas dentro del objeto *GameManager*, que funciona como singleton, para que el canvas no se destruya al cambiar de escenas, permitiendo que, al finalizar un nivel, la interfaz se encuentre en la misma vista desde la que se accedió al nivel. La interfaz estará formada por una serie de vistas independientes, agrupadas cada una en un *canvas group*. Este componente permite controlar la opacidad y la capacidad de interacción del conjunto de elementos que agrupa. De esta forma, el paso de una vista a otra consistirá en hacer transparente e intangible el *canvas group* activo y hacer visible y tangible el *canvas group* correspondiente a la vista que se desea visualizar.

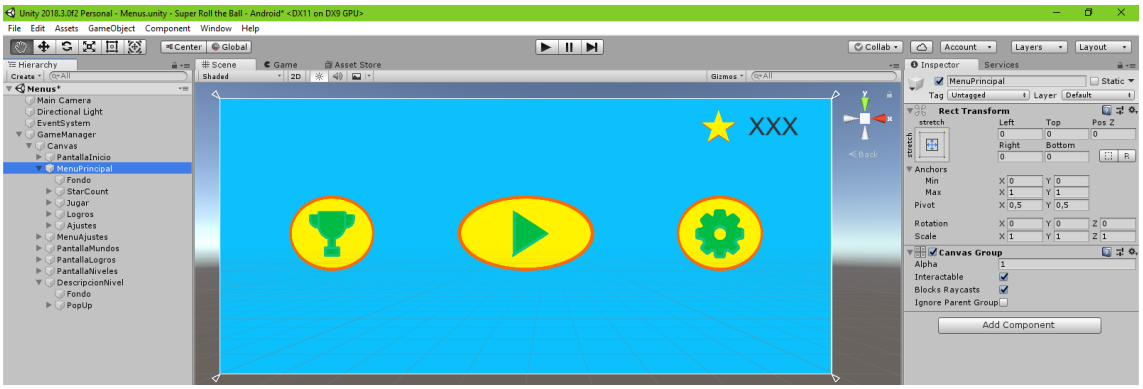


Ilustración 10.1 Jerarquía de paneles del canvas

Para gestionar estos cambios de vista se ha creado un *script* llamado *UIController*, con distintas funciones que permiten el paso desde la vista actual hasta otra concreta. Estas funciones se vincularán a los botones que generen un cambio de vista.

10.1.2 - Pantalla de título

Para navegar hasta el menú principal simplemente será necesario pulsar la pantalla. Para lograr esta funcionalidad se le ha añadido un componente *Button* (sin representación gráfica) al propio *canvas group*.

Ilustración 10.2 Prototipo de la pantalla de título

10.1.3 - Menú principal

Al pulsar sobre cualquiera de los botones presentes en el menú principal se almacena tanto el nuevo *canvas group* activo, como el *canvas group* anterior del que proceden. Y simplemente, se hace visible el *canvas group* activo e invisible el anterior.

Ilustración 10.3 Prototipo del menú principal

10.1.4 - Pantalla de logros

Para conseguir el desplazamiento vertical de los logros se ha recurrido a un *Scroll View* cuyo contenido se muestra únicamente en la zona central de la pantalla, pero su viewport abarca toda la pantalla, por lo que el deslizamiento puede producirse sobre cualquier punto de la pantalla. El contenido del viewport continúa fuera de la pantalla, lo que permite el desplazamiento, y está compuesto por cada uno de los logros, formados por imagen, texto y fondo. La visualización de cada uno de los logros se realiza por *script* para poder visualizar de forma adecuada aquellos que se han obtenido. También existe un botón para poder volver al menú principal.

Ilustración 10.4 Prototipo de la pantalla de logros

10.1.5 - Menú de opciones

Para ilustrar el proceso de modificación de los ajustes del juego, se han diseñado unos *Toggle Switch*, es decir, unos interruptores de valor ON u OFF, a partir de la barra deslizadora propia de Unity, restringiendo su rango de valores a únicamente los extremos, añadiéndole los textos de ON y OFF, modificando los colores a rojo y a verde, y añadiéndole un *script* que controle el cambio de color del manejador en función de si el interruptor se encuentra o no activado, además de permitir el uso de la barra deslizadora mediante su pulsación y no solo mediante el arrastre del manejador. Al igual que la anterior vista, cuenta con un botón para regresar al menú principal.

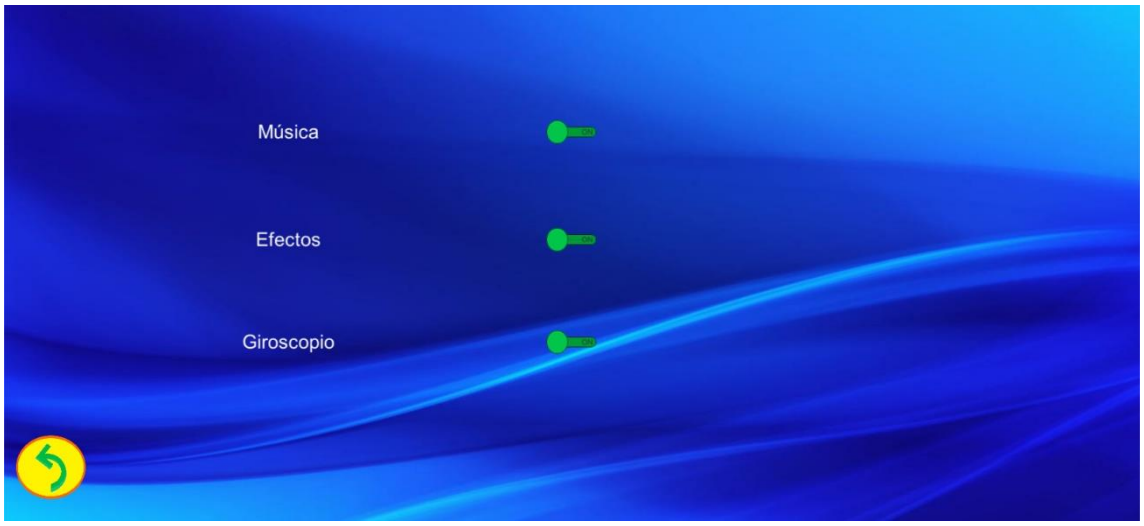


Ilustración 10.5 Prototipo del menú de logros

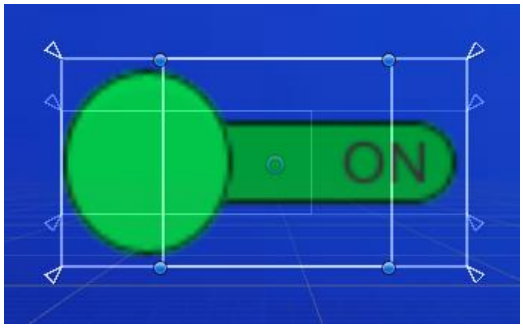


Ilustración 10.6 Detalle del interruptor a valor ON

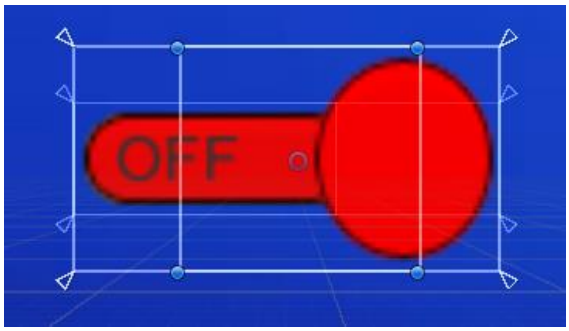


Ilustración 10.7 Detalle del interruptor a valor OFF

10.1.6 - Pantalla de selección de mundo

En esta vista también se ha recurrido a un *Scroll* View, pero en este caso para generar la posibilidad de un desplazamiento horizontal. El bloqueo y desbloqueo de los mundos, y también su correspondiente cambio estético con la eliminación del candado y el hecho de que el botón pase a ser interactuable se realiza mediante *script*.

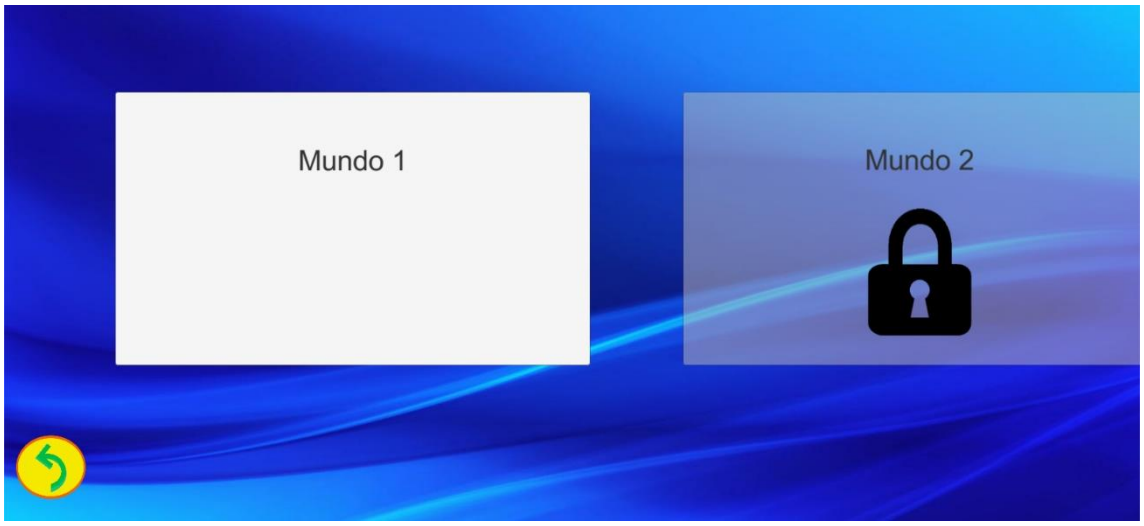


Ilustración 10.8 Prototipo de la pantalla de selección de mundo

Pese a emplear un botón para representar a cada uno de los mundos y conseguir así las animaciones propias de selección, pulsado o deshabilitado, estos botones no albergan ninguna funcionalidad. La funcionalidad se consigue mediante la captura de eventos por parte del *Event System* y su consiguiente procesamiento. Esto es así para lograr una mayor libertad de acción, pudiendo acceder al objeto sobre el que se ha pulsado, así como a sus hijos y componentes. A los elementos que queramos que sean procesados en base a los eventos recogidos por el *Event System* les tenemos que asignar un *script* (*GestorInteraccion*) para dotarles de la funcionalidad. Dicha funcionalidad variará en función de la etiqueta que tenga asignada el objeto receptor del evento. En este caso, los botones de los mundos tienen asignada la etiqueta “Mundo”, y su funcionalidad consiste en almacenar el mundo que se ha seleccionado, y preparar la pantalla de selección de nivel para que muestre la información correspondiente a los niveles del mundo seleccionado. Como es habitual, habrá también un botón para regresar al menú principal.

10.1.7 - Pantalla de selección de nivel

Como se ha podido intuir por lo anteriormente mencionado, no se tiene una vista de selección de nivel para cada mundo, sino que la única de la que se dispone se actualiza para representar la información de los niveles del mundo seleccionado.

Del mismo modo que en la vista precedente, los niveles bloqueados tendrán su botón deshabilitado y un icono de un candado sobre el nivel, y su visualización se lleva a cabo mediante un *script*. De igual forma, se recurre al *script* de gestión de eventos

de ratón para controlar la funcionalidad del botón de cada nivel. En este caso, la funcionalidad albergada para los elementos con la etiqueta “Nivel” consiste en almacenar el nivel que se ha seleccionado y mostrar una ventana emergente con la información exacta sobre el nivel. El botón de volver de esta vista nos devuelve a la pantalla de selección de mundo.

Ilustración 10.9 Prototipo de la pantalla de selección de nivel

10.1.8 - Ventana de descripción de nivel

Se trata de una ventana emergente que muestra toda la información del nivel seleccionado y para salir de esta se debe pulsar fuera de sus dominios, donde existe un panel transparente que alberga la misma funcionalidad que el botón de volver.

Ilustración 10.10 Prototipo de la ventana de descripción de nivle

10.1.9 - Interfaz dentro del juego

Al pulsar sobre el botón de pausa, el juego se detendrá o se reanudará si ya estaba detenido previamente, se modificará el icono del botón (icono de pausa si el juego está activo o un icono de reanudar si el juego está detenido) y se mostrará u ocultará el menú de pausa según convenga.

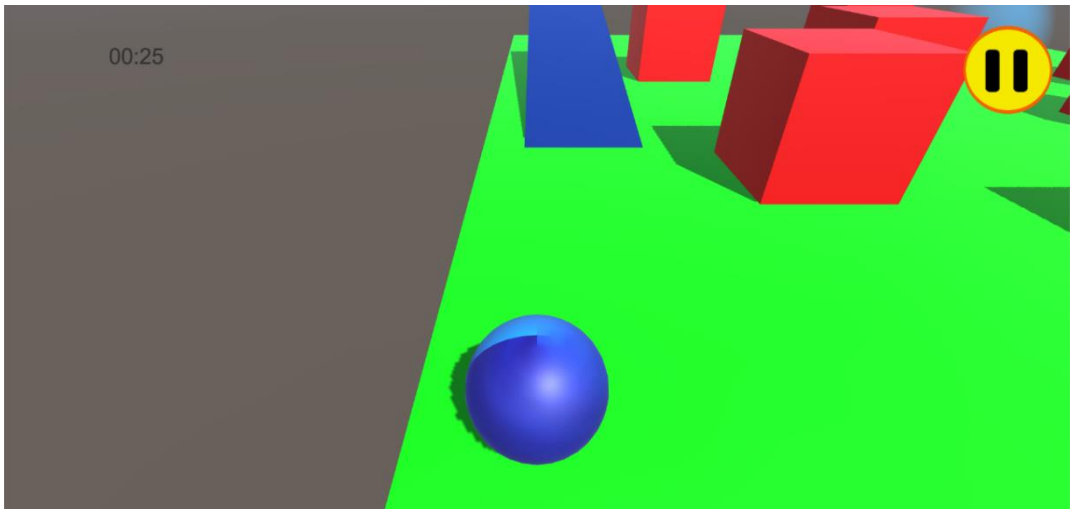


Ilustración 10.11 Prototipo de la interfaz dentro del juego

En este menú de pausa se presentan un botón para recalibrar punto de origen de coordenadas del sistema de control basado en la inclinación del dispositivo y un botón para acceder a la pantalla de opciones anteriormente descrita. También presenta un fondo gris semitransparente que recalca que lo activo es el menú y no el juego.

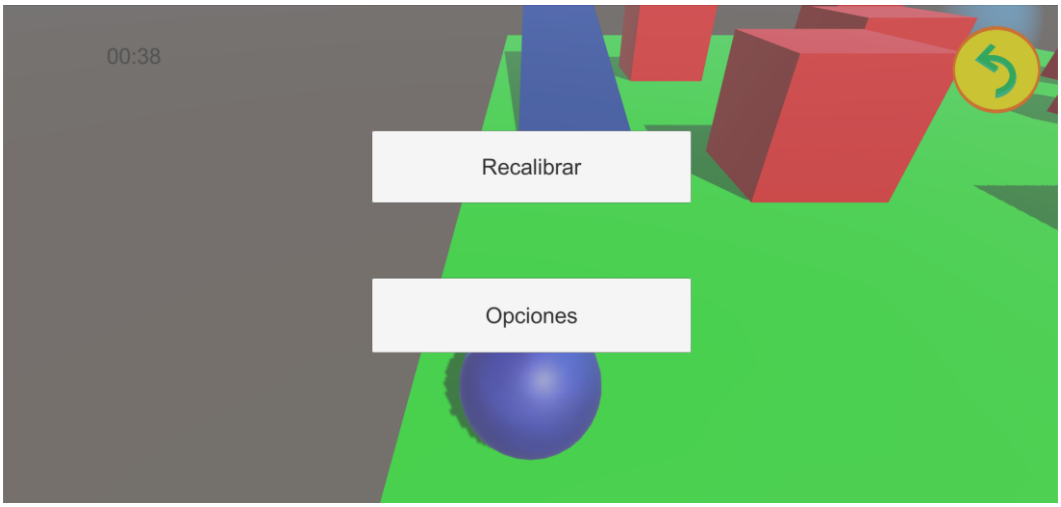


Ilustración 10.12 Prototipo del menú de pausa

10.1.10 - Nivel de prueba

Tras esto se elabora un nivel de prueba para probar el movimiento básico de la bola.

Para este movimiento básico se obtienen las lecturas del acelerómetro del dispositivo en cada fotograma de la aplicación, se les aplica un filtro de paso bajo y se ajustan para seleccionar únicamente sus coordenadas X e Y, que se hacen corresponder con las coordenadas X y Z (respectivamente) en coordenadas de mundo del vector de dirección de movimiento que se le desea dar a la bola. Esta dirección es normalizada en caso de que su magnitud sea superior a 1 y es multiplicada por *Time.deltaTime*, de forma que la magnitud de este vector de dirección sea relativa al tiempo transcurrido respecto al último fotograma, y de esta forma, la experiencia de juego sea la misma independientemente de la velocidad de procesamiento del dispositivo que la ejecute. Por tanto, en caso de que un dispositivo pueda ejecutar el juego a una mayor tasa de fotogramas que otro, se le reducirá el avance que realice la bola en cada uno de los fotogramas de forma proporcional a la propia frecuencia de generación de los fotogramas.

Tras realizar este proceso el vector de dirección disminuye notablemente su magnitud, por lo que es necesario escalarlo realizando el producto con un escalar que hará las veces de multiplicador.

Inicialmente se modificó la posición de la bola en función del vector de dirección obtenido, pero el resultado generado era demasiado rígido y la bola no rodaba. Se sopesó entonces modificar el vector de velocidad del cuerpo rígido de la bola en base al vector de dirección generado, pero los resultados seguían sin ser demasiado satisfactorios: la bola sí rodaba, pero los cambios de dirección eran extremadamente bruscos, ya que no conservaban la inercia que llevaba la bola.

Para solucionar este problema se decidió que el vector de dirección que se genera en cada fotograma fuera utilizado para suministrar una fuerza a la pelota. Para poder suministrar fuerza a la pelota y que se apliquen las leyes de físicas de Unity, es imprescindible que el objeto cuente con un *Rigidbody* y un *collider* que permita delimitar el volumen de forma aproximada de dicho *Rigidbody.* La fuerza se le aplica al *Rigidbody* de la bola en modo impulso. Con esto se obtuvo un sistema de control que permitía que la bola rodase y que tuviera presente la propia inercia de la bola

cuando se producía un cambio en el vector de dirección, es decir, en la entrada del sistema de interacción.

Esta primera iteración no era lo suficientemente grande como para suponer una primera entrega a los probadores por lo que su evaluación se realizó por los tutores y yo mismo como autor del proyecto.

Para construir este diseño de interfaz se emplearon iconos creados por el propio alumno e iconos modificados partiendo de otros obtenidos de internet, en concreto de FlatIcon [27].

10.2 - Segunda iteración

El objetivo de esta segunda iteración es implementar una serie de mejoras sobre el sistema de control primitivo que se ha definido en la iteración anterior y elaborar la lógica de la aplicación y de los niveles.

Empezando por el sistema de control, se detectaron dos aspectos que requerían una mejora. En primer lugar, se requería simular la desaceleración de la bola por el rozamiento con el suelo. Para ello fue necesaria la creación de varios *Physic Materials* [28]. Los *Physic Materials* son materiales que se aplican sobre objetos de cuerpo rígido (*Rigidbodys*) para dotarles de sus propiedades físicas como rozamiento o rebotabilidad. En concreto uno para la bola y otro para el suelo, caracterizados por tener un coeficiente de fricción tanto estática como dinámica de 1 y 2 unidades respectivamente. Esto, sumado a la propiedad “rozamiento” del *Rigidbody* de la pelota permite generar fricción entre la bola y el suelo haciendo que se vaya disminuyendo la velocidad de la bola si no se le siguen aplicando fuerzas.

El siguiente punto es justamente el contrario, ir acumulando cada vez mayor velocidad si se le aplica una fuerza de forma continua a la bola en una determinada dirección. Para ello se guarda un registro de la última fuerza aplicada y se calcula la nueva como la fuerza calculada en el fotograma actual más la fuerza que se aplicó en el anterior fotograma multiplicada por un factor de ponderación. Empíricamente se probó que en las circunstancias actuales un factor de ponderación de 0.58 ofrecía unos resultados bastante favorables, consiguiendo una aceleración suave y progresiva.

La idea detrás de esta característica era que la bola acelerase constantemente mientras se dirigiera en una dirección concreta y que esta velocidad, a su vez se reflejara en los cambios de dirección, requiriendo un mayor espacio de frenada cuanto mayor fuera la velocidad que llevaba la bola.

Después se consideró que un tipo de cámara en tercera persona en el que la cámara se situase siempre detrás de la bola podría favorecer la sensación de inmersión que se perseguía con el desarrollo de este videojuego. Uno de los problemas enfrentados al usar este tipo de cámara es que las direcciones obtenidas por el acelerómetro del dispositivo ya no serían aplicables directamente sobre la bola como fuerzas, pues la fuerza estaba expresada en coordenadas de mundo, pero la dirección de la fuerza que realmente se persigue aplicar debería depender de la orientación de la propia bola en el mundo. Es decir, las fuerzas deben aplicarse en el sistema de coordenadas de la propia bola (un vector de dirección apuntando hacia la derecha implica que la bola se desplace hacia su derecha, lo que podría suponer la izquierda en coordenadas de mundo). El proceso de transformación de coordenadas se muestra en la Ilustración 10.13.

Otro de los problemas encontrados al añadir una cámara como parte de la jerarquía de la bola, lo cual permite establecer una posición relativa de la cámara respecto de la bola, es que los ejes de coordenadas de la bola están en continua reorientación debido a que la bola se desplaza rodando. Esto provoca que la cámara de vueltas alrededor de la bola, rodando a la vez que rueda la bola.

Como solución a estos problemas se decidió no situar la cámara directamente como hijo de la bola, sino usar un objeto intermedio auxiliar, situado en la misma posición que la bola, pero cuya orientación fuera corregida en cada fotograma. La cámara, pues, se colocaba en una posición relativa a este objeto para que siempre quedase situada en la “espalda” de la bola. El vector arriba del nuevo objeto auxiliar, nombrado como “Player”, se establecía en cada fotograma para que correspondiera con el vector Y de las coordenadas del mundo. Con esto ya tendríamos solucionado el segundo problema.

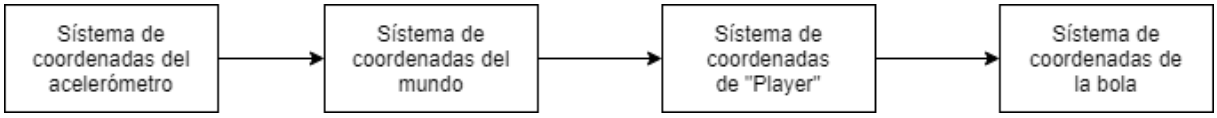


Ilustración 10.13 Transformaciones de coordenadas desde el acelerómetro hasta la bola

Además, es posible utilizar la matriz de transformación del objeto “Player” para traducir las entradas en formato de coordenadas de mundo a un formato de coordenadas locales basado en la posición y orientación de este objeto, y poder aplicar esta traducción como fuerza al *Rigidbody* de la bola.

Dado que se conserva la inercia de la bola, y, por tanto, se requiere cierto tiempo y espacio para realizar un cambio de dirección, se consideró necesario suministrar al jugador de un “feedback” que le indicase que sus acciones estaban teniendo repercusión sobre el juego. Por ejemplo, si el jugador intenta desplazarse en la dirección opuesta a la que está llevando la bola en este momento, se precisa primero frenar la bola antes de impulsarla en la otra dirección, y en los primeros instantes puede no verse claramente el cambio en la velocidad que lleva la bola. Esto propició un cambio en la filosofía del desplazamiento, pasando de aplicar manualmente fuerzas sobre la bola a considerar la posibilidad de inclinar el plano sobre el que se desplaza la bola y dejar que se apliquen las físicas de Unity para planos inclinados.

Para lograr un giro más fluido de la escena, y en concreto evitar que el cambio de orientación genere cambios abruptos de pendiente en las zonas más alejadas del centro de la escena, se ha decidido que el giro se produzca centrado sobre la posición de la bola.

Cabe destacar que la cámara se orienta en la dirección del movimiento de la bola. Mediante la orientación adecuada del objeto “Player” es posible apreciar la inclinación del escenario y facilitar la sensación de aceleración y desaceleración. La orientación concreta se calcula estableciendo como vector Forward (dirección que enfrenta), la proyección de la velocidad que tiene actualmente el *RigidBody* de la bola sobre el plano XZ en coordenadas de mundo (esto se hace para que el “Player” se oriente en la dirección del movimiento, pero permitiendo que se aprecie la inclinación del escenario mencionada anteriormente).

Para evitar que se produzcan orientaciones de la escena no deseadas, los cambios que han de producirse en cada fotograma se realizan primero sobre una transformación auxiliar, obtenida a partir de la transformación real de la escena en ese instante. En caso de que el ángulo entre la proyección de la vertical de la nueva escena resultante sobre el plano definido por el vector Forward del objeto Player y el vector Y del mundo no supere los 35 grados se podrá realizar la reorientación horizontal de la escena, es decir, se podrá realizar el giro de la escena respecto al eje

definido por el vector Forward de Player, y en caso de que supere los 35 grados ese giro no se realizará. Del mismo modo, sólo se permitirá realizar el giro sobre el eje definido por el vector Right de Player si el ángulo entre el vector formado por la proyección de la vertical de la escena sobre el plano definido por el vector Right de Player y el vector Y del mundo es inferior a 35 grados o si el cambio en la orientación de la escena que se desea aplicar prevé reducir dicho ángulo. Además, en el caso de que esta última comprobación sea falsa, se procederá a añadir una fuerza creciente cada fotograma sobre la bola en la dirección del movimiento con el objetivo de generar una mayor aceleración. En la ilustración 10.14 se muestran los ejes Forward, Right y Up de la bola.

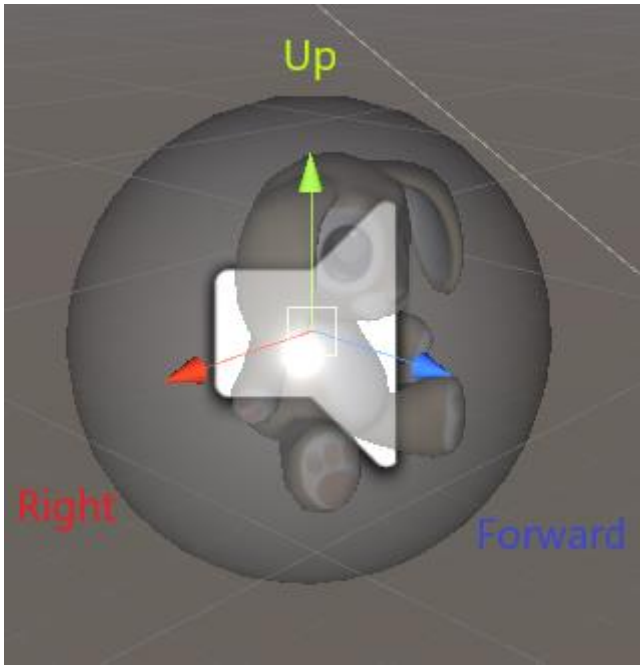


Ilustración 10.14 Ejes Forward, Right y Up de la bola

Finalmente, con la intención de mitigar los giros bruscos de cámara ocasionados por respectivos cambios abruptos de la dirección de movimiento de la bola, se ha diseñado un sistema por el cual los 20 fotogramas posteriores a la detección de un giro brusco se ajuste manualmente la cámara mediante una interpolación entre la dirección anterior al cambio brusco y la posterior a este. Sin embargo, este ajuste suave de cámara sólo se produce si la bola lleva una velocidad superior a un límite establecido, de forma que, si se produce un cambio de dirección brusco, pero la velocidad es pequeña no se produzca ni el ajuste suave de cámara ni la reorientación por defecto de “Player” en función de la velocidad, sino que se

mantenga la última orientación calculada. Esto último debe realizarse concienzudamente, manteniendo en todo momento el *Quaternion* correspondiente a la última modificación de la orientación, ya que, si no se fuerza a que mantenga la misma orientación [29], esta se modificará debido a la rotación de la bola como se ha descrito anteriormente y que llevó a considerar la reorientación del "Player" en cada fotograma.

Aún con las medidas tomadas para favorecer un movimiento de cámara suave, todavía se detectaron situaciones en las que se producía un comportamiento indeseado. Estas situaciones eran principalmente los rebotes causados por chocar con obstáculos cuando la bola adquiría cierta velocidad, pues se producía un cambio de dirección muy brusco y no siempre se reducía lo suficiente la velocidad como para que actuasen las otras medidas de suavizado de movimientos de cámara. Como solución a este problema se creó un *Physic Material* anti-rebote que se establece como material del componente *Collider* de los obstáculos generados. Este material tiene un coeficiente de fricción de cero, tanto estática como dinámica y un coeficiente de rebote de cero, permitiendo que la bola no se frene si contacta de forma oblicua y que tampoco salga despedida, repelida por la fuerza del choque.

Durante esta segunda iteración también se ha trabajado en desarrollar la lógica del juego, incluyendo el *GameManager* para mantener la lógica de toda la aplicación y realizar ciertas labores de cohesión entre partes, el *LevelController* para proporcionar la lógica dentro del propio nivel de juego y todos los demás *scripts* utilizados para conseguir que se cumplan las normas del juego o que sirvan de clases auxiliares, como las utilizadas para lograr la persistencia de datos del juego.

El *GameManager* funciona como un *Singleton [30]*. El patrón *Singleton* es una convención para asegurar que una clase tenga únicamente una instancia y proveer un punto de acceso global a esta instancia. De esta forma nos aseguramos de que la clase *GameManager* tenga una única instancia y que desde cualquier punto de la aplicación se pueda acceder a dicha instancia, independientemente de la escena en la que se esté y permita el paso de información entre escenas al mantenerse invariable y no destruirse con los cambios de escena.

Esta clase tiene variables que controlan qué mundo y qué niveles son los que están seleccionados en cada momento, y permite a cualquier otro *script* que lo necesite acceder a esta información, que puede ser útil para cargar el nivel correspondiente, mostrar la información correcta del mundo y niveles seleccionados y

favorecer la concesión de estrellas y la obtención de logros. También almacena una matriz con la información de todos los niveles y los datos de progresión, que es una clase que almacena datos del juego entre los que se incluyen los logros obtenidos, el total de estrellas, el total de niveles completados o de récords de tiempo superados. También cuenta con las clases que permiten traducir a binario estos datos y almacenarlos en fichero o extraerlos de uno. Además, almacena las opciones de control como el volumen de la música, los efectos o si está activada la opción de sustituir el control por giroscopio por un control de botón táctil deslizante. Las relaciones de esta clase se pueden comprobar en el diagrama mostrado en la Ilustración 10.15.

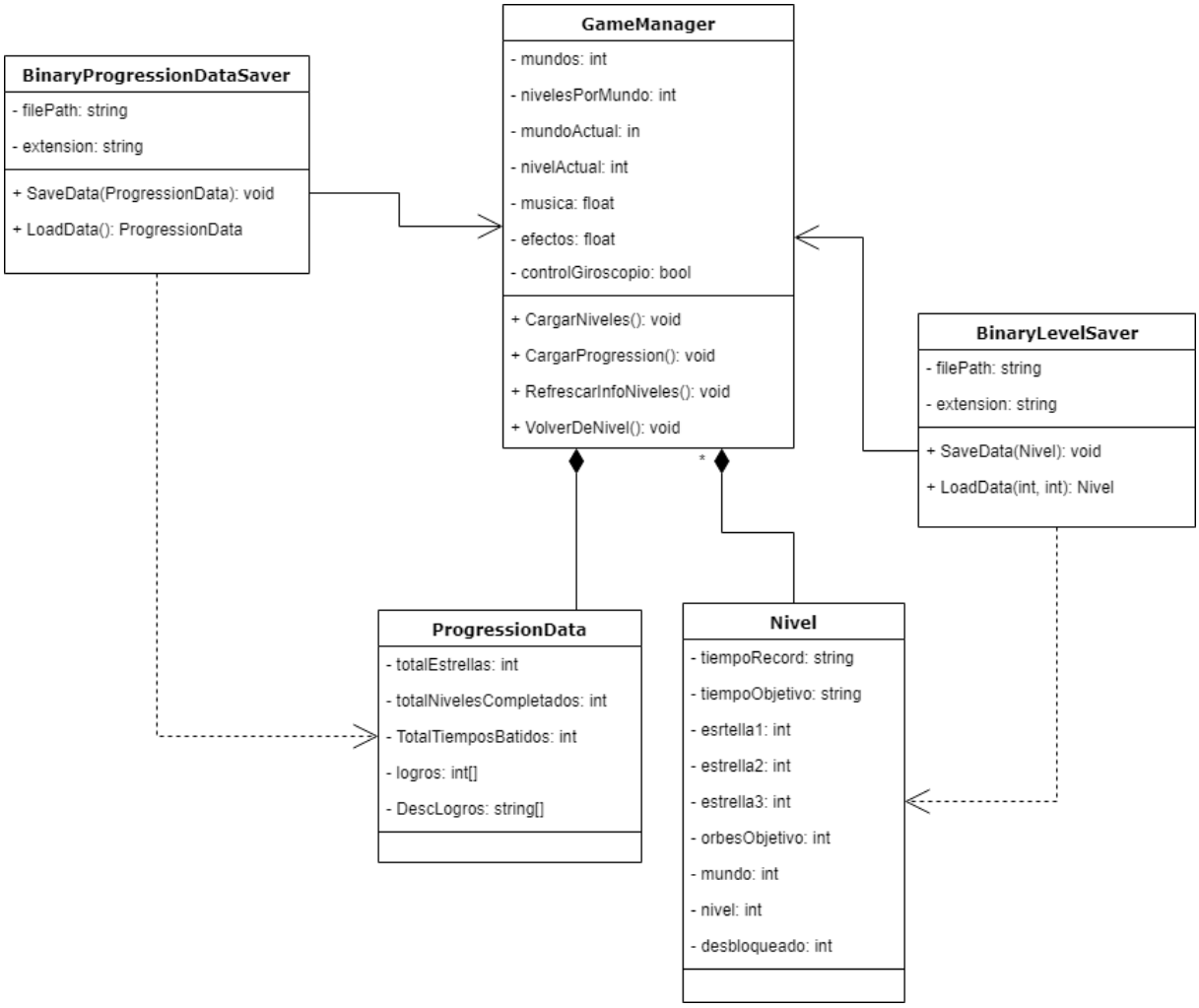


Ilustración 10.15 Diagrama de clases de GameManager

Al inicializar la instancia de *GameManager* se cargan desde fichero los datos de los niveles y de progresión en caso de que existan, y si no es así los crea y los

almacena. Posee un método para comprobar si se ha obtenido algún logro cuando se completa un nivel y se actualiza la información del propio nivel y los datos de progresión para que puedan ser mostrados actualizados por la interfaz. A través de este *script* se incrementa el número de estrellas o niveles completados, se desbloquean los nuevos niveles cuando se supera el último desbloqueado y se proporciona acceso a toda la información que recopila (datos de los niveles, nivel y mundo actual, valores de los ajustes, datos de progresión, etc.).

El *script LevelController*, por su parte, sólo existe mientras se está jugando algún nivel y permite centralizar la lógica del nivel. Se encarga de llevar el control del marcador de tiempo, del número de recolectables obtenidos en el nivel, de pausar el juego y de mostrar los mensajes que se produzcan durante el juego. También se encarga de realizar una pequeña animación al inicio del nivel para hacer que descienda la bola desde arriba antes de comenzar el nivel unido a una cuenta atrás y de presentar el menú de pausa implementando su funcionalidad, entre la que se incluye el acceder al menú de opciones, salir y reiniciar el nivel. A la hora de salir del nivel se comprueba si se ha superado el récord de tiempo anterior y si se han cumplido las condiciones necesarias para obtener alguna estrella, en cuyo caso se comunica con el *GameManager* para llevar a cabo las acciones necesarias y finalmente cambia a la escena principal de los menús.

Dentro del propio nivel se han asociado ciertos *scripts* a objetos del juego, cuyo contenido se ejecuta cuando se acciona el *Trigger* del *collider*. A los *colliders* que les asignamos a los objetos del juego podemos definirlos o no como *Triggers*. En caso afirmativo en lugar de que dicho *collider* se comporte como una barrera sólida, es posible delimitar un volumen que lleva a la ejecución de un trozo de código cuando otro *collider* colisiona con este. Esta funcionalidad se ha empleado para detectar con un plano situado debajo de la escena si la bola se cae. Cuando esto sucede se muestra un mensaje indicando que el jugador ha perdido y posteriormente se reinicia el nivel. También permite detectar si la bola ha llegado a la meta, en cuyo caso se muestra un mensaje indicando que el jugador ha completado el nivel y se finaliza el juego.

Tras finalizar esta iteración se proporcionará a los probadores una copia de la versión inicial del juego para conocer sus primeras impresiones sobre el juego y su interfaz, especialmente sobre el sistema de control del juego que se ha considerado como una parte fundamental del juego y en la que se ha destinado un importante

esfuerzo. Se ha obtenido información de los probadores con una pequeña encuesta orientada a los apartados mencionados anteriormente.

10.3 - Tercera iteración

A partir de los comentarios de las encuestas elaboradas por los probadores se ha decidido facilitar en todo lo posible que el control por giroscopio sea más cómodo de usar. Para ello se ha decidido incorporar una opción al menú de pausa del juego que permita recalibrar la posición de origen del giroscopio para este tipo de control. En segundo lugar, aportar al jugador más información sobre la inclinación actual del giroscopio mediante una especie de radar que muestre el valor de inclinación junto con su espectro de valores posibles, y que pueda de esta forma decidir mejor sus acciones. Este radar se implementará en la quinta iteración. Y, en tercer lugar, tener en cuenta la dificultad presentada al producirse impactos con los objetos que generan cambios bruscos en la dirección de desplazamiento de la bola para el diseño de los niveles. Concretamente se ha valorado la posibilidad de limitar este sistema de control para niveles más alargados y enfocados a la velocidad, con giros menos bruscos, y emplear el sistema de control que se desarrolló inicialmente para los niveles que, como el nivel de prueba presenten un mayor número de posibilidades de colisión y que precisen de cambios de dirección más pronunciados.

En cuanto a la interfaz, se va a mantener una estructura similar, a la vez que se mejora la estética empleando un framework de terceros, en concreto Marklight [31]. Este framework es gratuito y está disponible para su descarga a través de la tienda de Assets de Unity.

El uso de este framework ha requerido crear dos archivos de código por cada pantalla de la interfaz. El primero de ellos se corresponde a un archivo *XUML* (eXtended Unity Markup Language) [32], que es un lenguaje similar a *HTML* que permite configurar la estética de la pantalla, posicionando los botones, imágenes y demás elementos. El segundo de estos archivos es un *script* en *C#*, que debe tener el mismo nombre que el anterior y que se encarga de implementar la lógica de la interfaz en esa pantalla.

Para empezar, se debe crear un objeto en la escena de Unity al que se le asignará un *script* llamado *View Presenter* que permite construir en Unity, mediante

sus elementos de interfaz, las pantallas definidas en los archivos *XUML* que hayamos creado. En dicho objeto debemos modificar ciertos atributos del *script* para indicar cuál será la pantalla de principal de la interfaz y qué tema se va a emplear, en este caso, se utilizará el tema “Toon”. La pantalla principal de la interfaz deberá de incluir el *Canvas*, que se corresponde en MarkLight con *UserInterface* y que incluye un *script* que permite realizar ciertas acciones adicionales sobre su contenido necesarias para el uso de este framework, el *EventSystem*, y la que será la primera pantalla propiamente dicha de la interfaz (En nuestro caso el menú de inicio: *Menu*).

En la vista *Menu* se define un *ViewSwitcher*, que es un elemento de MarkLight que permite cambiar la vista que se está visualizando en una región, por otra perteneciente a un conjunto de vistas previamente especificado. Al *ViewSwitcher* se le puede añadir una animación para que la realice en la transición entre dos vistas. Se ha optado por realizar un escalado y una rotación para dar la sensación de que la nueva vista aparece en pantalla acercándose desde el fondo. Estas transiciones se producen de acuerdo a la escala de tiempo del juego, si el juego está detenido no se finalizará nunca la transición. Esto supone un problema cuando se quiere acceder a la pantalla de opciones desde el menú de pausa, en el que obviamente, el tiempo está detenido. Para solventar este problema se han utilizado dos *ViewSwitchers* diferentes.

En el primero se han incluido las pantallas del menú principal, opciones, logros, selección de mundos y selección de niveles, mientras que en el segundo sólo tenemos la interfaz “in-game” (la interfaz que se muestra mientras se está jugando a los niveles del juego) y el menú de opciones. El primer *ViewSwitcher* sí realizará animaciones entre las transiciones, pero el segundo no. Puesto que ambos se visualizarían a la vez en la misma región de la pantalla, se incorpora una nueva pantalla totalmente vacía que permitirá mostrar el contenido del otro *ViewSwitcher* sin la interferencia del contenido de su propio *ViewSwitcher*.

10.3.1 - Menú principal

En la vista *MainMenu* se crea el menú principal del juego, formado por una ventana con un lazo y un título en él, y dentro de la ventana tres botones: “Jugar”, “Opciones” y “Logros”, que llevan a las pantallas de selección de mundos, opciones y logros respectivamente. También se muestra un contador con las estrellas acumuladas hasta el momento en la esquina superior derecha. Cada botón está enlazado con un método que ha sido definido en el archivo *C#* y que se referencia por

medio del nombre del método. En este caso, cada botón tiene un método asociado, que simplemente cambia la vista mostrada por el *ViewSwitcher*.

En el caso del botón de opciones el cambio de vista incluye un envío de datos a la nueva vista, concretamente un booleano para indicar que se ha accedido al menú de opciones desde el menú principal y no desde el de pausa, y así modificar el destino del botón de vuelta. Existe un método llamado *Initialize()* que se ejecuta la primera vez que se va a cargar la vista (similar al *Start()* de Unity), en el que se asocia el *ViewSwitcher* existente a un atributo de la clase para poder utilizarlo. Por último, también se crea un método llamado *ViewActivated()* que se ejecuta cada vez que la vista vaya a ser mostrada en pantalla. En este método se actualiza el contador de estrellas obtenidas que se mostrará.

Ilustración 10.16 Menú principal

10.3.2 - Menú de opciones

En la vista *Options* se muestran las opciones de configuración presentes en el juego, que son básicamente el volumen de la música, el volumen de los efectos de sonido y si se quiere o no utilizar la inclinación del dispositivo como método de control. Las dos primeras opciones son configurables mediante un deslizador que permite ajustar el volumen de cada uno de forma independiente, mientras que el control por inclinación está representado por un *checkbox*. El valor de todos estos parámetros está enlazado con los atributos de la clase en *C#* de la vista. De esta forma, es posible

representar gráficamente el valor real de los atributos de la clase, que son los mismo que se emplean en la lógica del juego y modificarlos a gusto del usuario. Para reflejar los cambios que se han producido en la vista (y al mismo tiempo en los atributos de su clase) en los atributos presentes en el *script GameManager*, que es donde estos se aplican, se utilizan los denominados *ChangeHandler*. Estos son unas funciones asociadas a ciertos atributos de la clase que se ejecutan cada vez que se produce un cambio en el valor de estos, ya sea producido por un cambio en los elementos de la interfaz o a consecuencia de un trozo de código.

Se tiene un booleano para determinar si la pantalla de opciones se ha cargado mientras el juego estaba en el menú principal o en el de pausa en el medio de la partida. Y de acuerdo al valor de este atributo el resultado de pulsar el botón de atrás, situado en la esquina inferior izquierda y representado como una flecha hacia la izquierda, podrá diferir entre utilizar un *ViewSwitcher* para volver al menú principal o utilizar el otro para regresar a la interfaz “in-game”. Este atributo booleano es actualizado con la función *ViewActivated(bool data)*, que como se explicó anteriormente se ejecuta cada vez que la vista se va a visualizar y cuyo parámetro es enviado desde la función de cambio de vista de otras pantallas. En su inicializador se cargan y asocian los dos *ViewSwitchers* y los últimos valores registrados para las opciones de configuración: volumen de la música, volumen de los efectos de sonido y uso de control por inclinación.

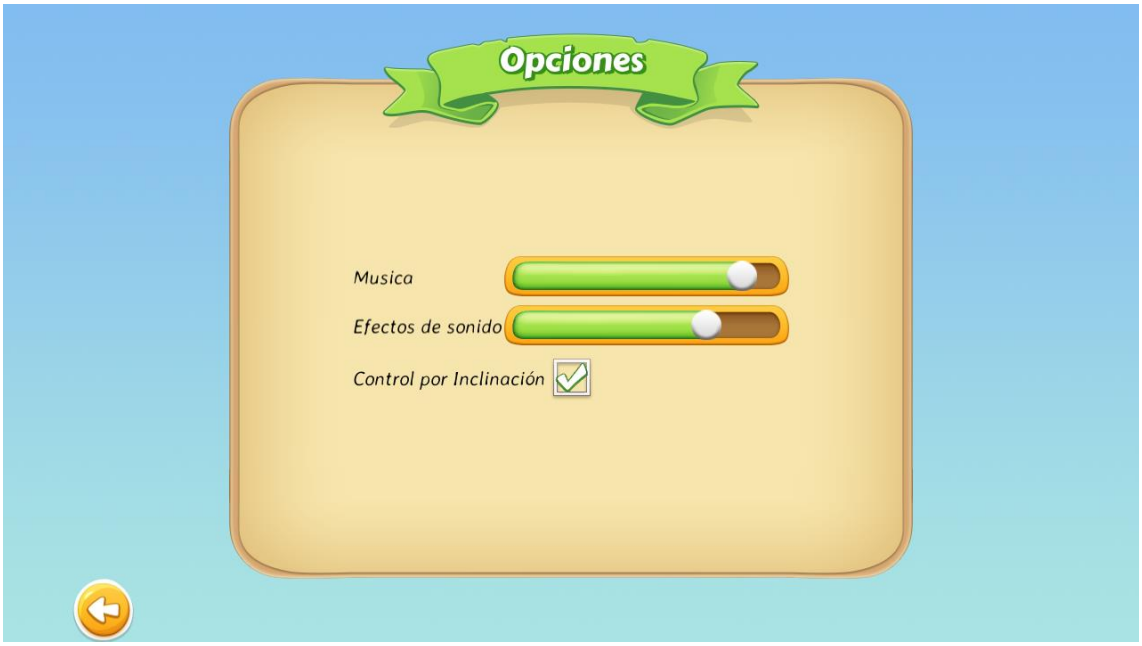


Ilustración 10.17 Menú de opciones

10.3.3 - Pantalla de logros

En la vista *Achivements* se muestran todos los logros posibles del juego, tanto los bloqueados como los desbloqueados en un formato de lista. Cada uno de los logros está formado por una pequeña ventana, un texto y la imagen de un trofeo. Si el logro está desbloqueado, la imagen del trofeo tendrá un color amarillo, mientras que, si el logro no está desbloqueado, se visualizará una silueta en negro del trofeo, el panel que lo engloba se mostrará gris y aparecerá un candado en la parte superior derecha del panel.

Para representar todos los logros posibles y la posibilidad de que cada logro esté bloqueado o desbloqueado, se recurre a una lista con dos plantillas de elementos posibles, una para los logros desbloqueados y otros para los bloqueados. En ambos casos el texto del logro será el del elemento de la lista que se esté procesando en ese momento, pues en MarkLight, las listas funcionan proporcionándole al elemento gráfico de la lista un atributo de la clase de la vista que contenga los elementos que se quieren procesar, y para cada elemento se le asigna una plantilla basándose en un ID, que se extrae mediante el método *GetTemplateID()*, el cual debemos definir para que asigne un ID a cada logro según las condiciones que se deseen. En este caso se otorgará un ID de “Locked” si el logro está bloqueado o de “Unlocked” si el logro está desbloqueado.

Cada vez que esta vista vaya a ser visualizada se activa el método *ViewActivated* para actualizar los valores de los atributos y recogerlos en una *ObservableList* para permitir su procesamiento por parte del elemento lista de MarkLight. Al igual que se ha mencionado en otras vistas, en el inicializador de esta clase se asocian los *ViewSwitchers* y se crea una lista vacía. También se tiene un botón para volver atrás en la esquina inferior izquierda que vuelve a mostrar el menú principal.

Ilustración 10.18 Pantalla de logros

10.3.4 - Pantalla de selección de mundo

La vista *WorldSelect* se comporta de manera similar a la anterior, mostrando en una lista los logros. Nuevamente existen dos plantillas de elementos de la lista, una para los mundos bloqueados y otra para los desbloqueados. Cada elemento de la lista presente en la *ObservableList* de la clase será asignado a una u otra plantilla en función de su ID. Dicho ID, es obtenido mediante la función *GetTemplateID()* que asigna a la clase *World* el string “LockedWorld” si está bloqueado o “World” si está desbloqueado. Los elementos de la lista que sean asociados a la plantilla de mundos bloqueados tendrán un tono gris, un candado en la esquina superior izquierda y no será interactuables. En cualquiera de los casos mostrarán una etiqueta que indique de que mundo se trata.

Ilustración 10.19 Pantalla de selección de mundo

En el archivo *C#* de esta vista, se ha definido el método *SelectWorld()*, que se ejecuta al pulsar sobre un mundo desbloqueado y que almacena el mundo seleccionado y se la transmite al *GameManager*, para que de esta forma se muestre la información correspondiente a los niveles de ese mundo, y posteriormente se realiza un cambio de vista con el *ViewSwitcher*. En el método *ViewActivated()*, que se ejecuta cada vez que la vista se vaya a visualizar, se obtiene el listado de los mundos para que estén actualizados. En la inicialización de la vista se asocia el *ViewSwitcher* y se crea la *ObservableList*. También se tiene el método para volver al menú principal cuando se pulsa sobre el botón atrás.

10.3.5 - Pantalla de selección de nivel

En la vista *LevelSelect* se tiene una lista para representar todos los niveles existentes dentro de un mundo con sus correspondientes puntuaciones. Los elementos de la lista se clasifican entre cinco plantillas de elementos. Las cuatro primeras se corresponden con niveles desbloqueados con las distintas posibles puntuaciones (cero, una, dos o tres estrellas) plasmadas mediante su imagen correspondiente, mientras que la última plantilla permite representar los niveles bloqueados, con el panel del nivel en gris, un candado y sin la posibilidad de interactuar con él. En todos los casos se muestra el número del nivel al que corresponde. La clasificación de los elementos de la lista entre las distintas plantillas

se realiza mediante un ID que puede tomar cinco valores distintos. Si el nivel está bloqueado su ID será “Locked”, y si no su ID será la concatenación de “Stars” con el número de estrellas obtenidas en ese nivel.

Ilustración 10.20 Pantalla de selección de nivel

En el archivo en *C#* de esta vista se tiene una *ObservableList* con los elementos que se procesarán en la lista, los cuales son actualizados cada vez que la vista vaya a ser visualizada mediante el método *ViewActivated()*. En la inicialización de la clase se crea una lista vacía y se asocia el *ViewSwitcher*. El método asociado al botón de volver cambia la vista representada por el *ViewSwitcher* para que sea la pantalla de selección de mundos.

Cuando se pulsa sobre un nivel desbloqueado se ejecuta un método en el que se registra el nivel sobre el que se ha pulsado como el nivel seleccionado para que la información posterior se muestre en relación a este nivel. También se crea y activa una nueva vista, que se superpondrá a la existente y proporcionará información más detallada sobre el nivel que se había seleccionado. Esta vista se almacena como atributo para que pueda ser accedida posteriormente para su destrucción mediante el método *DestroyPopUp()*, y así evitar sobrecargar de ventanas inactivas si no se borran. También incorpora un método llamado por el *GameManager* cuando se vuelve a la escena de los menús después de haber jugado algún nivel. Este método llamado *VolverDeNivel()* cambia la vista mostrada por los dos *ViewSwitchers* para que se vea

la pantalla de selección de nivel sin interferencias del segundo *ViewSwitcher* (cambiando este a la vista vacía) y crea el *PopUp* de la pantalla de descripción de nivel, para poder ver así el desempeño en el nivel (nuevo tiempo récord de nivel y nuevas estrellas obtenidas).

10.3.6 - Ventana de descripción de nivel

En la vista *LevelDescription* primero se muestran las estrellas obtenidas y a continuación se explican los requerimientos para obtener cada estrella, junto con una estrella vacía o normal dependiendo de si la estrella ya ha sido obtenida. La primera estrella simplemente requiere completar el nivel, la segunda recoger un número determinado de coleccionables y la tercera batir una marca de tiempo al completar el nivel. Además, en la parte inferior de la ventana se muestra el mínimo tiempo en el que se ha completado el nivel, el número del nivel y un botón para jugar al nivel. En este caso el botón de volver lo que hace es llamar a un método de la clase *LevelSelect* para destruir la ventana que se había generado.

En la clase de esta vista contamos con el método *ViewActivated()* para actualizar la información sobre el nivel antes de mostrarla. En él, se actualizan tanto el estado de las estrellas del nivel, como el récord de tiempo para el nivel. En función de si la estrella está o no desbloqueada se asignará una cadena de caracteres con la ruta de la imagen que le corresponda, la cual será utilizada en la vista para acceder a la imagen y visualizarla. En la inicialización de la clase se establece por defecto las estrellas como no obtenidas y se asocian los dos *ViewSwitchers* que se habían definido anteriormente. Cuando se pulsa sobre jugar se ejecuta un código en el que se oculta esta ventada, se cambia la vista del primer *ViewSwitcher* a una vacía y la del segundo a la interfaz dentro del juego, y finalmente se carga la escena que contiene el nivel que se ha seleccionado.

Ilustración 10.21 Ventana de descripción de nivel

10.3.7 - Interfaz dentro del juego

En la vista de MarkLight de *InGame* sólo está presente un botón en la esquina superior derecha que abre un menú dentro del juego y el juego se pausa. Otros elementos de la interfaz del juego, como el contador de tiempo, los mensajes que se muestran en pantalla o el botón deslizante se encuentran dentro de un canvas estándar de Unity para facilitar la comunicación con los mismos. A nivel de código, sólo se tiene la función que se ejecuta al pulsar el botón del menú, que consiste en crear un *PopUp* con el menú de pausa y pausar el juego o en destruir esa vista si el botón se ha pulsado cuando ya se estaba visualizando el menú y reanudar el juego.

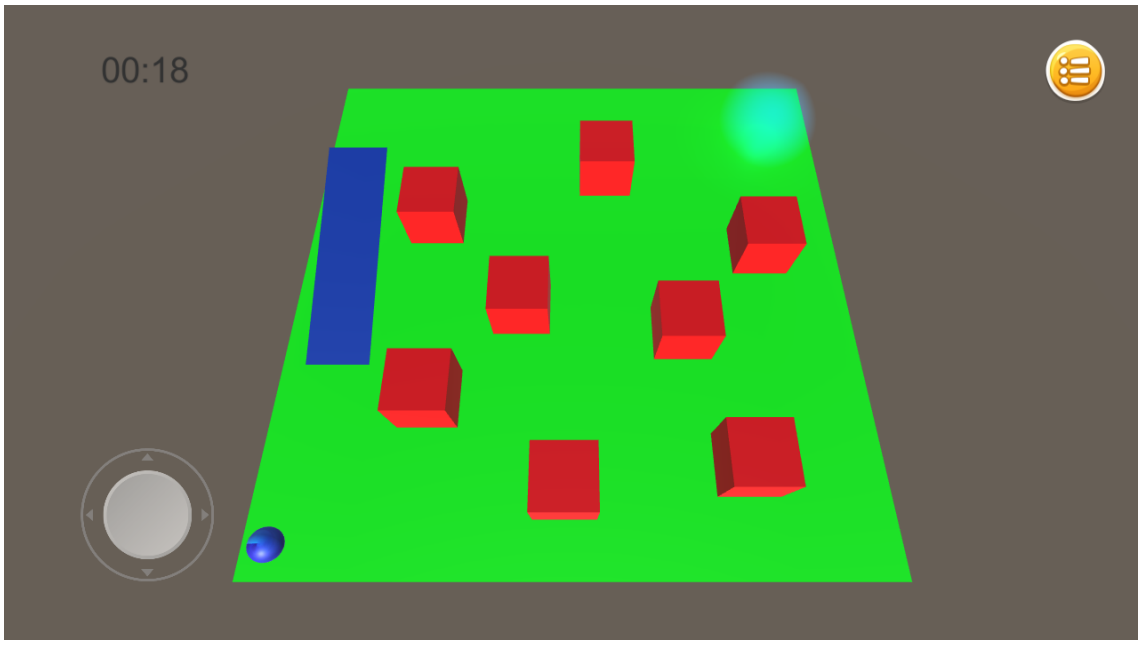


Ilustración 10.22 Interfaz dentro del juego

Finalmente, en la vista *InGameMenu* se presenta un menú de pausa con cuatro botones:

- "Calibrar": para volver a calibrar el acelerómetro del móvil y establecer la posición actual como el punto de origen del sistema de referencia para la inclinación.
- "Opciones": para acceder al menú de opciones desde el segundo ViewSwitcher, evitando así el problema de la transición cuando el juego está en pausa. Para esto se usa un booleano que indica si se ha accedido desde el menú de pausa y, por tanto, se deberá volver a ese mismo menú.
- "Reiniciar": para volver a empezar el nivel.
- "Salir": para salir del nivel y volver a la pantalla de descripción del nivel.

Los dos últimos botones además cierran el menú de pausa y vuelven a activar el tiempo dentro del juego, evitando así un error de versiones anteriores en el que en ciertas circunstancias se mostraba el menú sin pausar el juego y el juego se pausaba al cerrar el menú.

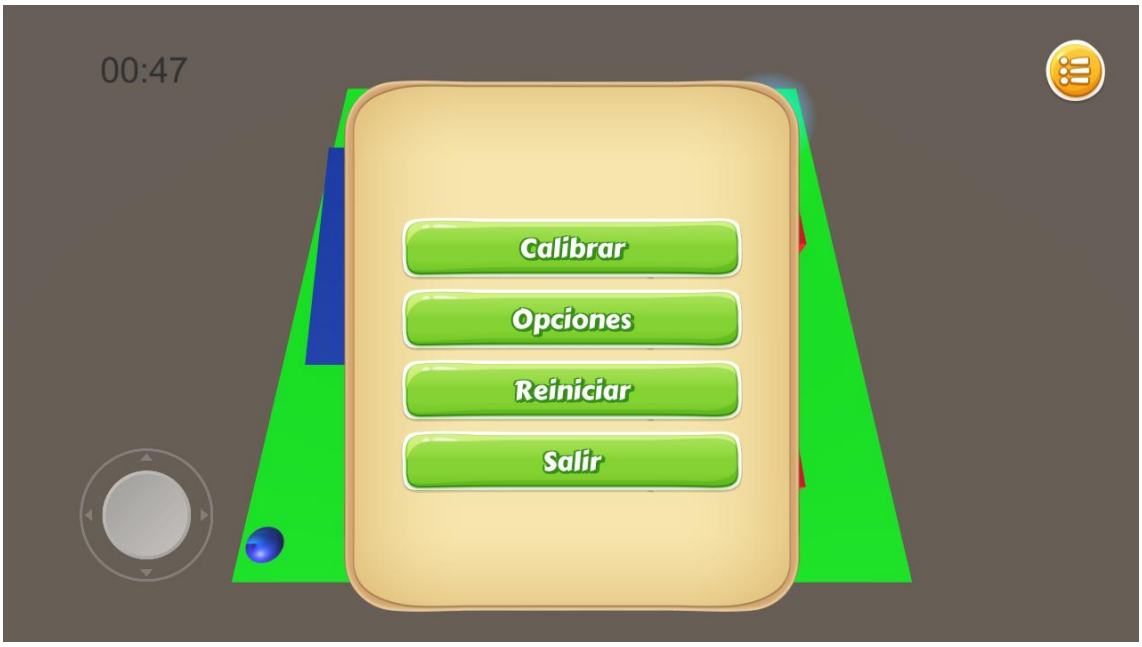


Ilustración 10.23 Menú de pausa

10.3.8 - Aspectos generales:

Este no es el único error que se ha detectado y solucionado en esta iteración. Se solucionó un error por el que al salir de un nivel desde el menú de pausa se permitía registrar el tiempo transcurrido como récord del nivel, sin necesidad de haber completado el mismo. Se solucionó otro error en el que el plano de detección de caídas no era lo suficientemente grande y provocaba la posibilidad de que la bola cayera al vacío indefinidamente.

Dado que se van a utilizar dos controladores del movimiento de la bola diferentes en el juego, pero con un comportamiento similar, se ha decidido implementarlos mediante herencia. Para ello se han agrupado los métodos comunes en la clase *BallController*, entre los que se incluyen la obtención del vector de movimiento por medio de los diferentes métodos de entrada posible: acelerómetro, botón deslizante y el movimiento por defecto de Unity (WASD y flechas del teclado). También son comunes una serie de atributos, como los booleanos *giroscopio* y *permiteMovimiento*, con sus respectivos *Getters* y *Setters* que indican si se está usando control por inclinación y si se permite aplicar el movimiento a la bola. También está en esta clase el método para calibrar el acelerómetro y establecer la posición neutra u origen de su sistema de coordenadas.

Posteriormente se tienen otras dos clases que heredan de esta y son *BallController1* y *BallController2*, en las cuales se definen los métodos específicos de

cada sistema de movimiento y que básicamente son el *Start* y el *Update*. *BallController1* se corresponde al sistema de control con cámara a la espalda y movimiento relativo a la orientación de la cámara. *BallController2* es una de las versiones primitivas de movimiento, en el que en cada fotograma se aplican fuerzas de tipo impulso a la bola basándose en la dirección proporcionada por el usuario y la última fuerza que se aplicó sobre la bola. Este *script* no debe preocuparse por el control de la cámara, puesto que para este sistema de control se ha decidido utilizar una de las cámaras proporcionadas dentro de los *Standard Assets* de Unity, en concreto la *MultipurposeCameraRig*. Con el uso de esta cámara podemos tener una vista cenital de la escena, a la vez que la cámara sigue a la bola en su desplazamiento, pero manteniendo la misma posición relativa respecto a la misma.

En esta iteración no se realizará una evaluación del progreso por parte de los probadores, sino que será suficiente con la revisión del tutor. La nueva interfaz gráfica podrá ser evaluada junto con el conjunto de niveles del primer mundo en la cuarta iteración.

10.4 - Cuarta iteración

El objetivo de esta iteración es diseñar y construir el conjunto de niveles que componen el primer mundo del juego. El diseño de los niveles partirá de los bocetos sobre papel mostrados en la sección “Diseño de Niveles”.

En primer lugar, se han generado los objetos de juego principales que serán recurrentes en todos los niveles, como barreras para evitar que la bola se caiga de una plataforma, las propias plataformas, algunos obstáculos y los recolectables. Estos objetos, junto con otros ya generados como la bola, la meta, el *LevelController* o la cámara se guardarán como “prefabs” para poder ser usados de forma rápida en otros niveles [33]. Los “prefabs” son objetos prefabricados de los que pueden crearse instancias en la escena, heredando estas todas las propiedades, tanto actuales como futuras (causadas por la modificación de los “prefabs”), del objeto original, pero permitiendo, a su vez, la alteración de la instancia para ajustar el “prefab” a las necesidades concretas de la escena en la que se usa.

Con esta metodología ha sido posible acelerar en gran medida el proceso de creación de nuevos niveles, ya que para crear un nuevo nivel sólo era necesario crear

una escena y añadir a ella todos los “prefabs” que fuera necesarios. Todos los niveles deben contar con el “prefab” de la bola que se deberá modificar incluyendo el *script* para el comportamiento de la bola adecuado según el sistema de control que se desee utilizar. En caso de que el sistema de control sea el de vista cenital deberá desactivarse la cámara (que forma parte del “prefab” de la bola) e incluir una variación del “prefab” de la cámara “MultipurposeCamera” de los “Standard Assets” de Unity, y si se utiliza el otro sistema de control habrá que indicar al *script* el objeto cuya matriz de transformación se empleará como raíz de la jerarquía de objetos de la escena, y por tanto, será ese objeto al que se le apliquen las rotaciones de la escena en las que se basa este sistema de control.

Cada nivel deberá incluir también una instancia del “prefab” *LevelController*, que será encargado de gestionar la lógica dentro del nivel mediante el *script* homónimo que tiene como componente, una instancia del “prefab” del *canvas* para la interfaz gráfica dentro del juego que complementa a la descrita y desarrollada durante la iteración previa. Y dentro de un objeto vacío llamado “Escena” se incluirán una instancia del “prefab” de una *DirectionalLight,* una instancia del “prefab” *Limite* (*collider* para la detección de la caída de la bola) y una instancia del “prefab” de la meta y tres instancias del “prefab” de los recolectables.

Según el diseño de cada nivel se incluirán dentro del objeto “Escena” tantas instancias de los tipos de “prefab” restantes como sean necesarios y aplicándoles las modificaciones que sean precisas.

Esta metodología para la construcción de niveles tiene otra gran ventaja, y es que si se desea realizar algún cambio sobre alguno de los elementos que se han guardado como “prefabs” no será necesario modificar cada una de las instancias ya creadas de estos, sino que simplemente se modifica el “prefab” y los cambios provocados en este se reflejarán automáticamente en todas las instancias ya creadas, por lo que se puede ahorrar una gran cantidad de tiempo.

A continuación, se mostrarán los prototipos de los niveles del primer mundo, realizados durante esta iteración siguiendo el proceso anteriormente descrito. En iteraciones posteriores se elaborará la versión definitiva de estos niveles modificando los “prefabs” que se usan para darles la apariencia estética que se desea e incluyendo algunos elementos decorativos extras.

10.4.1 - Nivel 1-1

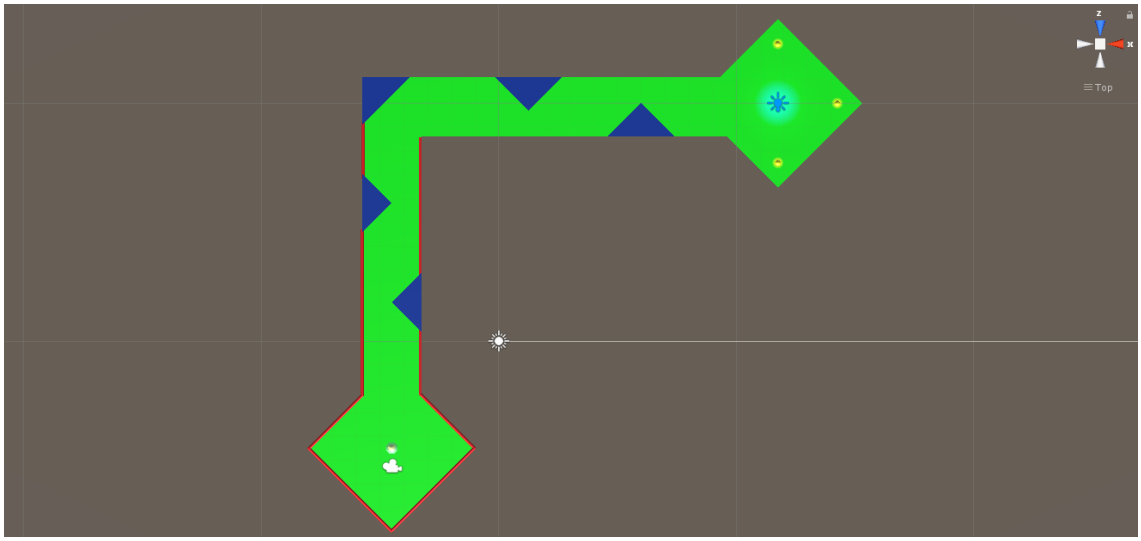


Ilustración 10.24 Nivel 1-1 vista en proyección

Los prismas triangulares de este nivel se han creado mediante la herramienta "ProBuilder", obtenida de la "Asset Store" de Unity.[23] Esta herramienta permite crear y modelar objetos 3D de forma sencilla desde el propio editor de Unity. En este caso se ha utilizado una herramienta con la que se define un polígono mediante sus vértices y posteriormente se le confiere una altura convirtiendo el polígono en la base de un prisma de esta altura. Una vez generado el obstáculo se almacena como "prefab" por si se vuelve a requerir su uso en un futuro. Las barreras son cilindros y las plataformas son planos, y ambos se almacenan como "prefabs" para su uso en niveles posteriores.

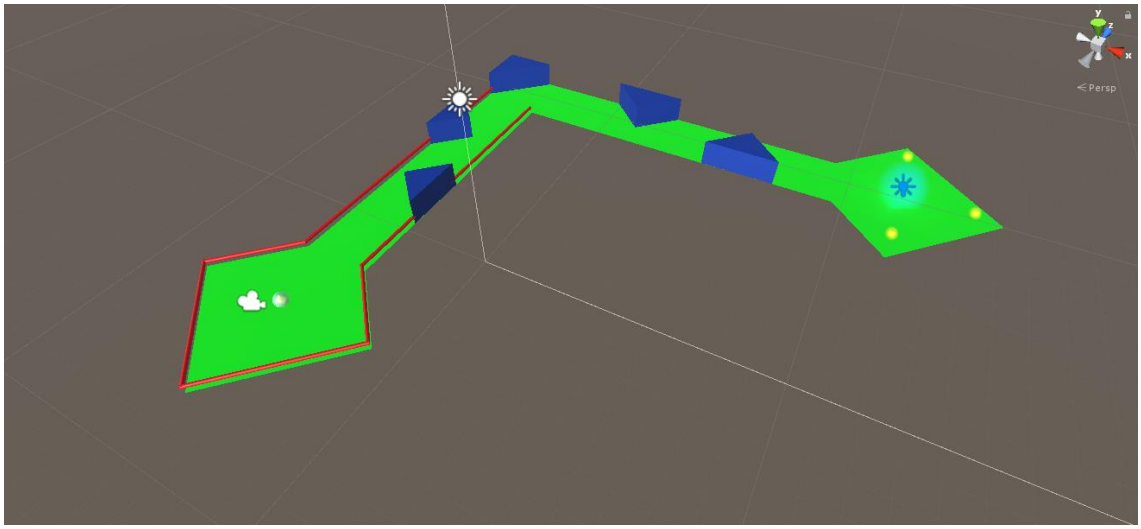


Ilustración 10.25 Nivel 1-1 vista en perspectiva

10.4.2 - Nivel 1-2

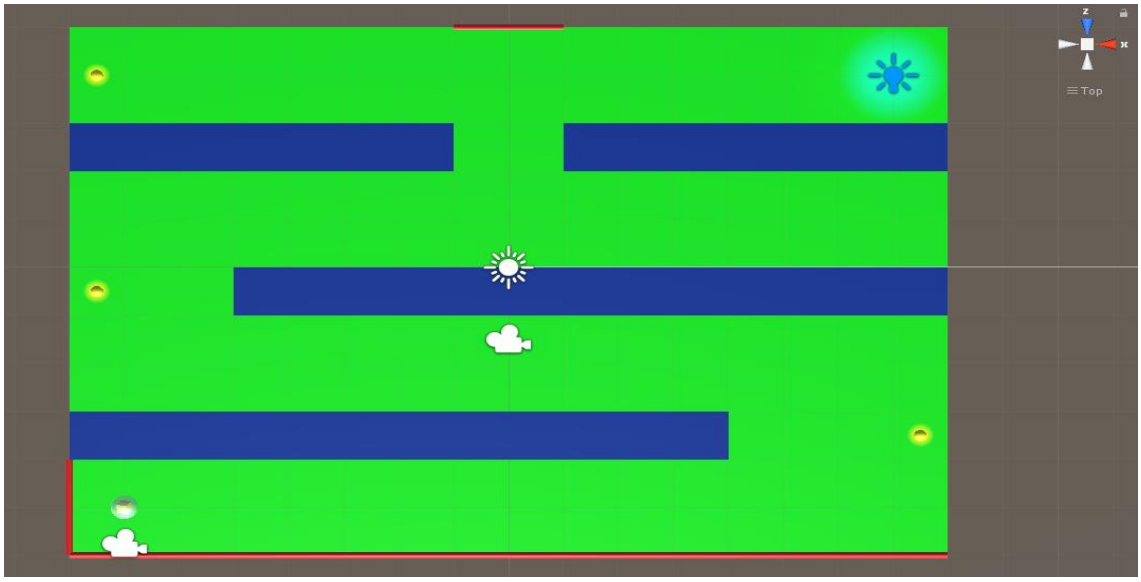


Ilustración 10.26 Nivel 1-2 vista en proyección

Este nivel está construido con los “prefabs” de una plataforma y barreras y se ha creado un obstáculo nuevo de tipo muro con un cubo escalado que también se almacena como “prefab”.

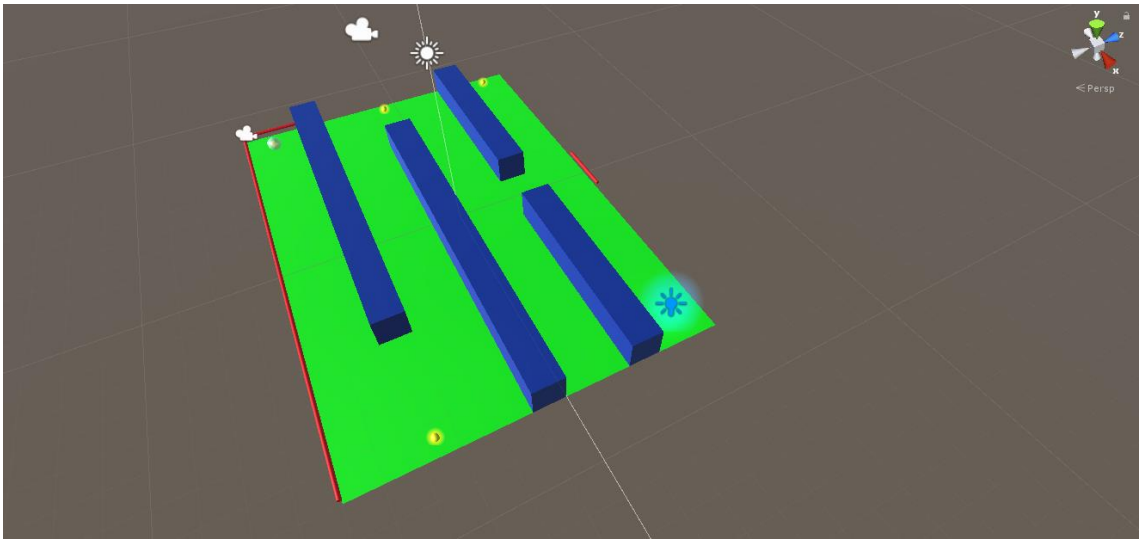


Ilustración 10.27 Nivel 1-2 vista en perspectiva

10.4.3 - Nivel 1-3

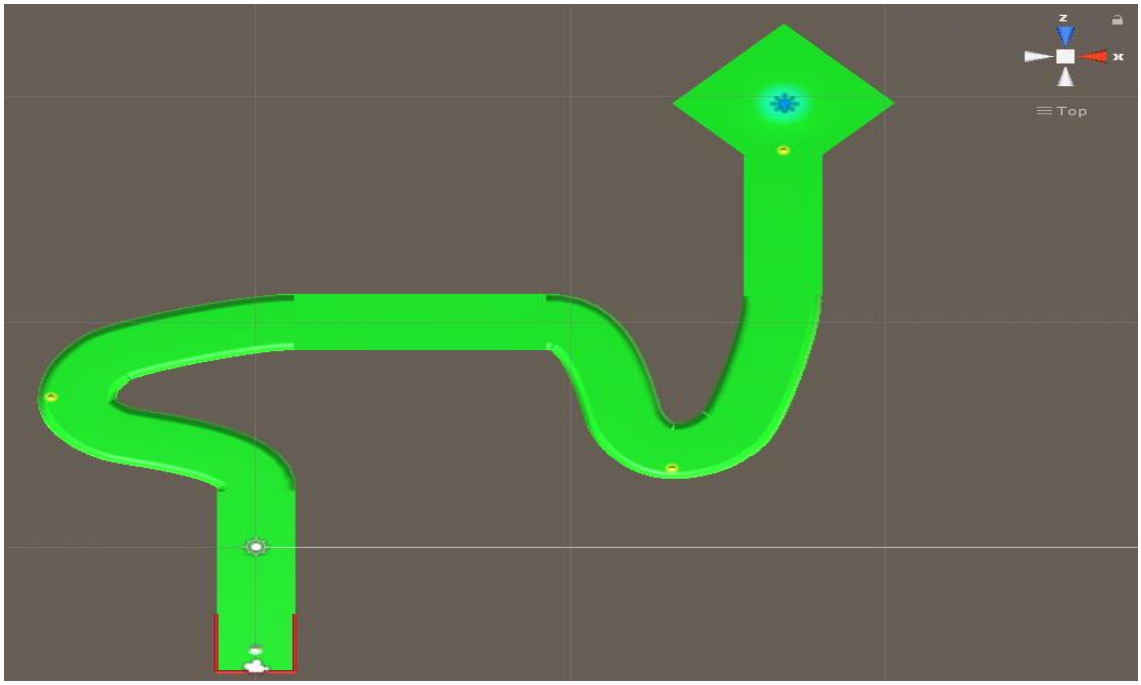


Ilustración 10.28 Nivel 1-3 vista en proyección

Las curvas de este nivel están construidas mediante la herramienta “ProBuilder”, pero en este caso haciendo uso de una herramienta para generar curvas de Bézier, que posteriormente se han escalado en la altura para que sean planas.

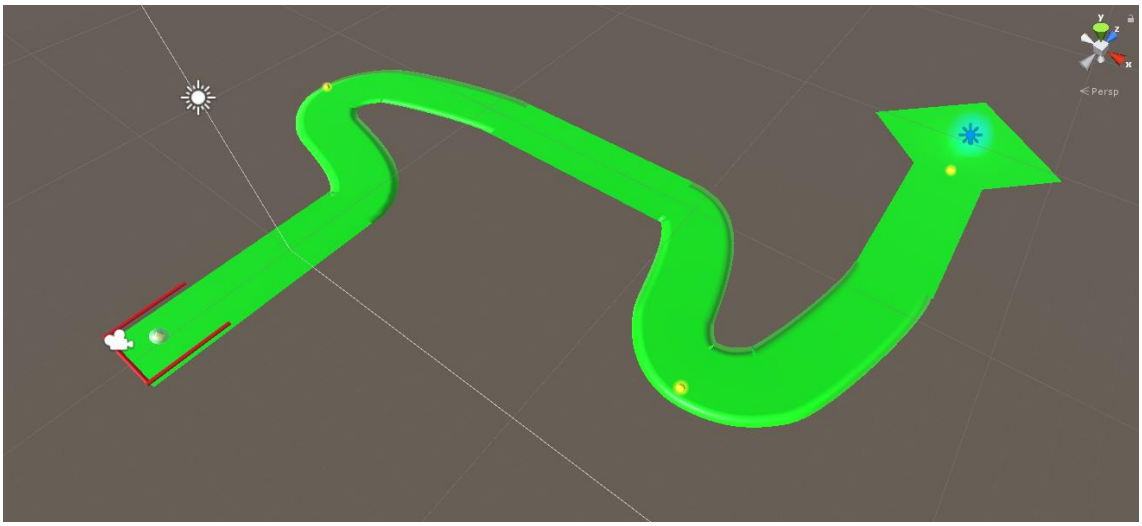


Ilustración 10.29 Nivel 1-3 vista en perspectiva

10.4.4 - Nivel 1-4

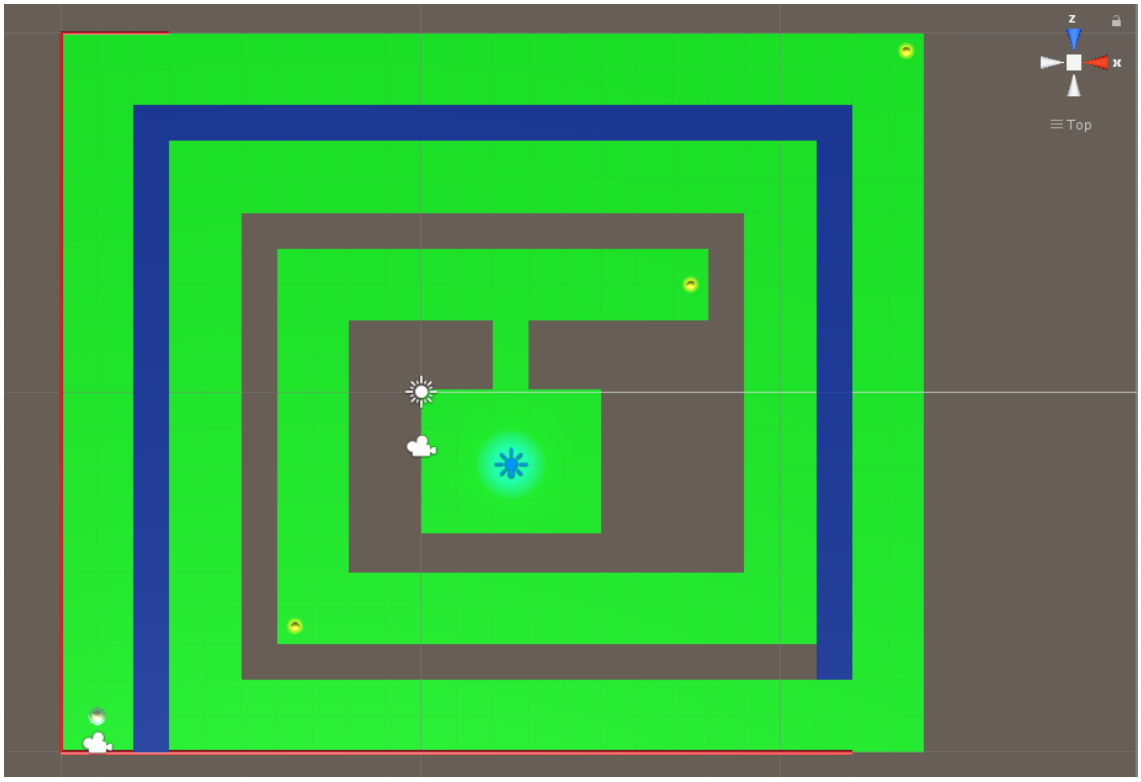


Ilustración 10.30 Nivel 1-4 vista en proyección

Este nivel se ha diseñado como una especie de recorrido en espiral con cada vez menos barreras de ayuda. La forma en espiral está construida utilizando varios planos rectos.

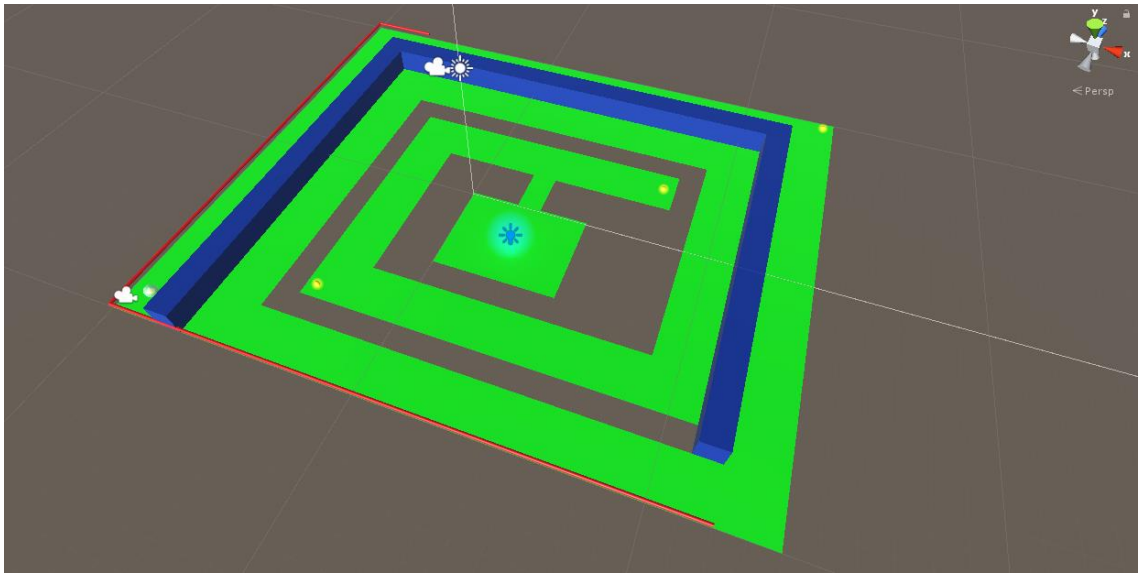


Ilustración 10.31 Nivel 1-4 vista en perspectiva

10.4.5 - Nivel 1-5

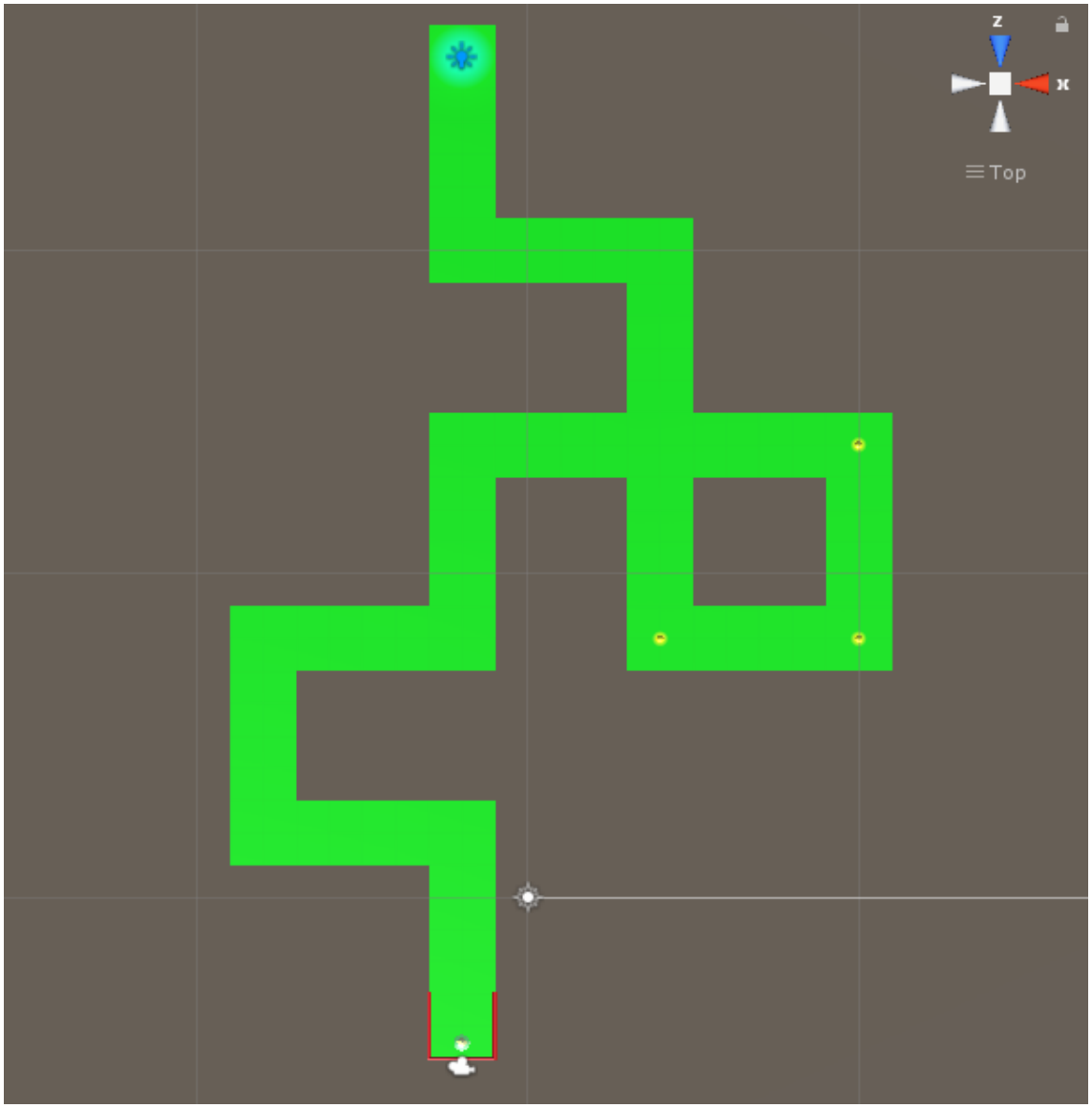


Ilustración 10.32 Nivel 1-5 vista en proyección

Este nivel está planteado como un recorrido con giros rectos con un tramo omisible en el que se encuentran los recolectables.

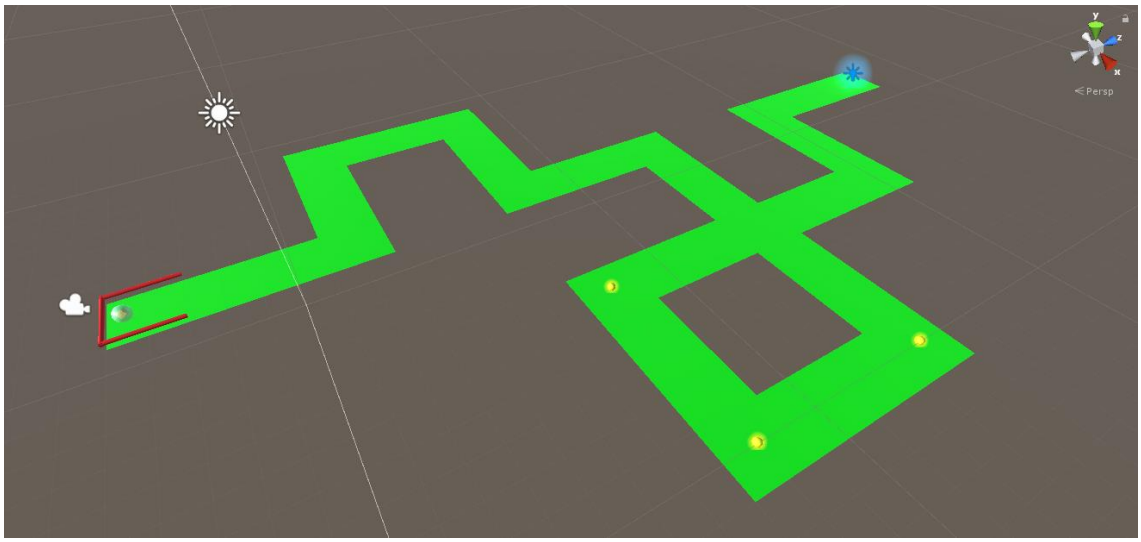


Ilustración 10.33 Nivel 1-5 vista en perspectiva

10.4.6 - Nivel 1-6

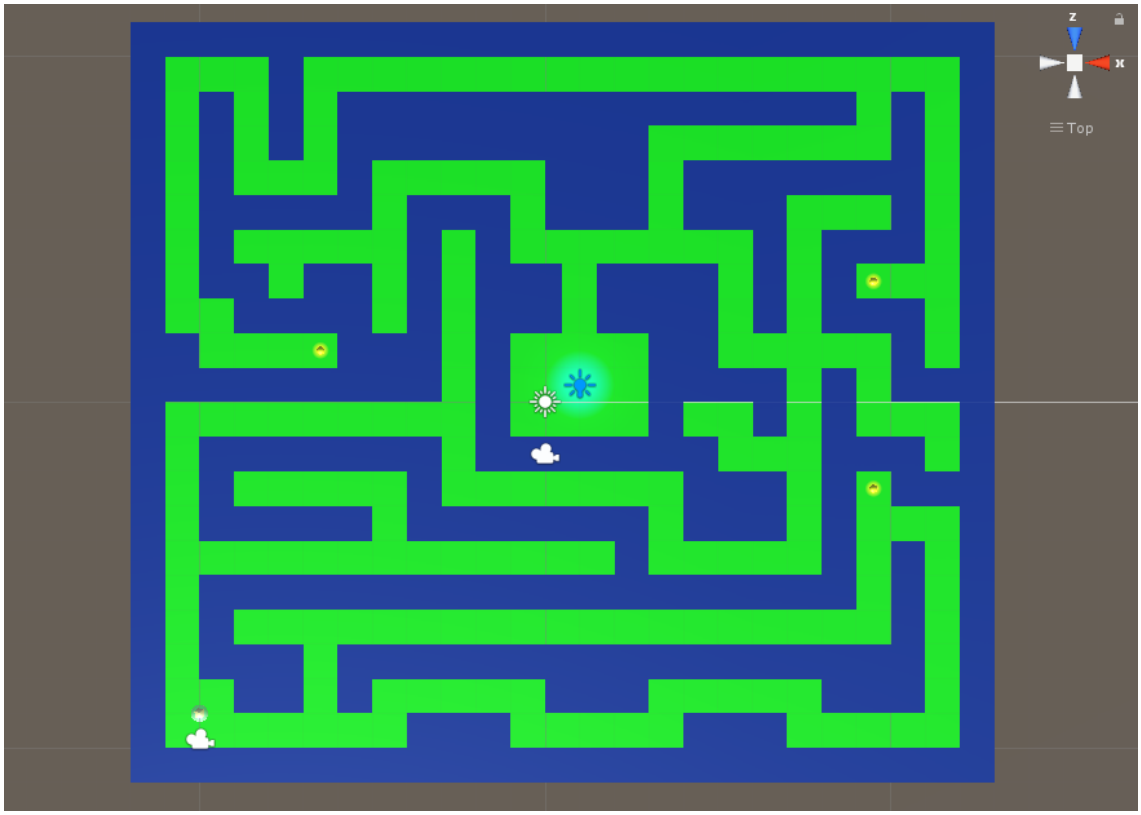


Ilustración 10.34 Nivel 1-6 vista en proyección

Este nivel es un laberinto construido a partir del “prefab” de muro en el que el jugador aparece en una de las esquinas y deberá llegar al centro del laberinto. Para

hacer más interesante el recorrido del laberinto se ha empleado una cámara más cercana al jugador que la presente en otros niveles.

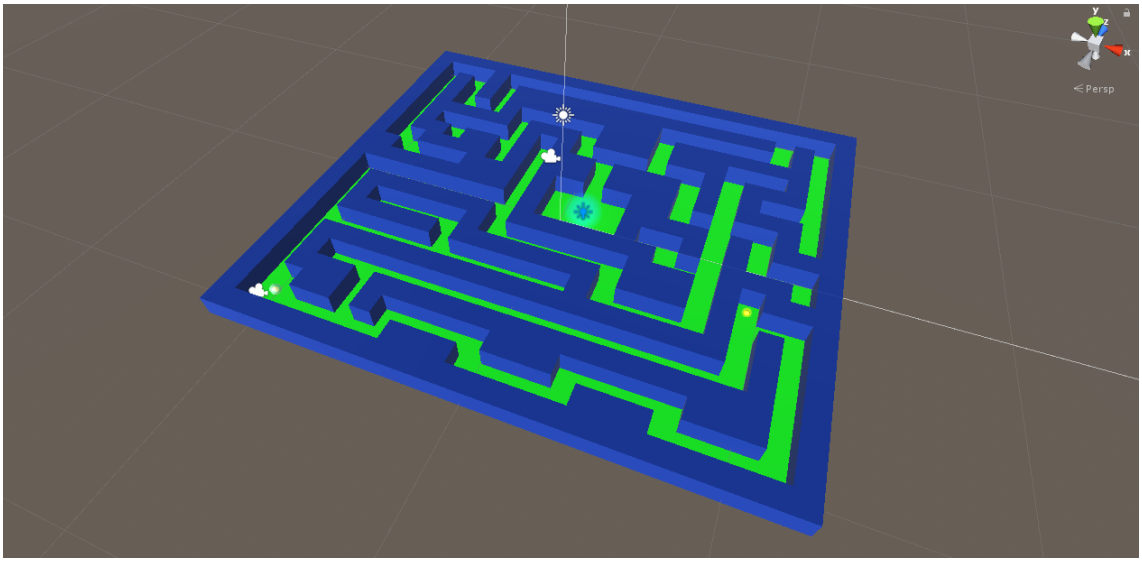


Ilustración 10.35 Nivel 1-6 vista en perspectiva

10.4.7 - Nivel 1-7

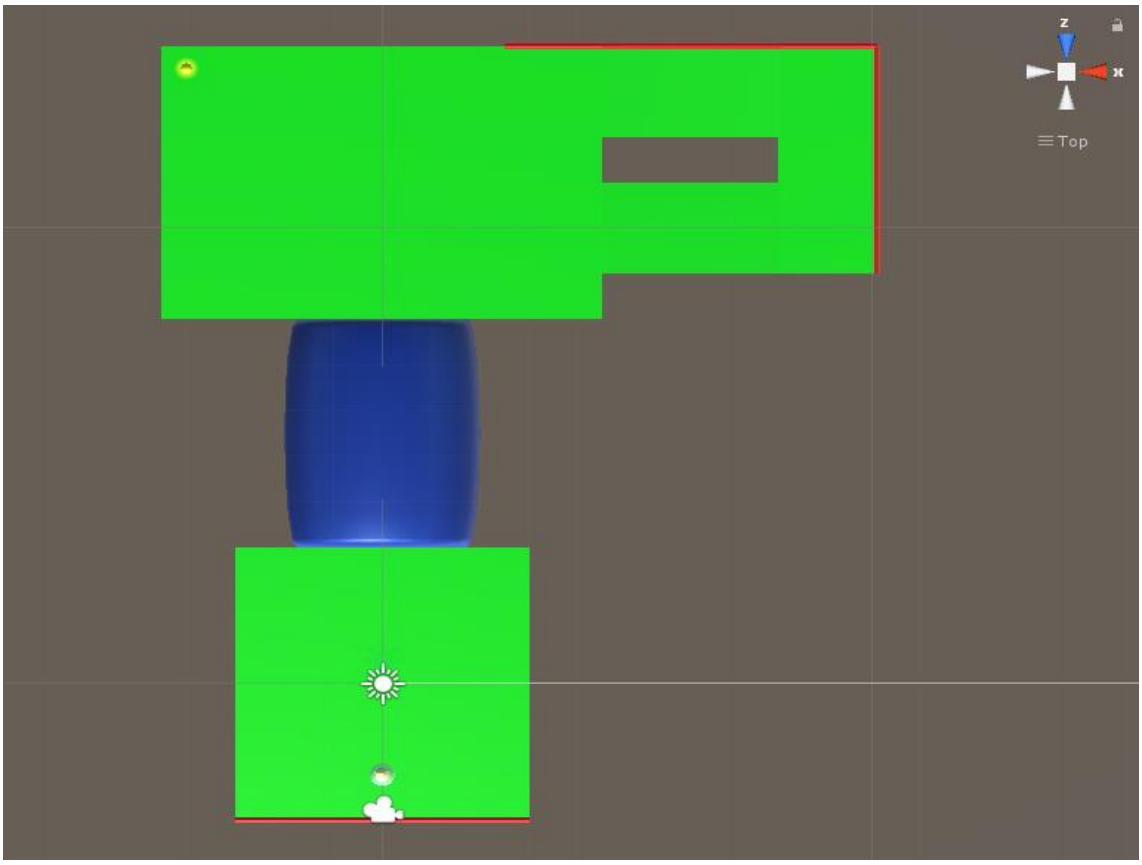


Ilustración 10.36 Nivel 1-7 vista en proyección

En este nivel se emplean por primera vez plataformas a distintas alturas y rampas para ir de una a otra. Las rampas no tienen una inclinación demasiado elevada (entorno al 37% de inclinación) para evitar que el usuario pierda el control de la bola al bajarlas, especialmente en su primer encuentro con las rampas. Es por esto que se usan también barreras en la primera rampa. Otro de los atractivos de este nivel es un túnel, que primero se cruza por encima y luego se atraviesa. Este túnel es en realidad un toroide construido y escalado mediante la herramienta “ProBuilder”. En este nivel es necesario que las plataformas de las alturas superiores sean visibles desde abajo. Por defecto esta característica está deshabilitada debido al “back-face culling” que se lleva a cabo a nivel de shader. El “back-face culling” [34] es una técnica utilizada para ahorrar recursos evitando renderizar la parte trasera de los objetos. La forma más sencilla de resolverlo en este caso es colocar otra plataforma con la misma forma y posición que la anterior, pero que su normal esté orientada en el sentido opuesto.

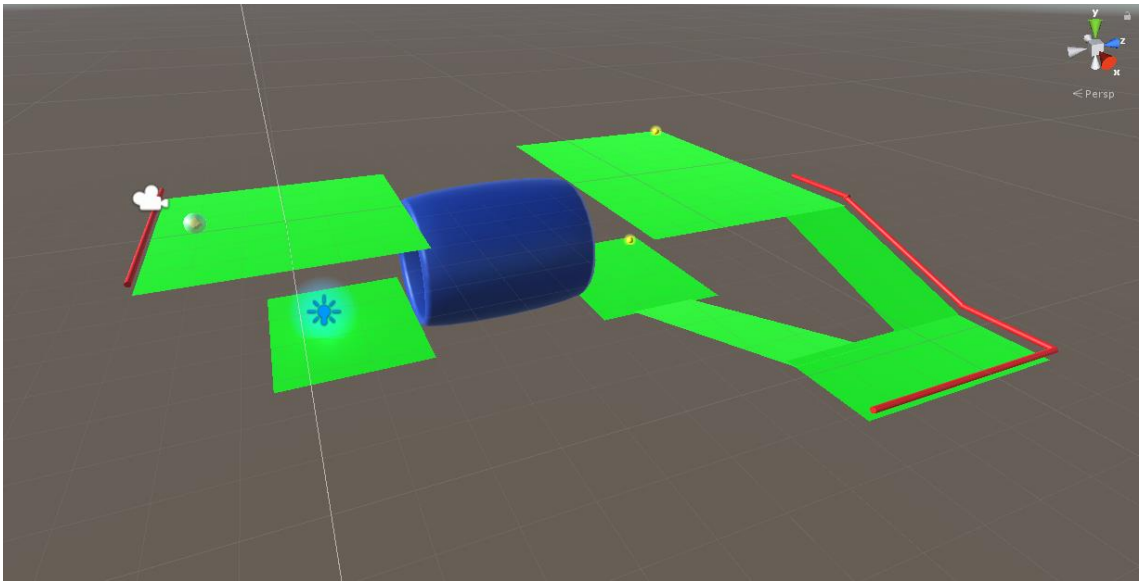


Ilustración 10.37 Nivel 1-7 vista en perspectiva

10.4.8 - Nivel 1-8:

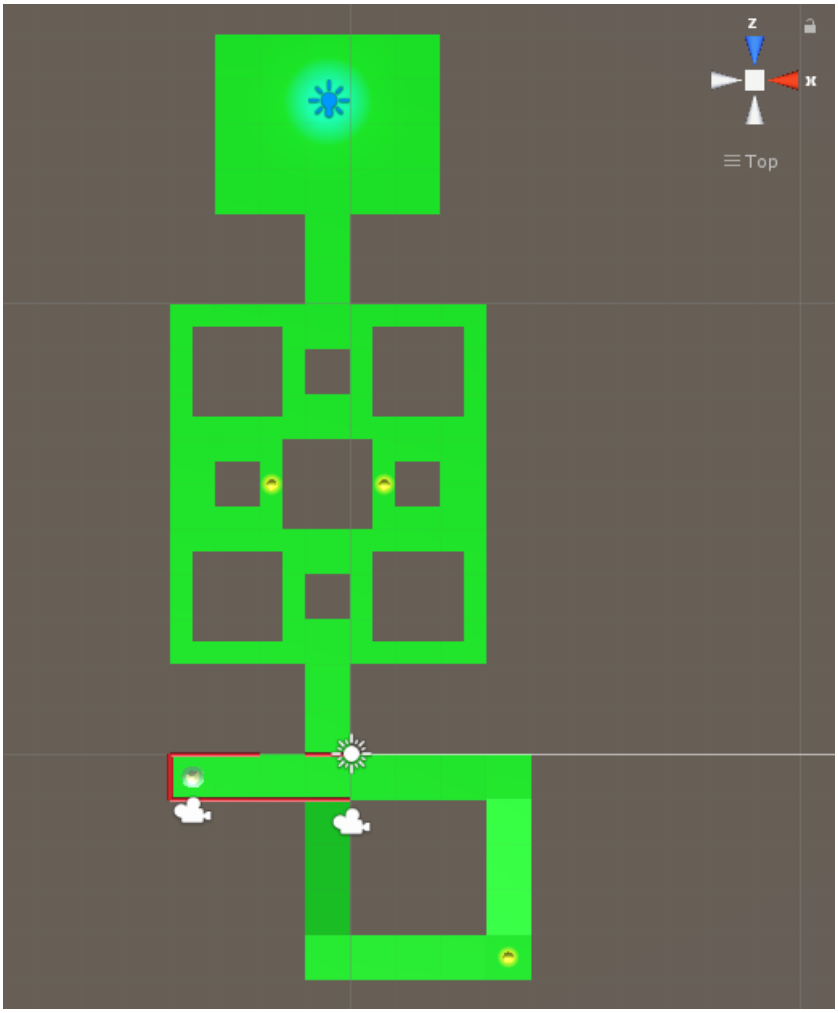


Ilustración 10.38 Nivel 1-8 vista en proyección

En el inicio de este nivel hay una bajada de cuatro rampas sin escaleras, por lo que la inclinación de cada rampa se ha reducido todavía más hasta el 16%Tras esto llega una plataforma plagada de agujeros con múltiples caminos posibles. Realmente la plataforma se ha construido utilizando múltiples “prefabs” de planos escalados.

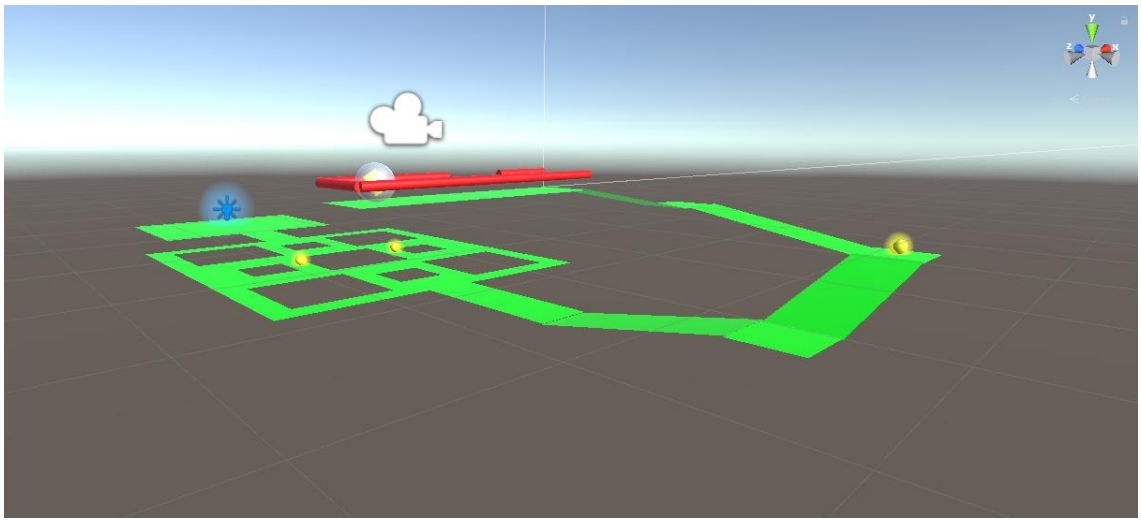


Ilustración 10.39 Nivel 1-8 vista en perspectiva

10.4.9 - Aspectos generales

Durante la creación y prueba de los niveles se estimó que sería más adecuado usar una cámara más cercana a la bola que la que se había planteado en las primeras iteraciones. También se detectó y corrigió un error en el *script BallController1* por el que en ciertas ocasiones se impedía realizar una inclinación lateral del escenario. Este error venía causado por que se impedían las rotaciones de la escena si el escenario ya estaba inclinado lo máximo permitido salvo si inclinaba hacia atrás. La solución a este error pasaba por permitir también cualquier movimiento que generase una inclinación lateral del escenario menor a la que hubiera actualmente.

Por otro lado, se modificó la naturaleza del impulso aplicado sobre la bola para que continuara ganando velocidad sin limitarse en una velocidad máxima. Hasta ahora se había trabajado con una aceleración lineal para estos casos, en los que se aumentaba la velocidad en cada fotograma en una cantidad proporcional a la propia velocidad que llevara la bola. Esto se cambió por una aceleración que iba aumentado de forma logarítmica por cada fotograma en que se estuviese moviendo la bola en una dirección sin frenarse.

Sin embargo, tras realizar varias pruebas en distintos dispositivos, y pese a emplear una compensación a la aceleración en función del tiempo transcurrido entre dos fotogramas (*Time.deltaTime*), se detectó que en unos dispositivos más potentes se conseguía alcanzar una mayor velocidad que en otros menos potentes, y, por consiguiente, era posible completar los niveles en tiempos inferiores que los tiempos mínimos para otros dispositivos. Es por esto que para este prototipo se ha decidido

suprimir esta característica, y la bola no contará con una aceleración adicional. Precisamente el uso de cámaras más cercanas ayuda a conseguir una mayor sensación de velocidad en compensación por la velocidad real que se ha perdido al eliminar esta aceleración extra.

10.5 - Quinta iteración

El objetivo de esta quinta iteración es realizar una serie de mejoras que ayuden al juego a dar un salto de calidad. Entre estas mejoras se incluyen la sustitución de los materiales básicos que se han ido utilizando hasta ahora por texturas, la inclusión de modelos tridimensionales a los niveles para añadir detalles estéticos a los niveles y como sustitución de otros ya empleados, música y efectos de sonido y la inclusión de una pequeña ventana en la esquina inferior derecha para ver en tiempo real el valor del acelerómetro.

En primer lugar, para construir el radar con la información sobre el valor de inclinación suministrado por el acelerómetro se ha empleado un nuevo *Canvas Group* en la interfaz del juego. Este *Canvas Group* únicamente se muestra si el tipo de control que se ha seleccionado para el juego es el control por inclinación y está formado por un panel para destacarlo frente al fondo, un radar que muestra el espacio de valores por el que puede oscilar el valor del acelerómetro y un punto rojo que muestra el valor de este en un instante concreto. Todos estos elementos son imágenes de la interfaz. Durante la función *Update()* del *script LevelController* que controla la lógica del nivel, se desplaza la imagen de la interfaz correspondiente con el punto rojo de forma proporcional a la entrada del acelerómetro y de forma relativa a su punto de anclaje. Se hace coincidir el valor máximo del acelerómetro con el circulo exterior del radar.

En la Ilustración 10.40 se puede apreciar el radar del giroscopio.

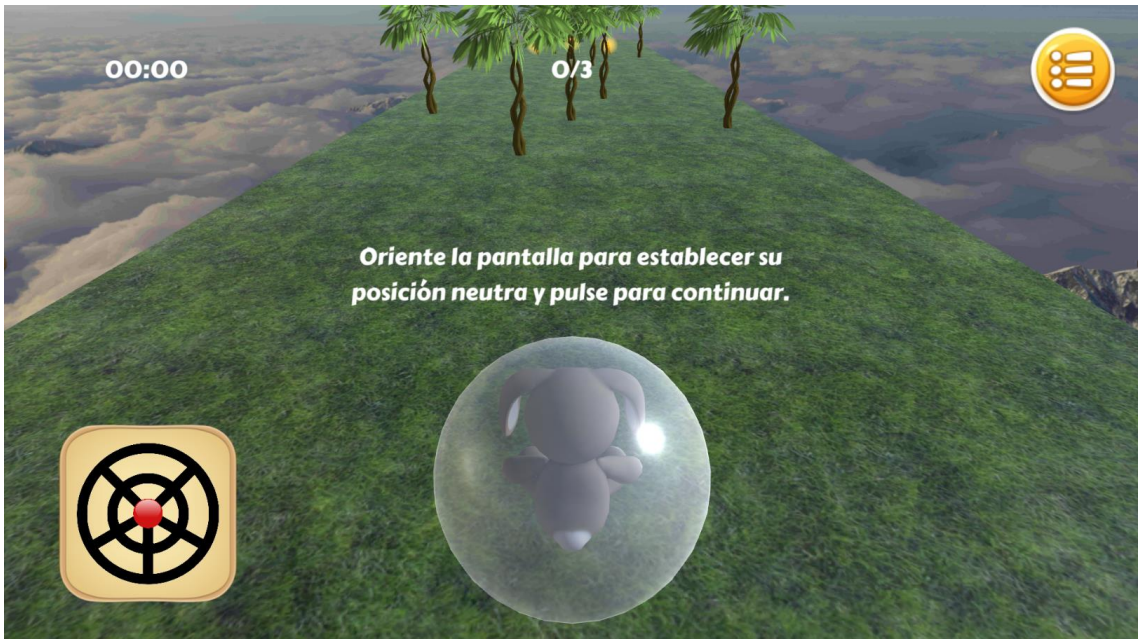


Ilustración 10.40 Interfaz dentro del juego con inclusión de radar para giroscopio

Como textura a aplicar sobre las plataformas del suelo se ha buscado una textura que permita su repetición tanto de forma horizontal como de forma vertical sin mostrar de forma aparente el cambio entre el fin de una textura y el inicio de otra. Para poder hacer esto de forma cómoda con plataformas a distintas escalas será necesario un "tiling" variable [35]. El "tiling" proviene del inglés "poner baldosas" y es equiparable a teselar, es decir, dividir un espacio en un determinado número de unidades del mismo tamaño con las que generalmente se dibuja ese espacio. En este contexto se utiliza el "tiling" para determinar el número de repeticiones de la textura que se desean en el plano. Para esta textura se ha decidido emplear un "tiling" de 5 tanto horizontal como verticalmente, es decir, se producirán 5 repeticiones de la textura a lo largo del eje x y otras cinco repeticiones a lo largo del eje y.

El problema, en este caso, es que estas plataformas están siendo escaladas de forma continua a partir del valor original del "prefab", y al escalar la plataforma también se escala la textura, por lo que no todas las baldosas van a seguir siendo del mismo tamaño. Por este motivo se ha implementado un *script*, denominado *TextureTiling*, con el que se modifica de forma automática el "tiling" que le corresponde a la plataforma en función de su escala para lograr que todas las baldosas tengan el mismo tamaño y no sea necesario modificar de forma manual el "tiling" cada vez que se quiera escalar una instancia del "prefab" plataforma. Los resultados del nuevo "tiling", sin embargo, sólo son visibles en tiempo de ejecución, por lo que mientras se

está editando el nivel parecerá que las texturas están a distintas escalas. Con este sencillo *script* se está consiguiendo nuevamente agilizar el proceso de construcción de un nivel, que como ya se ha comentado anteriormente en esta memoria, es muy importante para un videojuego de estas características, en el que existen una gran cantidad de niveles independientes, y que se espera ir añadiendo más en el futuro de forma regular.

Para generalizar más el *script* y hacerlo válido para otras texturas y otras superficies en las que se vaya a aplicar distintas de planos se han añadido algunas variables públicas a este. Por ejemplo, permitiendo escoger el "tiling" en la coordenada x y en la y para una escala estándar del objeto. De esta forma, el mismo *script* será válido para texturas que requieran un mayor "tiling" de partida que otras. La otra modificación incorporada es para los objetos cilíndricos, en los que el "tiling" vertical ya no se corresponde con la dimensión z, como sucedía con los planos, sino con la dimensión y del objeto.

Para los obstáculos se ha utilizado una textura que simula muchas hojas que impiden el paso, mientras que para las barreras se ha empleado una textura de madera. En ambos casos se hace uso del *script* anterior.

Cabe destacar que para lograr un mayor control sobre la aplicación de textura sobre el "prefab" del muro, que estaba basado en un cubo, se ha sustituido el cubo por 6 planos para formar el cubo, integrados dentro de un objeto vacío que incluye la escala y el *collider* del cubo. Este cambio en el "prefab" se debe a que al aplicar la textura directamente sobre el cubo, no se aplicaba de forma correcta el "tiling" en dos de las caras, pues se tomaban los valores de escala del cubo en general, pero podían no corresponderse con los de las caras. Es decir, al aplicar un escalado en una única coordenada se modificaba el "tiling" en esa coordenada, pero el objeto resultante dejaría de ser un cubo para ser un prisma, en cuyas bases no se habría producido el efecto de escala, pero sí se aplicaría en el "tiling" de su textura.

Se han sustituido los modelos básicos del interior de la bola y los recolectables por otros modelos algo más complejos y con una estética más llamativa. El cubo del interior de la bola se ha sustituido por un modelo de un pequeño conejo y la esfera de los recolectables se ha sustituido por una zanahoria. Se pueden apreciar los modelos utilizados en las Ilustraciones 10.41 y 10.42.

Ilustración 10.41 Modelo 3D del conejo

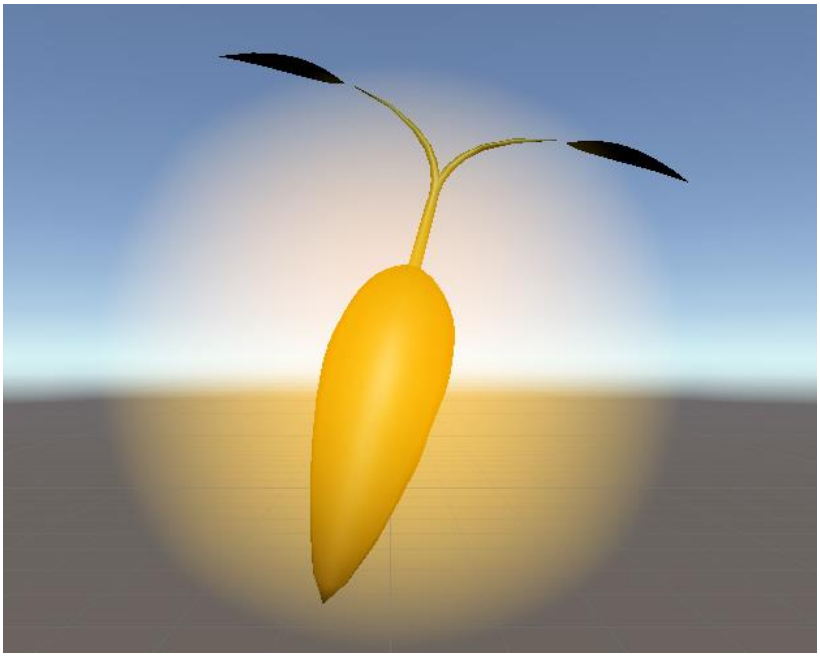


Ilustración 10.42 Modelo 3D de la zanahoria

Por otra parte, y para hacer más atractivos a los usuarios la recolección de los plátanos se animado los recolectables para que giren. Esta acción se realiza en la función *Update()*, por lo que nuevamente será necesario recurrir a *Time.timeScale* para que el ángulo de movimiento por cada fotograma sea proporcional al tiempo transcurrido respecto al último fotograma, y así evitar que la velocidad de giro varíe en función de la potencia del dispositivo en el que se ejecute el juego.

También se ha añadido un nuevo modelo 3D a la meta, que hasta ahora sólo contaba con una luz, y ahora se ha incorporado el modelo de una bandera de cuadros. Este modelo se ha obtenido de una web especializada en modelos 3D gratuitos externa a Unity llamada "TurboSquid" [36]. A este modelo se le ha aplicado una textura de cuadros extraída de otra web distinta denomidada "icons-icons" [37].

Se ha incorporado un "skybox" [38], que es una textura esférica o cúbica formada por 6 texturas planas para representar el fondo de la escena en todas las direcciones. Para los niveles se ha empleado una textura que muestra las nubes desde arriba que se ha obtenido de un sitio web externo llamado HDRI-Hub [39]. También se ha creado un "skybox" para el menú de inicio para que haga la labor de pantalla de carga y se muestre mientras se cargan los niveles. Se trata de una textura negra con letras blancas que pone "Cargando…" con la misma fuente que se ha empleado para los textos de la interfaz.

Un último aspecto para conferirle mayor personalidad y encanto al juego es por medio de la música y el sonido. Se ha descargado de la Asset Store un paquete de música selvática con la que dar ambientación. Tanto el emisor de este sonido como el receptor global de los sonidos se encuentran ubicados en el "prefab" de la bola. El receptor de sonido, está ubicado en el objeto "Player" de la bola mediante el componente "Audio Listener" y el emisor de la música se encuentra en uno de los hijos de "Player", denominado "Music Player" mediante el componente "Audio Source". Esta fuente de audio se encuentra ubicada en la misma posición que el receptor de audio porque no se requiere un sonido 3D para la música, ya que no se va a tratar como un elemento diegético, sino que va a ser un sonido externo al mundo representado en la escena del juego.

También se ha incorporado otra pista de música para el menú de inicio. Se trata de música hawaiana [40] que se ha recortado para reducir su duración.

Hay presentes otros sonidos en el juego, pero en este caso son efectos de sonido que no se reproducen de forma continua en el nivel, sino que se escuchan

cuando se desencadenan ciertos eventos. En concreto, se han incluido un par de sonidos para representar el efecto sonoro del choque de la bola contra un obstáculo y un sonido que se reproduce cuando se consigue un recolectable. Se han empleado dos efectos de sonido distintos para los golpes para que se pueda escoger aleatoriamente uno de los dos al detectar la colisión de la bola contra un *collider* que tenga la etiqueta “Obstaculo” y no de una sensación demasiado repetitiva, pues este sonido puede escucharse repetidas veces dentro de un nivel. La fuente de este sonido está también ubicada en el “prefab” de la bola, concretamente en el objeto “Sound Effect Player”, que es otro hijo del objeto “Player”.

Para el sonido del recolectable simplemente se añade un componente de “Audio Source” al sistema de partículas que se crea cuando se obtiene un recolectable y se selecciona que se reproduzca al despertarse el objeto. No es posible ubicar el “Audio Source” directamente sobre el recolectable porque este se destruye en el momento en el que la bola lo toca, y por tanto se destruiría antes de poder reproducir el efecto sonoro.

Al tener fuentes de sonido distintas se pueden reproducir a la vez la música y los efectos de sonido, y también utilizar un volumen distinto para la música y los efectos. Este volumen es configurable por el usuario en el menú de opciones. Para que los cambios de volumen se hagan efectivos mientras se está en el menú de pausa es necesario pausar primero la música durante un instante, modificar el volumen e inmediatamente reanudar la reproducción. Si esto no se realiza así los cambios de volumen se harán efectivos cuando el juego se reanude, pues la modificación del volumen es susceptible a la escala de tiempo.

Todos estos efectos de sonido se han obtenido de otra web externa de sonidos gratuitos llamada SoundBible [41-43].

También se ha creado un icono para la aplicación, generado a partir de una captura de pantalla de la bola en el juego. Se presenta en la Ilustración 10.43.

Ilustración 10.43 Icono de la aplicación

Finalmente, y en base a los comentarios recibidos por los probadores se han realizado una serie de cambios en algunos de los niveles. En el nivel 1-4 se ha reducido el trazado, haciendo que sea algo más corto, pero a la vez un poco más difícil al reducir en varios puntos el ancho de las plataformas. En el nivel 1-7 se ha bajado la altura de la plataforma sobre la que se situaba la meta, para que se encuentre por debajo de la altura del interior del túnel. En el nivel 1-8 se ha corregido la posición y orientación de las rampas, ya que se creaban unos pequeños escalones indeseados que impedían que la bola avanzará con naturalidad.

También se ha corregido un fallo que impedía que se desbloqueara el mundo siguiente al completar el último nivel del primer mundo. Se ha cambiado el tipo de letra de la interfaz dentro del juego para que se corresponda con el utilizado en el resto de la interfaz.

Y, por último, se han establecido los tiempos concretos a batir en cada nivel a partir de los tiempos mínimos obtenidos por los probadores y por mí mismo.

10.6 - Sexta iteración

El objetivo de esta iteración es diseñar los ocho niveles correspondientes al segundo mundo del juego tomando como base los prototipos sobre papel mostrados en el apartado “Diseño de Niveles”.

A continuación, se mostrarán las versiones finales de los niveles, que se han construido siguiendo la misma metodología explicada en la cuarta iteración.

10.6.1 - Nivel 2-1:

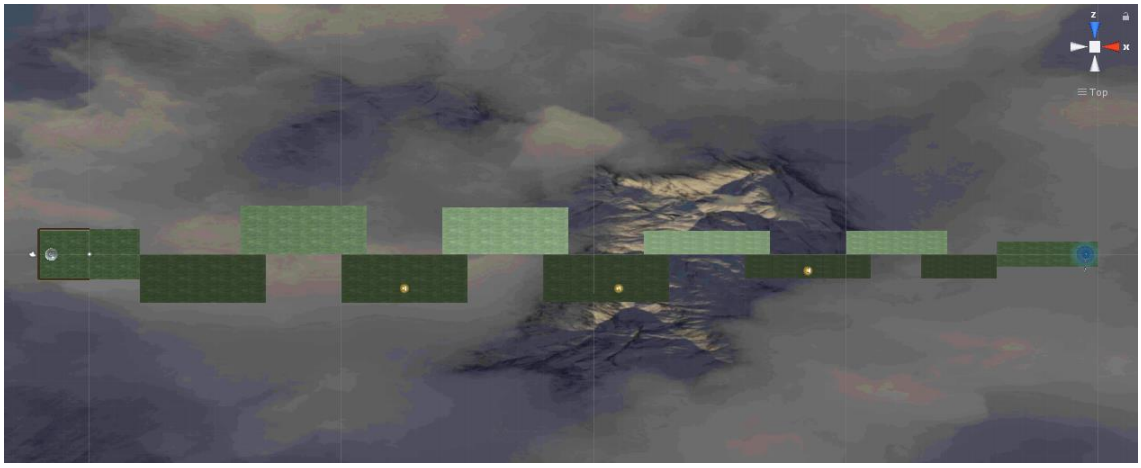


Ilustración 10.44 Nivel 2-1 vista en proyección

En este nivel se han utilizado los “prefabs” de las plataformas y se han inclinado en el eje x. El grado de inclinación va aumentando conforme se desarrolla el nivel, comenzado con 15 de inclinación y terminando con 20. También se aumenta la dificultad en la zona final del nivel al reducir el ancho de las plataformas de dos unidades a una. Los recolectables están colocados para forzar al jugador a recorrer las plataformas por la zona más elevada, lo que en este caso más que una desventaja puede suponer una ventaja para superar el nivel, por lo que se está guiando al jugador hacia la trayectoria más beneficiosa atrayéndole con los recolectables.

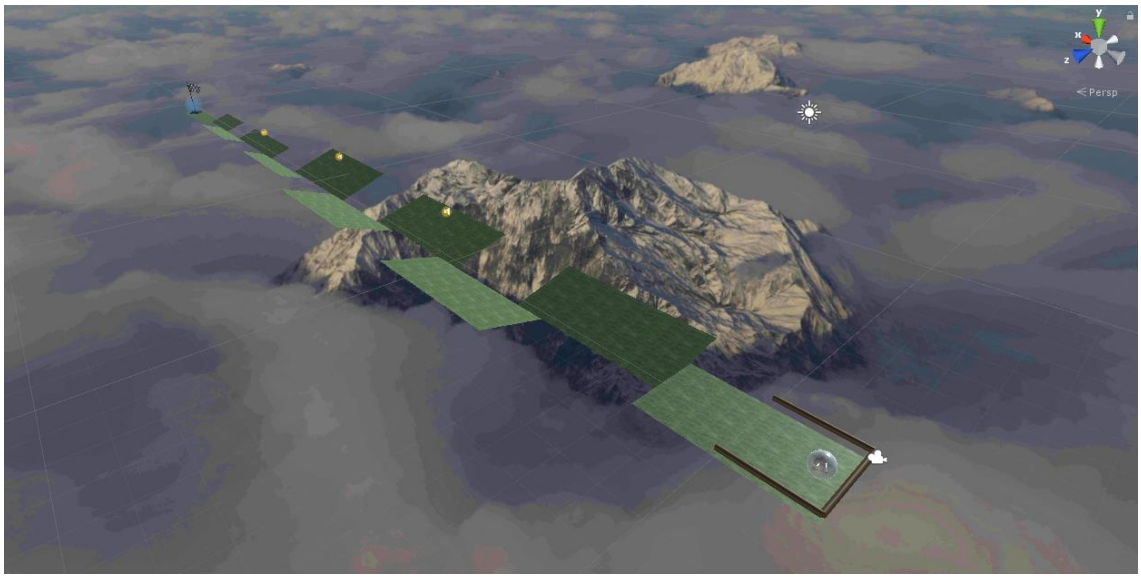


Ilustración 10.45 Nivel 2-1 vista en perspectiva

10.6.2 - Nivel 2-2

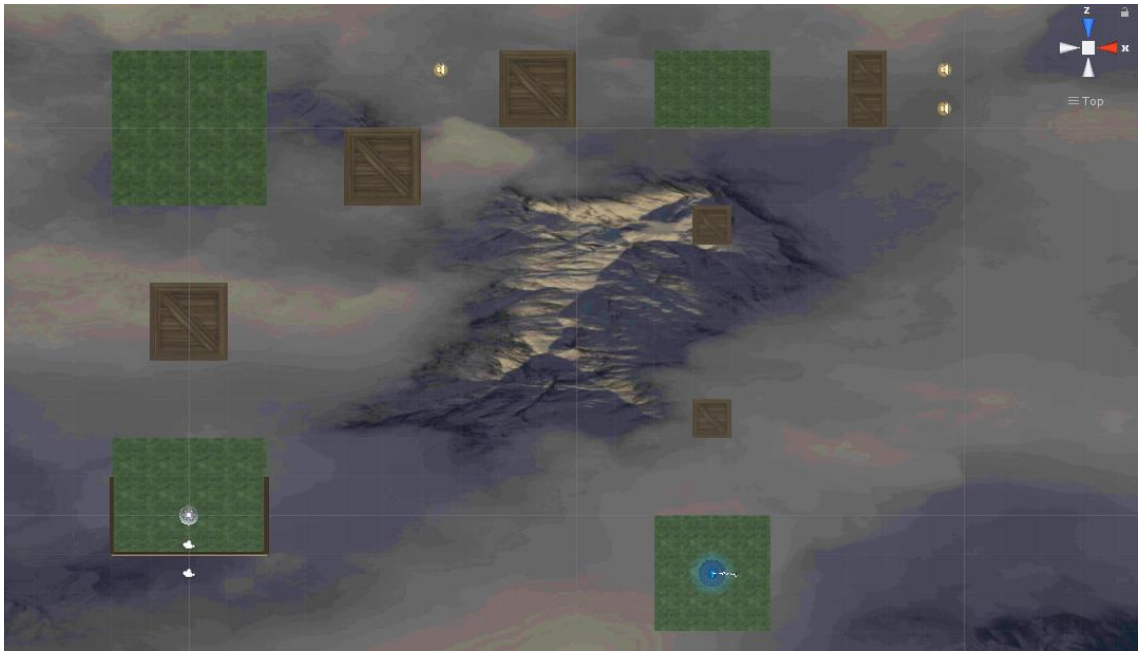


Ilustración 10.46 Nivel 2-2 vista en proyección

Para elaborar el funcionamiento de las plataformas móviles se ha creado un *script* llamado *MovingPlatforms* que se asigna a un plano. Este plano tiene una textura propia que simula una caja de madera y se guarda para utilizar como “prefab”. Este *script* tiene principalmente dos funciones: hacer que la plataforma se mueva constantemente entre dos puntos y permitir que la bola permanezca sobre la

plataforma cuando esta se mueve. El *script* cuenta con tres parámetros claves que son configurables de forma pública para cada instancia del "prefab". Estos son: un tiempo de desfase (durante el cual la plataforma permanece inmóvil al inicio del nivel), un tiempo de espera (es el tiempo que la plataforma permanece parada cada vez que llega al final de su trayectoria) y el tiempo de movimiento (es el tiempo que tarda la plataforma en completar el recorrido, esto es llegar desde el punto de origen al destino o viceversa). Gracias al uso de estos parámetros es posible adaptar la dificultad del juego, lo que permite una mayor versatilidad para desarrollar futuros niveles o implementar distintos niveles de dificultad. También deben suministrarse la posición de inicio del recorrido y la posición final. Al igual que otros "prefabs" sujetos a ser escalables, se le aplica el *script* para el ajuste automático del "tiling" en función de la escala.

El primero de estos objetivos es logrado mediante una corrutina [44] llamada *Patrulla()* que se ejecuta desde el inicio de la vida del objeto. Una corrutina es una función que puede interrumpir su ejecución durante un tiempo determinado o hasta que se produzca cierto evento y permitir mientras la ejecución de otras funciones. En este caso se empieza esperando el tiempo correspondiente al desfase mediante un *yield return new WaitForSeconds()*. Tras esto se inicia un bucle infinito en el que la plataforma irá desde el punto de origen hasta su destino y después volverá de nuevo al origen. En cada iteración de este ciclo infinito se reinicia el contador de tiempo que ha pasado desde el inicio del trayecto y se espera el tiempo indicado para las posiciones finales de la trayectoria mediante un *yield return new WaitForSeconds()*.

En este punto se inicia un nuevo bucle while hasta que el tiempo transcurrido desde el inicio de la trayectoria sea superior al tiempo marcado para la trayectoria. En cada iteración se espera un fotograma mediante *yield return new WaitForEndOfFrame()*, se suma al contador de tiempo el tiempo transcurrido respecto al último fotograma y se calcula la posición actual de la plataforma con una interpolación lineal entre la posición de inicio y la posición final para el cociente de la división del tiempo transcurrido desde el inicio de la trayectoria entre el tiempo estipulado para la trayectoria completa. Se repite este bucle para el trayecto de vuelta de la plataforma, reiniciando el tiempo transcurrido desde el inicio de la trayectoria y tomando como destino la posición de origen y como origen la posición de destino.

Para este nivel se establece un tiempo de desfase uniforme que permita al jugador subir la bola a la plataforma antes de que esta inicie su movimiento, un tiempo

de espera en los extremos de un segundo y tres segundos como tiempo que tarda la plataforma en desplazarse desde un extremo hasta el otro.

Para conseguir que la bola se mantenga sobre la plataforma cuando esta se mueve simplemente debemos asignar la plataforma como padre de la bola en el momento en el que se detecte una colisión entre la bola y la plataforma. De esta forma, la bola mantendrá la misma posición relativa a la plataforma, aunque esta se mueva. En el momento en el que la bola deje de estar colisionando con la plataforma, siempre y cuando la bola no haya adquirido un nuevo padre (al colisionar sobre otra plataforma antes de abandonar la primera), se establece *null* como padre de la bola.

Sin embargo, para que esto funcione sin problemas es crucial que las plataformas tengan escalados uniformes. De lo contrario, al heredar la matriz de transformación del padre se aplica un escalado inverso para compensar la falta de uniformidad en el escalado del padre y eso provoca la deformación del hijo. Esto obliga a que la plataforma móvil de la parte superior derecha de la imagen anterior (de tamaño de 1x2) sea separada en dos plataformas independientes (de tamaño 1x1), pero con el mismo comportamiento.

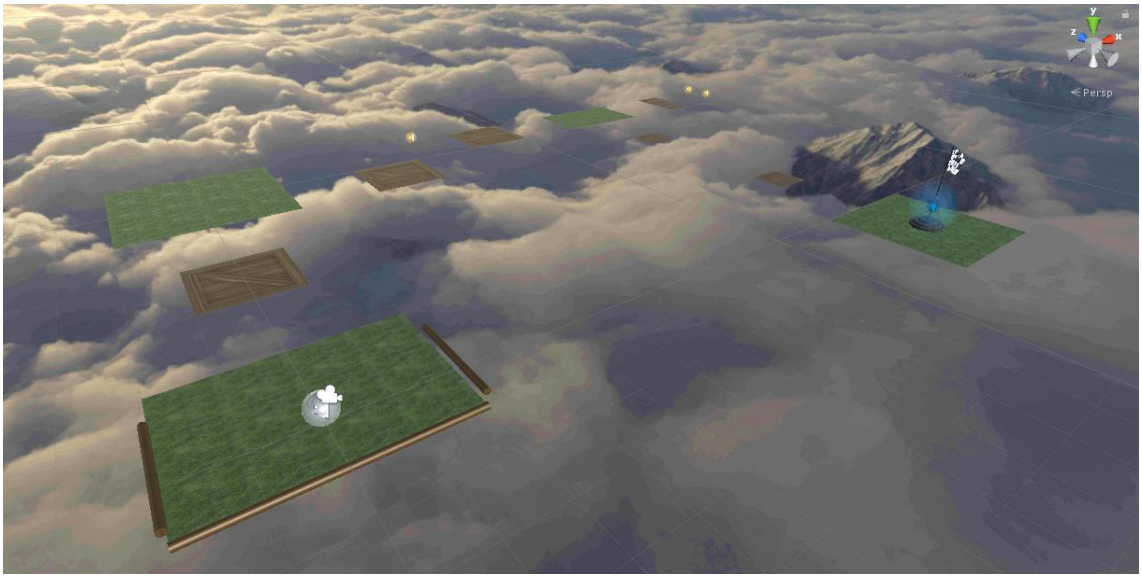


Ilustración 10.47 Nivel 2-2 vista en perspectiva

10.6.3 - Nivel 2-3

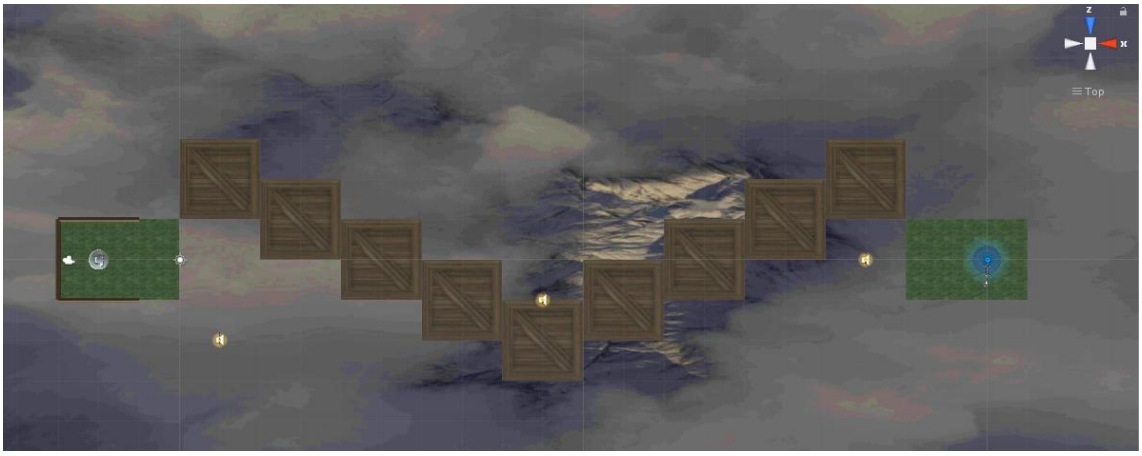


Ilustración 10.48 Nivel 2-3 vista en proyección

En este nivel se ha reducido la velocidad de las plataformas, de forma que tardan tres segundos y medio en recorrer su trayectoria, que es la misma distancia que las plataformas del nivel anterior. Se ha eliminado el tiempo de espera al llegar al final de la trayectoria, y se han establecido diferentes tiempos de desfase para las plataformas y se muevan de forma escalonada.

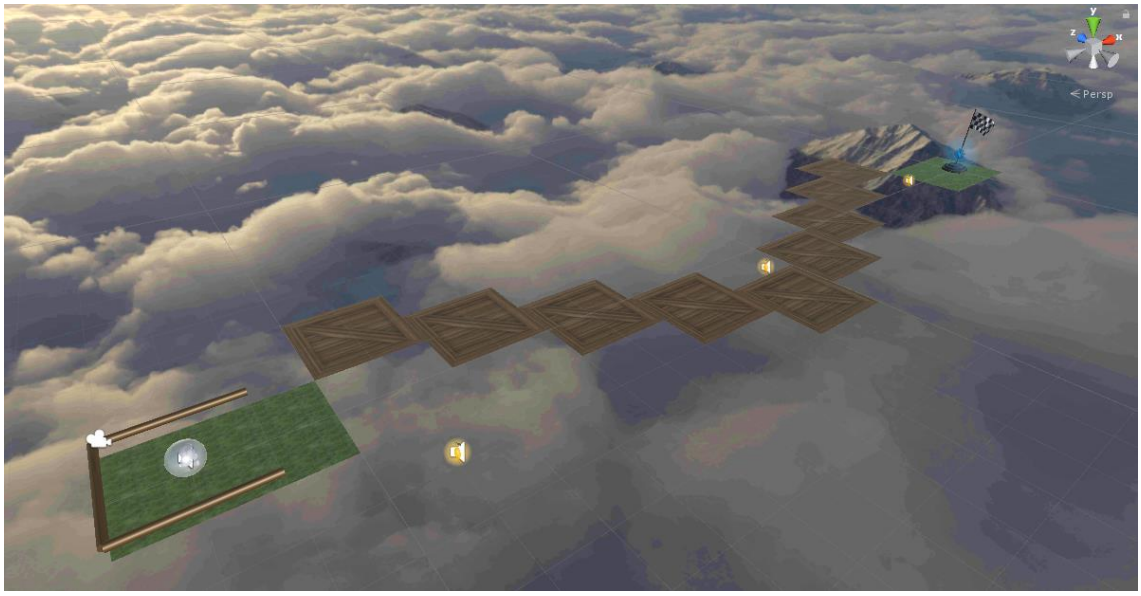


Ilustración 10.49 Nivel 2-3 vista en verspectiva

10.6.4 - Nivel 2-4

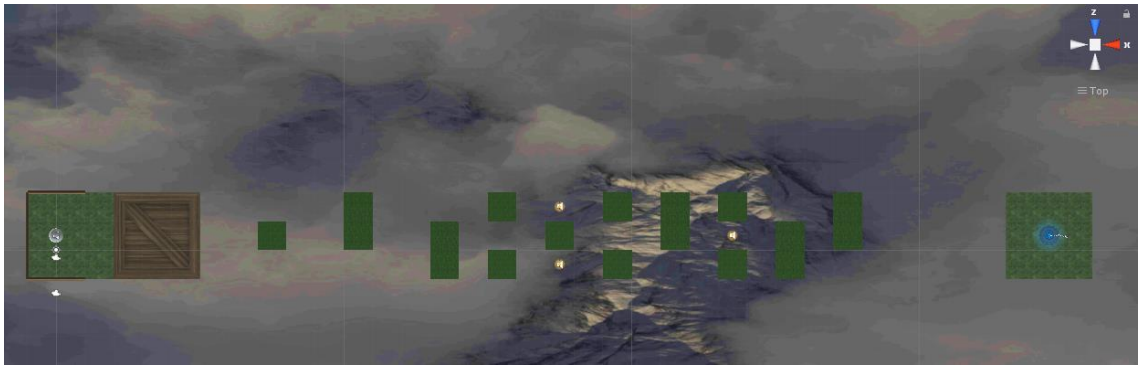


Ilustración 10.50 Nivel 2-4 vista en proyección

Este nivel cuenta con una única plataforma móvil que tarda 20 segundos en llegar a su destino, espera 1 segundo al llegar a los extremos y tiene 4 segundos de desfase inicial para permitir al jugador subirse a la plataforma.

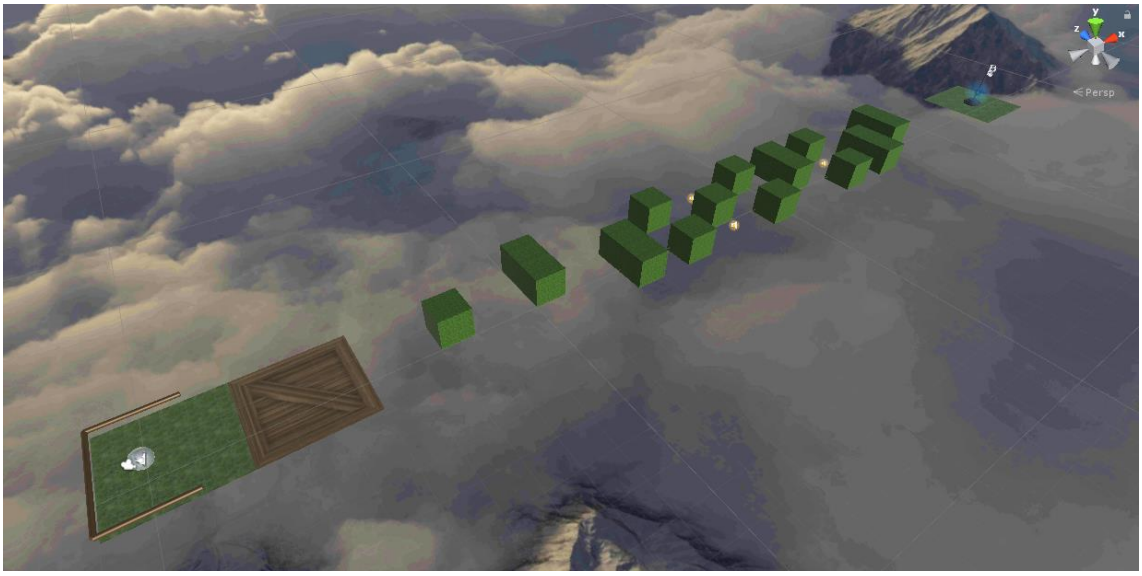


Ilustración 10.51 Nivel 2-4 vista en perspectiva

10.6.5 - Nivel 2-5

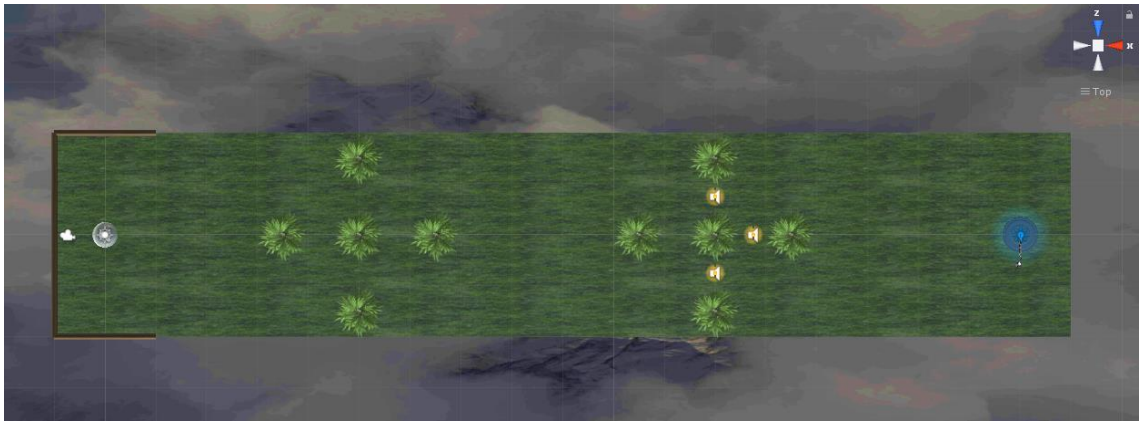


Ilustración 10.52 Nivel 2-5 vista en proyección

En este nivel se tienen dos estructuras formadas por objetos que hacen que la bola rebote cuando impacta contra ellos y que giran a alta velocidad en distintos sentidos.

Para lograr que los objetos giren respecto a un punto concreto se crea un objeto vacío y se le añade un *script* que se ha creado llamado *Rotating*. En este *script*, para cada nuevo fotograma se modifica la rotación local, es decir, la rotación relativa al padre, para modificar el valor de rotación en el eje y en función del tiempo transcurrido respecto al último fotograma y la velocidad de rotación deseada. Es necesario que se modifique la rotación local en lugar de la absoluta, ya que, en las escenas con la cámara en la espalda, se rota toda la escena, provocando que no coincidan el eje y del mundo con el eje y local y produciéndose así giros no deseados en otros ejes.

Este *script* se aplica a un objeto vacío ubicado en la posición del eje de rotación y se agregan como hijos todos los elementos que se quiera que roten a su alrededor. Para que las estructuras giren en sentido contrario sólo hay que indicar una velocidad negativa. En este nivel se ha usado los valores de velocidad de 75 y -75 respectivamente.

Para conseguir que los objetos reboten primero hay que añadirles un *collider* para que se produzcan las colisiones. A este *collider* se le añade un material físico con alto índice de rebote. Sin embargo, este rebote no es suficiente, por lo que se recurre a otras técnicas para lograr un rebote mayor. Se ha creado un *script* llamado *Bumper*, que se añade al objeto rebotador y que, cuando detecta una colisión aplica

una fuerza en forma de impulso en la dirección del vector formado por la unión de las posiciones del objeto rebotador y la bola respectivamente.

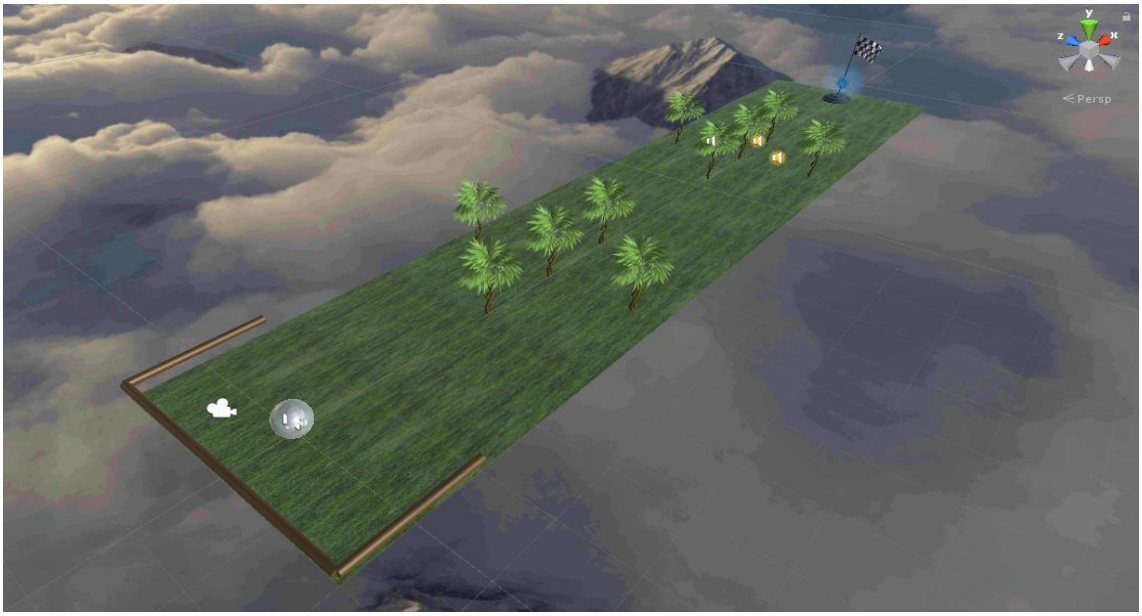


Ilustración 10.53 Nivel 2-5 vista en perspectiva

10.6.6 - Nivel 2-6

Ilustración 10.54 Nivel 2-6 vista en proyección

Este nivel se ha construido de forma similar a la explicada en el nivel anterior, con la diferencia de que se ha incrementado la velocidad de rotación de las estructuras a 100 y -100, y se han añadido otros obstáculos estáticos.

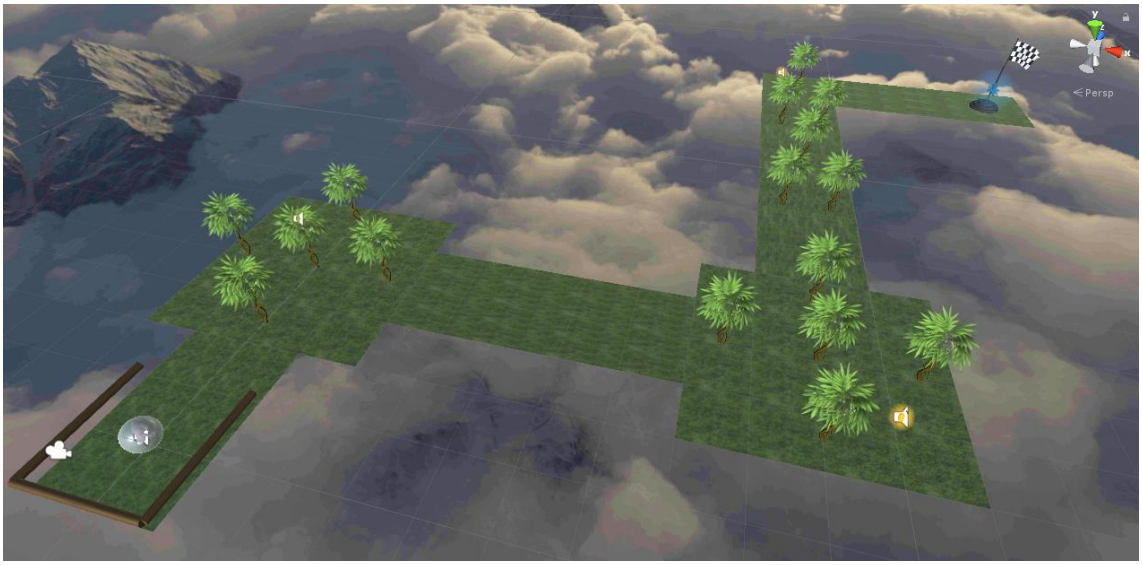


Ilustración 10.55 Nivel 2-6 vista en perspectiva

10.6.7 - Nivel 2-7

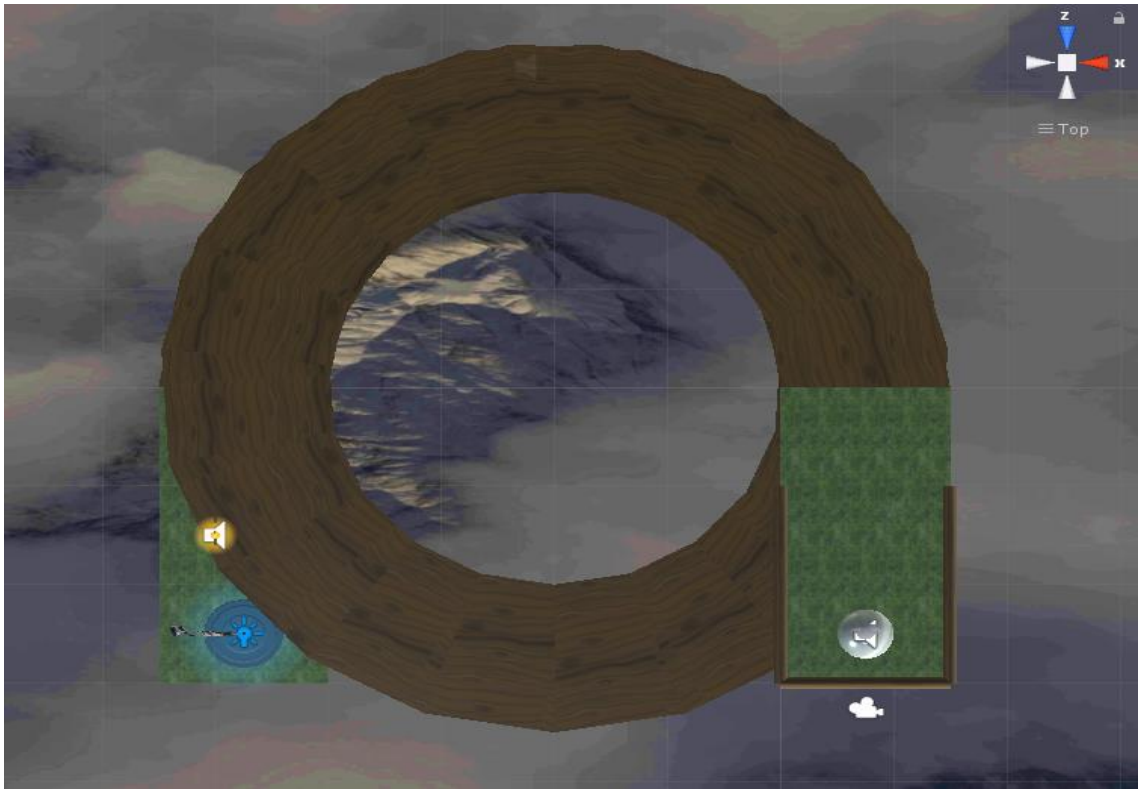


Ilustración 10.56 Nivel 2-7 vista en proyección

La escalera de caracol de este nivel ha sido creada gracias a la herramienta “ProBuilder” que se explicó anteriormente. Esta herramienta incluye una opción para crear escaleras, y proporciona la posibilidad de aplicarles cierta curvatura, definir el ancho de los escalones, la altura de la escalera y el número de escalones. Para lograr la forma final se han concatenado dos escaleras con 270 grados de curvatura cada una, y 1.5 unidades de ancho.

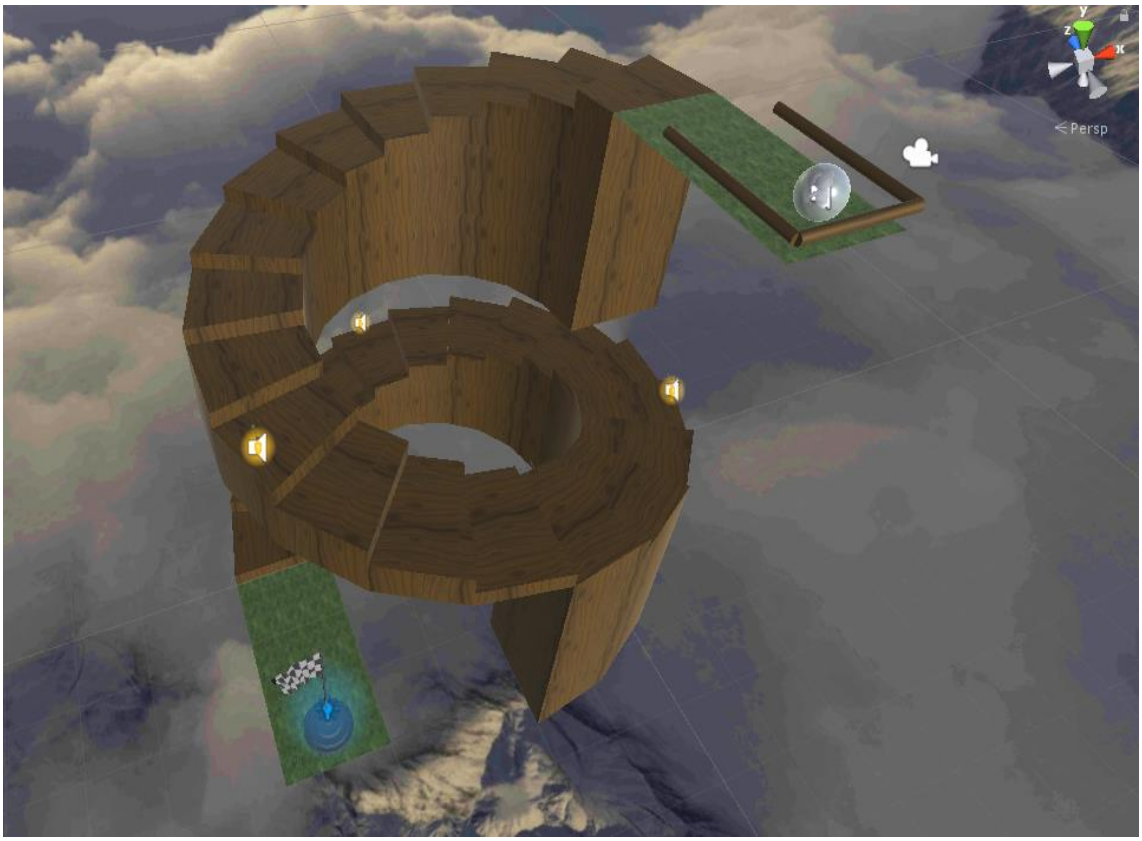


Ilustración 10.57 Nivel 2-7 vista en perspectiva

10.6.8 - Nivel 2-8

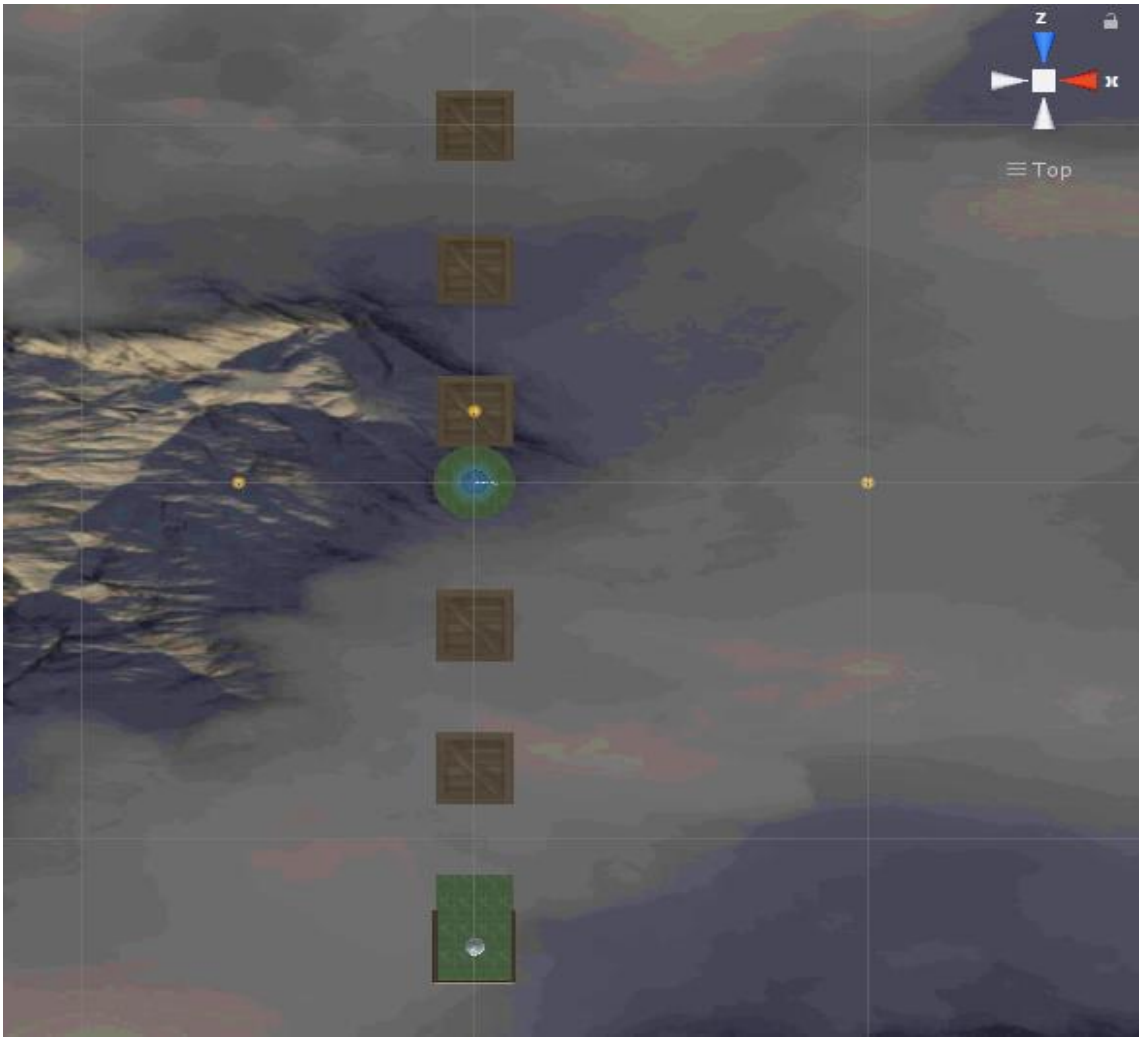


Ilustración 10.58 Nivel 2-8 vista en proyección

Para construir este nivel se han creado 5 objetos vacíos ubicados en el centro de la escena, cada uno con una plataforma y con un *script* para hacer que roten. La velocidad de rotación varía para cada plataforma, y va aumentando desde los 15 hasta los 35. Las plataformas adyacentes se mueven en sentidos opuestos, es decir, las velocidades van alternando su signo entre positivo y negativo.

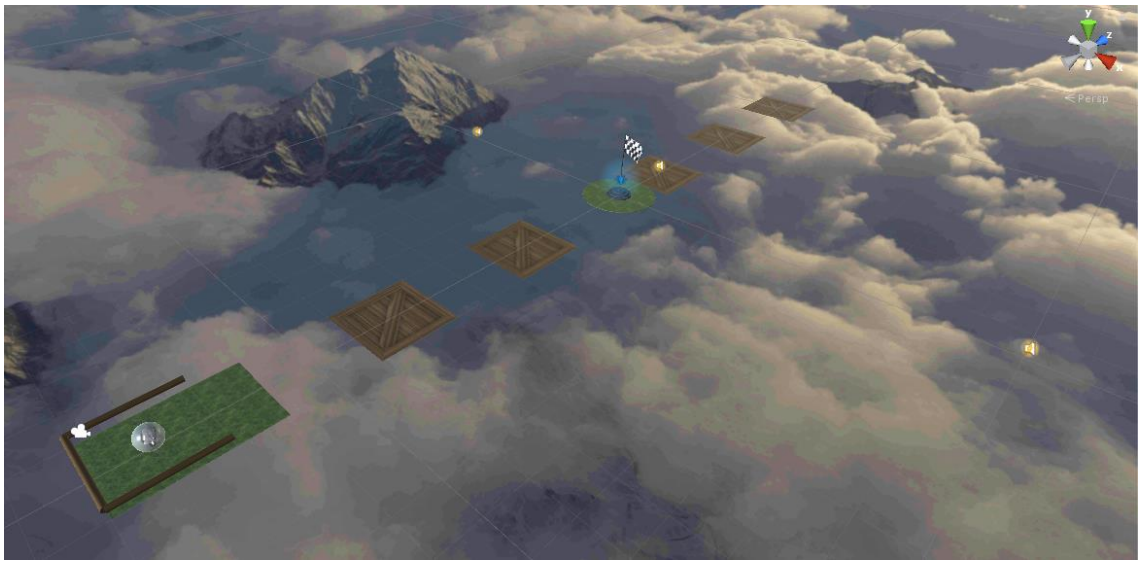


Ilustración 10.59 Nivel 2-8 vista en perspectiva

10.6.9 - Aspectos generales

Tras realizar algunas pruebas se han corregido algunos errores por los cuales algunas plataformas se superponían y provocaban que las texturas superpuestas parpadeasen.

Con la finalización de esta iteración se considera que el prototipo está terminado y se enviará una versión del juego a los probadores para comprobar sus impresiones.

Gracias a los comentarios de los probadores se han podido detectar y corregir otros fallos. Por ejemplo, se ha ampliado la zona abarcada por el *collider* utilizado para detectar las caídas en el nivel 2-1, y se ha solucionado un error que impedía que se modificase el volumen de la música del menú de inicio

11 - COMERCIALIZACIÓN

Una vez se tenga el producto totalmente terminado se pretende lanzarlo en la Google Play Store como un juego “Free to Play”, y cuyos beneficios provengan principalmente de los anuncios incorporados en la aplicación. Estos anuncios se mostrarán en forma de videos omitibles al salir de un nivel o cada dos muertes.

También puede ser una opción razonable incorporar al juego un sistema de vidas limitadas que se renueven cada cierto tiempo y que pueda acelerarse la recuperación de vidas mediante compras dentro de la aplicación.

El juego tendría un total de 10 vidas. Al morir se pierde una vida. Cuando el usuario llegue a cero vidas no podrá seguir jugando. Para recuperar una vida deberá esperar una hora, o comprar un paquete de vidas. De forma alternativa, también es posible recuperar una vida viendo un video de forma íntegra. También existe la posibilidad de mejorar el juego a una versión premium, en la que no existe límite de vidas y se elimina totalmente la publicidad. Los precios de las microtransacciones se recogen en la Tabla 11.1.

Producto	Precio
Paquete 10 vidas	2,00€
Vidas infinitas durante 1 hora	5,00€
Actualizar a versión premium	15,00€

Tabla 11.1 Tabla de precios de las transacciones dentro del juego

Por lo general para que la monetización proveniente de los anuncios sea rentable se necesita una gran base de usuarios, pues no todos los usuarios del juego van a hacer clic sobre los anuncios, que suele ser la forma en la que se obtiene el dinero. El uso de videos como medio para reclamar recompensas favorece la visión que los usuarios pueden tener del juego, ya que no es considerada como una publicidad tan invasiva, tal y como se ve en la Ilustración 11.1.

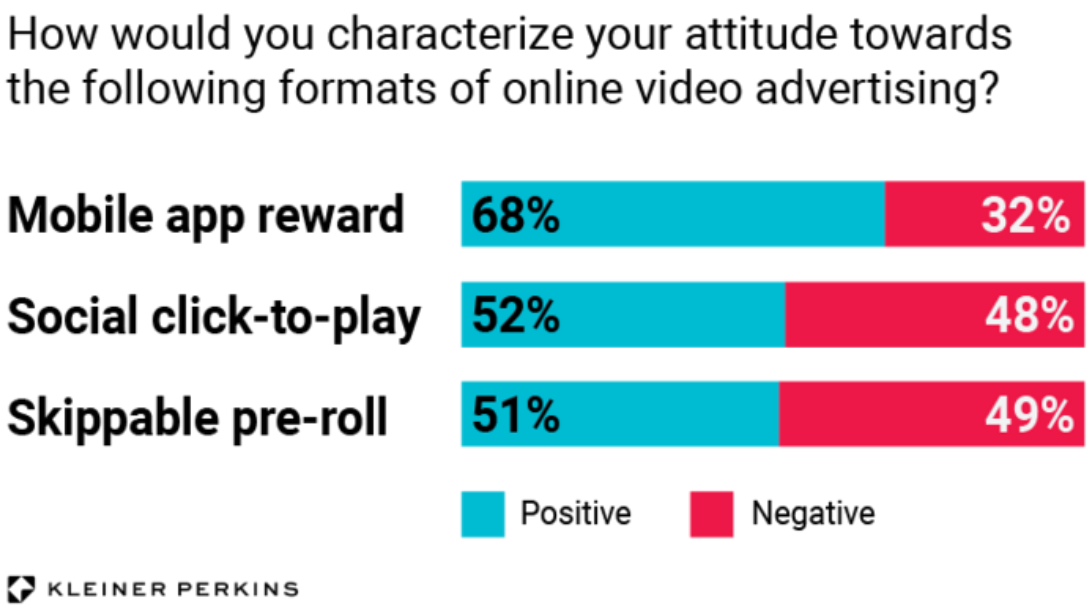


Ilustración 11.1 Resultados de encuesta a usuarios sobre su actitud frente a distintos tipos de publicidad en videojuegos

Unity cuenta con varias herramientas para facilitar la monetización de los juegos y para ofrecer anuncios acordes con los gustos de los usuarios del juego. Una de estas herramientas es Unity Ads [45]. Con esta herramienta es posible aumentar las ganancias por cada visualización de un anuncio, ya que sólo se mostrarán videos sobre juegos, por lo que se aumenta la garantía de que el video resulte interesante para el usuario. El uso de videos más interesantes también favorece a los anunciantes, lo que conlleva que aumente el valor del emplazamiento publicitario, y consecuentemente se aumentan las ganancias para el desarrollador. Además, Unity Ads favorece la consecución del anunciante que más dinero ofrece por mostrar su publicidad debido al uso de subastas unificadas en múltiples redes.

De acuerdo con los foros de Unity [46], el eCPM (Effective Cost per Mile), es decir, el beneficio medio que obtiene el juego por cada 1000 inicios de reproducción de los videos publicitarios, se sitúa por lo general entre los $6 y los $12. Para realizar los cálculos de rentabilidad del producto se tomará como aproximación un valor de 9€ como eCPM.

Con una estimación rápida se puede suponer que el presupuesto final para desarrollar el juego al completo sea del doble de lo presupuestado para este prototipo, lo que supondría un total de 16.307,50€. Sin embargo, para asegurar una cantidad de

usuarios constantes, y evitar que estos abandonen el juego en cuanto lo completen, podría ser necesario ir creando nuevos mundos de forma regular, con nuevos niveles que mantengan la base de usuarios activa.

Dado que el videojuego contará con 32 niveles en su salida y que se produce al menos una visualización de un anuncio por cada nivel, podemos considerar que cada usuario será responsable de aproximadamente 32 visualizaciones de publicidad. Para que el juego fuera rentable únicamente en base a los anuncios, se requerirían aproximadamente 1.811.944 visualizaciones de videos, que, según las anteriores estimaciones, se corresponderían con 60.399 usuarios.

Si a los ingresos generados por los anuncios les sumamos los ingresos generados por las microtransacciones, el número de usuarios requeridos descendería notablemente. Si consideramos el gasto medio por usuario como 2€ (precio del paquete de 10 vidas), el número de usuarios necesarios para amortizar el juego se establece en 7.128.

12 - CONCLUSIONES Y TRABAJO FUTURO

La realización de este proyecto ciertamente ha contribuido en la adquisición de una experiencia valiosa para el alumno, así como para mejorar las capacidades del mismo en múltiples áreas. Un proyecto para la elaboración de un prototipo de videojuego como el presente supone sin duda un desafío multidisciplinar, en el que se ponen a prueba conocimientos sobre programación, sistemas gráficos, diseño de algoritmos, diseño de interfaces, gestión de proyectos, etc. Además, la autonomía con la que se desarrolla el proyecto, y la ausencia de un guion o planificación previa como sucede con las prácticas de las asignaturas (debido a que es una de las tareas que debe realizar el alumno a lo largo del proyecto) supone una adquisición de responsabilidad fundamental de cara al futuro del alumno.

A lo largo del proyecto también se ha desarrollado la creatividad e imaginación del alumno para afrontar los desafíos encontrados de la mejor forma posible y diseñar los desafíos a los que se enfrentarán los usuarios del juego.

Este proyecto consiste en la elaboración de un prototipo, el cual se puede utilizar como base para construir el juego final. Sin embargo, antes de la comercialización del juego será necesario completar una serie de tareas que se procederán a listar a continuación:

- Obtener modelos de mayor calidad para el interior de la bola y añadirle las animaciones necesarias para dar la sensación de que son los personajes del interior de la bola los que hacen que esta se mueva.
- Incorporar un sistema de recompensas en el que emplear las estrellas obtenidas. Siendo estas recompensas de carácter estético y pueden comprender tanto los nuevos modelos para el interior de la bola como nuevas texturas para la bola o cambios en la estética de los niveles (en lugar de la estética selvática se podría desbloquear una estética futurista o una estética medieval).
- Refinar los movimientos de cámara para la perspectiva con cámara a la espalda, con el objetivo de lograr una mayor suavidad en los giros de cámara. En concreto se ha pensado incluir un algoritmo por el cual la velocidad con la que la cámara gira para adaptarse a la nueva trayectoria de la bola dependa de la velocidad que lleva la bola en ese instante. De esta

forma, cuando la bola se mueve despacio hacia un lado, la cámara gira despacio, pero si la bola hace un cambio de dirección más brusco la cámara tardará menos tiempo en adquirir el nuevo ángulo de visión.

- Mejorar el rendimiento y la optimización del juego. Esto puede ser especialmente importante debido a las limitaciones inherentes a la plataforma para la que se ha planteado el juego: el dispositivo móvil.
- Añadir soporte para varios idiomas (idealmente inglés, francés, italiano, portugués y alemán).
- Mejorar el soporte para dispositivos móviles de distintas resoluciones.
- Añadir nuevos mundos con nuevas mecánicas que mantengan el juego fresco. Se deberían añadir al menos dos mundos nuevos antes de su salida al mercado.
- Adaptar el juego para otras plataformas, como IOS, consolas y PC.

13 - ANEXOS

13.1 - Informe de los probadores

13.1.1 - Versión 0.1

Versión	0.1
Nombre	Probador A
Fecha	25 Junio 2019

Comenta tus impresiones sobre la interfaz, tanto a nivel estético como funcional:

A nivel estético se ve poco atractiva y a nivel funcional bastante completa e intuitiva.

Comenta tus impresiones sobre el sistema de movimiento y los tipos de controles. ¿Cuál de ellos te ha gustado más? ¿Cuál de ellos te ha parecido más cómodo?

A nivel de entretenimiento es más divertido el modo con el giroscopio ya que proporciona una experiencia más inmersiva, pero en cuanto a comodidad el modo joystick da un mejor control sobre el movimiento de la bola

¿Qué aspectos son los que más te han gustado del juego?

El movimiento con el giroscopio, me parece innovador.

¿Qué aspectos son los que menos te han gustado del juego?

El diseño de la interfaz

¿Qué aspectos cambiarías del juego?

Haría un poco más suave la ayuda del movimiento de cámara cuando choca con algún objeto para posicionarse detrás.

Comenta cualquier bug o problema que hayas encontrado en el juego.

Bug en los giros iniciales con el modo del giroscopio nada más empezar.

Versión	0.1
Nombre	Probador B
Fecha	25 Junio 2019

Comenta tus impresiones sobre la interfaz, tanto a nivel estético como funcional:

Está bien, queda mejor una interfaz simple y poco cargada como la que tiene que hacer algo muy cargado.

Comenta tus impresiones sobre el sistema de movimiento y los tipos de controles. ¿Cuál de ellos te ha gustado más? ¿Cuál de ellos te ha parecido más cómodo?

A mí me parece más disfrutable el control táctil en vez del control por movimiento del móvil porque tienes más control de lo que pasa en la pantalla y creo que es mejor que tener que ir moviendo el móvil. El otro control funciona bien también, pero hay que adaptarse a él e igual lleva más tiempo si nunca has jugado a juegos así.

¿Qué aspectos son los que más te han gustado del juego?

Es simple y funciona bien para pasar el rato, igual es demasiado fácil una vez lo pillas, pero es el primer nivel así que supongo que luego se complicará bastante más.

¿Qué aspectos son los que menos te han gustado del juego?

Igual los 5 segundos que tienes que esperar entre partida y partida es demasiado y sería mejor que la pantalla cargase en un segundo y así favorecería a que fuese más adictivo jugar una partida tras otra.

¿Qué aspectos cambiarías del juego?

La posición inicial de la bola, al ponerla en el borde al comienzo del nivel es más probable caer y se hace más tedioso jugar. Hacer algo más amplio el mapa y con obstáculos más interesantes en otros niveles. (Este para ser el “mapa tutorial” está bien)

Comenta cualquier bug o problema que hayas encontrado en el juego.

Cuando caes cerca de la meta, la bola continúa rodando hacia abajo infinitamente, tienes que forzar el cierre de la app para volver a empezar la partida.

Versión	0.1
Nombre	Probador C
Fecha	25 Junio 2019

Comenta tus impresiones sobre la interfaz, tanto a nivel estético como funcional:

La interfaz a nivel estético me parece más que aceptable, ya que recuerda a otros juegos de progresión de niveles como Angry Birds o Geometry Dash, haciéndola mucho más intuitiva.

Comenta tus impresiones sobre el sistema de movimiento y los tipos de controles. ¿Cuál de ellos te ha gustado más? ¿Cuál de ellos te ha parecido más cómodo?

El control por joystick me ha gustado más, es mucho más cómodo para el jugador y con él se presenta un mayor manejo de la bola. Por el contrario, el control por giroscopio me resulta más incómodo y difícil de manejar, por no hablar de que tienes que calibrar muy bien la posición y la más mínima desviación afecta a la movilidad.

¿Qué aspectos son los que más te han gustado del juego?

Por ahora lo que más me ha gustado es la interfaz y la obtención de logros.

¿Qué aspectos son los que menos te han gustado del juego?

La jugabilidad con giroscopio y la escasez de pruebas y enemigos.

¿Qué aspectos cambiarías del juego?

La jugabilidad con giroscopio la haría mucho más cómoda o al menos probarla a una menor velocidad y con un circuito más largo.

Comenta cualquier bug o problema que hayas encontrado en el juego.

Al chocar contra objetos se aprecia un cambio de pantalla algo más brusco durante décimas de segundo.

Versión	0.1
Nombre	Probador D
Fecha	25 Junio 2019

Comenta tus impresiones sobre la interfaz, tanto a nivel estético como funcional:

Se trata de una interfaz limpia y simple, que indica de manera directa los elementos básicos que hay en cualquier videojuego (trofeos, jugar y ajustes), aunque, considero que lo hace con demasiada simpleza. En primer lugar, por el diseño de los objetos en sí, y en segundo por el uso del color en la pantalla de inicio, que es demasiado saturado y no utiliza una paleta de color homogénea (que podría compensar esa potencia visual y que, además, daría una mayor identidad al juego.

Cuando tenga algo más de desarrollo podrían añadirse elementos estéticos a la pantalla sin obstaculizar la visión central y periférica de la esfera y para así embellecerlo.

Comenta tus impresiones sobre el sistema de movimiento y los tipos de controles. ¿Cuál de ellos te ha gustado más? ¿Cuál de ellos te ha parecido más cómodo?

Al iniciar una partida vemos la esfera caer mientras sale una cuenta atrás, creo que es redundante y que se podría plantear otro sistema para dar el aviso al jugador, como, por ejemplo, que tras caer aparezca un brillo que de la señal o que directamente aparezca quieta con la cuenta atrás.

Los controles son sencillos y cumplen el objetivo de mover la bola por el escenario hasta llegar a la meta. Sin embargo, podría ser un buen punto para cuestionarse si equilibrar la pantalla para que la bola se mueva es la forma de juego más cómoda y apetecible para cualquier jugador, pues implica estar en una posición concreta que no siempre queremos tener, proponer opciones de postura podría ser interesante.

¿Qué aspectos son los que más te han gustado del juego?

Con el poco desarrollo del juego, puedo destacar que ha llamado especialmente mi interés que la esfera se guíe según la posición del móvil.

¿Qué aspectos son los que menos te han gustado del juego?

La simpleza de los modelos, pero que, al tratarse de una beta probablemente estará incompleta y no será el resultado final.

¿Qué aspectos cambiarías del juego?

Creo que el hecho de ser una bola que se desliza es algo bastante simple, quizás la sustitución de ese modelo por otro que le dé una temática distinta podría ser un aspecto interesante a estudiar para diferentes modos de juego o para, sencillamente, si quieres personalizarlo.

Comenta cualquier bug o problema que hayas encontrado en el juego.

La cámara tras cada colisión tiene un efecto extraño, como un cambio direccional muy brusco, que provoca una ruptura con la continuidad y suavidad que ofrece el juego.

13.1.2 - Versión 0.2

Versión	0.2
Nombre	Probador A
Fecha	04 Agosto 2019

Comenta tus impresiones sobre la interfaz, tanto a nivel estético como funcional:

La interfaz es agradable e intuitiva. Tanto a nivel estético como funcional me parece adecuada y bastante completa.

Comenta tus impresiones sobre el diseño de los niveles. ¿Demasiado cortos/largos, fáciles/difíciles, etc.?

Los niveles son bastante originales y de duración están bien. La dificultad en algunos niveles puede ser alta al estar utilizando el control por el giroscopio.

¿Qué aspectos son los que más te han gustado del juego?

El diseño de los niveles con atajos.

¿Qué aspectos son los que menos te han gustado del juego?

Algunos bugs de cámara en la que la redirige y te desorientas.

¿Qué aspectos cambiarías del juego?

Quizá la interfaz a nivel de fuego, sobre todo cuando acabas un nivel o mueres para que quede más claro.

Comenta cualquier bug o problema que hayas encontrado en el juego.

La finalización de algunas plataformas hace que se descontrole la bola, y en el nivel 8 hay una especie de rebaba en la unión de dos escalones.

Versión	0.2
Nombre	Probador B
Fecha	04 Agosto 2019

Comenta tus impresiones sobre la interfaz, tanto a nivel estético como funcional:

La interfaz está bien, el menú de los logros supongo que aún no lo has hecho y por eso sale así de descuadrado, pero por lo demás bien.

Comenta tus impresiones sobre el diseño de los niveles. ¿Demasiado cortos/largos, fáciles/difíciles, etc.?

Algunos mapas son demasiado grandes (por ejemplo, el nivel 4 creo que era), al hacerlo así pierde el componente adictivo porque si te caes a mitad da más pereza empezar de nuevo que si fuesen mapas más pequeños con el contenido más comprimido.

La dificultad está bien, aunque quizás le vendría bien al juego que fuese más difícil en el caso de que adaptes los mapas y los hagas más pequeños, si el nivel dura menos tiempo, aunque sea difícil no se hará tan frustrante porque podrás intentarlo muchas veces.

¿Qué aspectos son los que más te han gustado del juego?

Añadir las barras de los laterales puede dar bastante juego y servir para los primeros niveles para que se haga más fácil, lo de poner monedas en los extremos del mapa hace que sea difícil de pillar y tiene su gracia.

¿Qué aspectos son los que menos te han gustado del juego?

El nivel 6 en general no creo que tenga mucho sentido, no hay ninguna prueba de habilidad ni nada, es solo llegar a la meta cuando te conozcas el camino.

¿Qué aspectos cambiarías del juego?

Igual más que cambiar, añadir cosas, por ejemplo, que además de recoger monedas puedas recoger otros objetos que te hagan ir más rápido o más lento o tengan otros efectos. Podría estar bien también añadir un contador hacia atrás y si llega a 0 perder la partida.

Comenta cualquier bug o problema que hayas encontrado en el juego.

Se me quedó pillado en el menú una vez, pero en general va bien, no he encontrado bugs dentro de las partidas.

Versión	0.2
Nombre	Probador C
Fecha	04 Agosto 2019

Comenta tus impresiones sobre la interfaz, tanto a nivel estético como funcional:

Más que aceptable, mucho mejor que la de la versión 0.1. Visualmente bonita, intuitiva y estandarizada.

Comenta tus impresiones sobre el diseño de los niveles. ¿Demasiado cortos/largos, fáciles/difíciles, etc.?

Los niveles siguen una progresión, siendo el 8 el más complicado. Todos tienen en común que son más fáciles con el control por joystick. También son mucho más sencillos los que poseen la cámara desde arriba, frente a los que se ven en primera persona. Estos últimos con el giroscopio me resultan muy difíciles. Me han faltado más obstáculos en todos los niveles.

¿Qué aspectos son los que más te han gustado del juego?

La interfaz, la variedad de niveles y el control por joystick.

¿Qué aspectos son los que menos te han gustado del juego?

El control por giroscopio (sobre todo en primera persona) y la falta de elementos/obstáculos en los niveles.

¿Qué aspectos cambiarías del juego?

Los mencionados en el apartado anterior. Añadiendo elementos e intentando que sea más fácil el uso de giroscopio.

Comenta cualquier bug o problema que hayas encontrado en el juego.

No he encontrado bugs, quizás hay un pequeño cambio brusco de cámara, en el modo giroscopio, al cambiar de dirección.

Versión	0.2
Nombre	Probador D
Fecha	04 Agosto 2019

Comenta tus impresiones sobre la interfaz, tanto a nivel estético como funcional:

Una interfaz divertida y rápida que permite acceder a los contenidos rápidamente (además, de que aparecen el número total de estrellas que tenemos). En mi opinión, hay pocas opciones, pero, la idea de que se pueda activar o desactivar el control por inclinación es una mejora interesante (aunque en el juego desactivar la casilla invalida toda la jugabilidad, no hay alternativa para dejarla activada, no tiene modo alternativo de movilidad)

Para acabar, añadir que hay una unión coherente en los elementos (estética en general) que forman la interfaz que le proporcionan identidad al juego.

Dentro del juego el tiempo y el número de estrellas son muy discretos. Añadir que no cuenta con un botón para continuar una vez pausas el juego y obliga a deducir que hay que pulsar la misma opción para cerrar la pestaña.

Comenta tus impresiones sobre el diseño de los niveles. ¿Demasiado cortos/largos, fáciles/difíciles, etc.?

Los diseños son simples y se ve claramente los objetivos, aunque, utilizando el control por inclinación ganan interés. De duración y la facilidad dependen del nivel:

Primer nivel: Duración corta y bastante fácil.

Segundo nivel: Duración media y de dificultad normal.

Tercer nivel: Duración corta y se vuelve difícil por la cámara.

Cuarto nivel: Duración muy largo (aparte se vuelve aburrido porque apenas hay elementos) y salvo el final que es difícil, el resto es muy sencillo.

Quinto nivel: Duración media y es especialmente difícil por la cámara.

Sexto nivel: Duración muy larga es un nivel sencillo, aunque, como el tercero, se vuelve aburrido por la falta de estimulación.

Séptimo nivel: Aparece un elemento nuevo, un cilindro hueco, que gira y puede hacer caer nuestra esfera (aunque la rotación se intuye, puesto que le falta algún elemento que nos lo de a entender) la duración es corta, pero nuevamente la cámara aumenta la dificultad muchísimo.

Octavo nivel: La duración es larga y es más difícil de los anteriores (hace honor a ser el nivel final del mundo 1). Aparte, es el más entretenido pues

requiere más habilidad para que no caiga al “vacío” y la ausencia de detalles no se echa en falta.

En definitiva, los niveles que tienen la vista cenital son mejores con respecto al control de la bola porque la cámara no interfiere en el movimiento y te permite ver y planificar el movimiento con anterioridad (con la otra cámara al avanzar se pone demasiado abajo y es muy difícil ver lo que tenemos delante).

Además, todavía sigue faltando una estética en todo el modelado 3D del juego en general que aúne la interfaz con el juego en sí.

¿Qué aspectos son los que más te han gustado del juego?

El control por inclinación que sigue funcionando bien y tiene sensibilidad como para esquivar las caídas y los bloques (en el primer nivel).

¿Qué aspectos son los que menos te han gustado del juego?

La cámara, aún no ha sido tratada y es muy molesta a la hora de mover la esfera, además, perjudica mucho en determinados niveles que los vuelve un sufrimiento, la cámara se mueve con la parte “trasera” de la esfera y los giros bruscos o los movimientos que no sean suaves terminan mareando o desorientando.

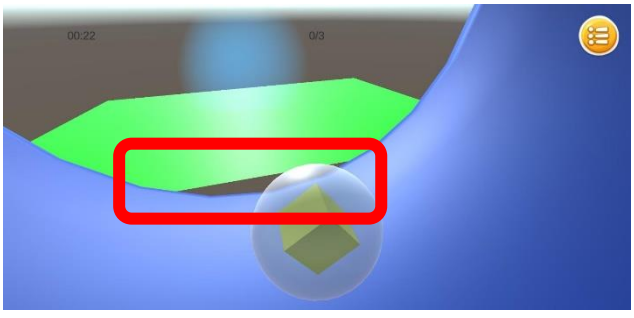
¿Qué aspectos cambiarías del juego?

Que el tipo de cámara pudiese modificarse en cualquier momento para poder tener diferentes perspectivas como, por ejemplo, una vista de cenita y sortear así los distintos obstáculos con más facilidad.

También podría ser interesante estudiar si se pudiese jugar con una cámara de seguimiento fija que no se viese alterada su visión dependiendo del control que hagamos de la inclinación.

Comenta cualquier bug o problema que hayas encontrado en el juego.

Por ahora no he detectado ninguno, los errores son más bien por la falta de desarrollo.



13.1.3 - Versión 0.3

Versión	0.3
Nombre	Probador A
Fecha	27 Agosto 2019

Comenta tus impresiones sobre la el estilo artístico del juego (gráficos y música):

El estilo artístico me parece adecuado y acorde con la idea del juego, es original y agradable.

Comenta tus impresiones sobre el diseño de los niveles. ¿Demasiado cortos/largos, fáciles/difíciles, etc.?

La duración de los niveles es adecuada, aunque la dificultad la considero un poco alta en algunos niveles especialmente usando el giroscopio.

¿Qué aspectos son los que más te han gustado del juego?

El estilo del juego y la originalidad de los niveles.

¿Qué aspectos son los que menos te han gustado del juego?

La frustración al morir repetidas veces por la dificultad.

¿Qué aspectos cambiarías del juego?

Notificar que has conseguido tal logro de alguna manera visual.

Comenta cualquier bug o problema que hayas encontrado en el juego.

Algún bug en las juntas de dos plataformas y en los choques contra los árboles que se mueven y te lanzan con tanta fuerza que se puede descontrolar.

Versión	0.3
Nombre	Probador B
Fecha 28-08-2019	27 Agosto 2019

Comenta tus impresiones sobre la el estilo artístico del juego (gráficos y música):

Simple tanto los gráficos como la música, pero están bien, el regulador de volumen de la música no funciona e igual algún sonido cuando completas un nivel estaría bien.

Comenta tus impresiones sobre el diseño de los niveles. ¿Demasiado cortos/largos, fáciles/difíciles, etc.?

A mí me han parecido difíciles, sobre todo el 4. En cuanto a los niveles, cada uno es distinto, en un nivel te pide sortear obstáculos, en otro te encontrar la salida en un laberinto, en otro te pide maniobrar para no caerte, y en otro te pide paciencia para pasar de una plataforma a otra, no sé si tu intención es que haya un poco de todo o centrarte y perfeccionar una de esas cosas. De los niveles del mundo 1 cambiaria unos cuantos que ya te dije la otra vez que eran demasiado largos, los que he podido ver del 2 en cuanto a tamaño estaban bien.

¿Qué aspectos son los que más te han gustado del juego?

Creo que es bastante adictivo o va por buen camino.

¿Qué aspectos son los que menos te han gustado del juego?

Sigo viéndolo lento, creo que nada más morir deberías poder empezar ya el nivel y no esperar 3 segundos, es poco tiempo de espera, pero cuando mueres 10 o 20 veces ya no es tan poco y te cansa. El tema de los logros igual es algo mejorable.

¿Qué aspectos cambiarías del juego?

Yo me centraría en una temática concreta, si es moverte por una plataforma pues haría los niveles de eso, si es esquivar obstáculos pues de esquivar obstáculos, en vez de tener un poco de todo.

Comenta cualquier bug o problema que hayas encontrado en el juego.

El botón para ir atrás cuando estás en la selección de nivel a veces falla. El regulador de volumen del juego no va.

Versión	0.3
Nombre	Probador C
Fecha	27 Agosto 2019

Comenta tus impresiones sobre la el estilo artístico del juego (gráficos y música):

Entorno y música agradable que fomenta un ambiente relajado para el jugador. Los gráficos son correctos, aunque quizás algo monótonos.

Comenta tus impresiones sobre el diseño de los niveles. ¿Demasiado cortos/largos, fáciles/difíciles, etc.?

La duración de los niveles es correcta y la dificultad elevada, haciendo que se distancie de jugadores casual. La dificultad baja al usar el joystick.

¿Qué aspectos son los que más te han gustado del juego?

La variedad de niveles, la interfaz y su temática.

¿Qué aspectos son los que menos te han gustado del juego?

El giroscopio ya que depende de la sensibilidad que posea el móvil y a veces te hace acabar en una posición rara mientras juegas, al intentar hacer giros bruscos.

¿Qué aspectos cambiarías del juego?

Más variedad de decorados y algún monstruo que interactúe más con la bola (perseguir, acometer…)

Comenta cualquier bug o problema que hayas encontrado en el juego.

Al finalizar el nivel 7 del mundo 1, la pantalla de inicio empezó a parpadear mostrando cargando y la propia pantalla, tuve que cerrar la aplicación. En el nivel 1 del mundo 2, al caerme en la última plataforma donde estaba la meta, no salió el mensaje de has perdido y la bola quedó atrapada en un bucle girando.

Versión	0.3
Nombre	Probador D
Fecha	27 Agosto 2019

Comenta tus impresiones sobre el estilo artístico del juego (gráficos y música):

La idea de una jungla sobre el cielo me parece interesante, y esa diferenciación se presenta más acentuada gracias a que el fondo es muy realista y a que el personaje principal está más inspirado en el estilo cartoon. Hay una separación entre ambos conceptos que enriquece el juego. Los gráficos por su parte, me parecen sencillos y muy acertados, pues, si se encontrasen muy elaborados no todos los móviles podrían tirar del juego y quizás no sería tan fluido como lo es ahora. La música no está mal, aunque considero que sería más enriquecedor que tuviese más variedad.

Comenta tus impresiones sobre el diseño de los niveles. ¿Demasiado cortos/largos, fáciles/difíciles, etc.?

Considero que los niveles del “mundo 2” son mucho más interesantes y que ponen más a prueba la habilidad del jugador que los del “mundo 1”. Se arriesga más en el segundo que en el primero con diseño de escenarios más complejos (esquivar árboles, plataformas móviles, etc.). En general mi opinión sobre el mundo 1 se mantiene en esta nueva versión, y la impresión sobre el mundo 2 puede resumirse en que la duración de los niveles es corta (unos 27 segundos aproximadamente) aunque también son más difíciles, mi opinión es global salvo en el nivel final que es más demandante y por lo tanto requiere más tiempo (unos 60 segundos) y que es mucho más difícil.

¿Qué aspectos son los que más te han gustado del juego?

La implementación de una pantalla en la interfaz que muestre la inclinación de la bola, la rapidez de carga de los niveles y la nueva estética selvática que tiene el juego.

¿Qué aspectos son los que menos te han gustado del juego?

La cámara, es molesta cuando la esfera impacta en algún objeto o cae, y eso se ve mucho en el nivel 7 del mundo 2 dónde se vuelve incomoda con tanta caída.

¿Qué aspectos cambiarías del juego?

La cámara, que cuando impacte con algún objeto no sea tan brusca y que estuviese un poco más alta para poder anticipar mejor el escenario y planificar tu movimiento.

Comenta cualquier bug o problema que hayas encontrado en el juego.

Ninguno, los problemas que tenía la anterior versión están corregidos y adaptados.

13.2 - Bocetos de los niveles

13.2.1 - Nivel 1-1

Este primer nivel será la toma de contacto de los jugadores con el juego y por lo tanto debe ser algo más simple y seguro. Para ello se han colocado barreras en toda la primera mitad del nivel, lo que supone un colchón de seguridad a la hora de afrontar los primeros giros. Los siguientes giros son más complicados, pues los obstáculos aumentan en tamaño y no hay barreras. Los tres coleccionables se situarán por detrás de la línea de meta, obligando al jugador a que de una vuelta por una plataforma sin barreras para obtenerlos.

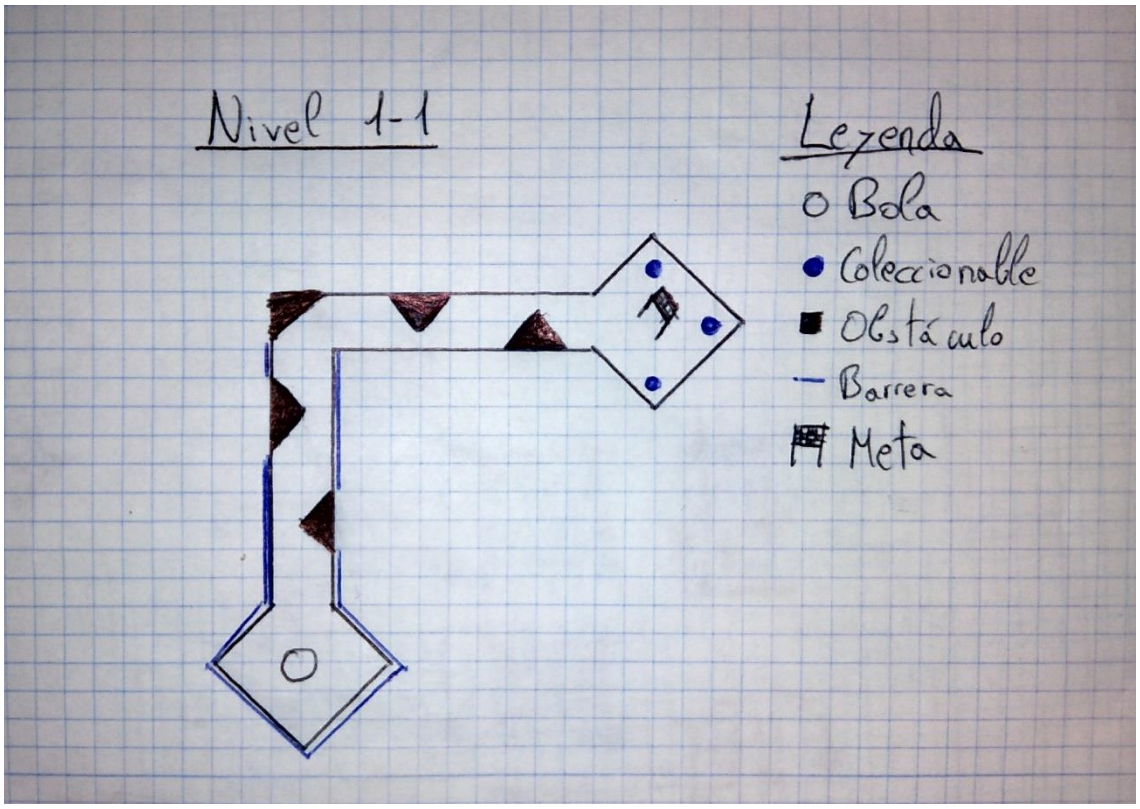


Ilustración 13.1 Boceto nivel 1-1

13.2.2 - Nivel 1-2

En este nivel se introduce el sistema de movimiento con vista cenital. El inicio del nivel cuenta con barreras para ayudar a que el jugador tome confianza. El espacio por el que la bola debe pasar se va reduciendo conforme se avanza en el nivel. Intentar

coger los coleccionables implica realizar los giros de una forma menos segura al obligar al jugador a desplazarse más cerca del precipicio.

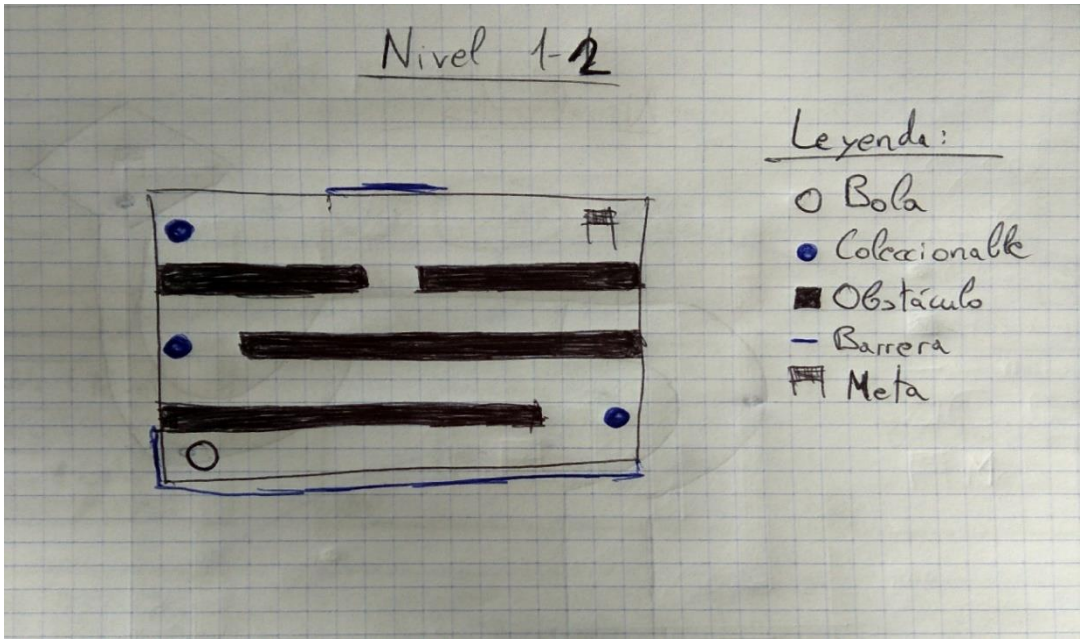


Ilustración 13.2 Boceto nivel 1-2

13.2.3 - Nivel 1-3

Nivel algo más complicado, con barreras solo al inicio y trazado con curvas.

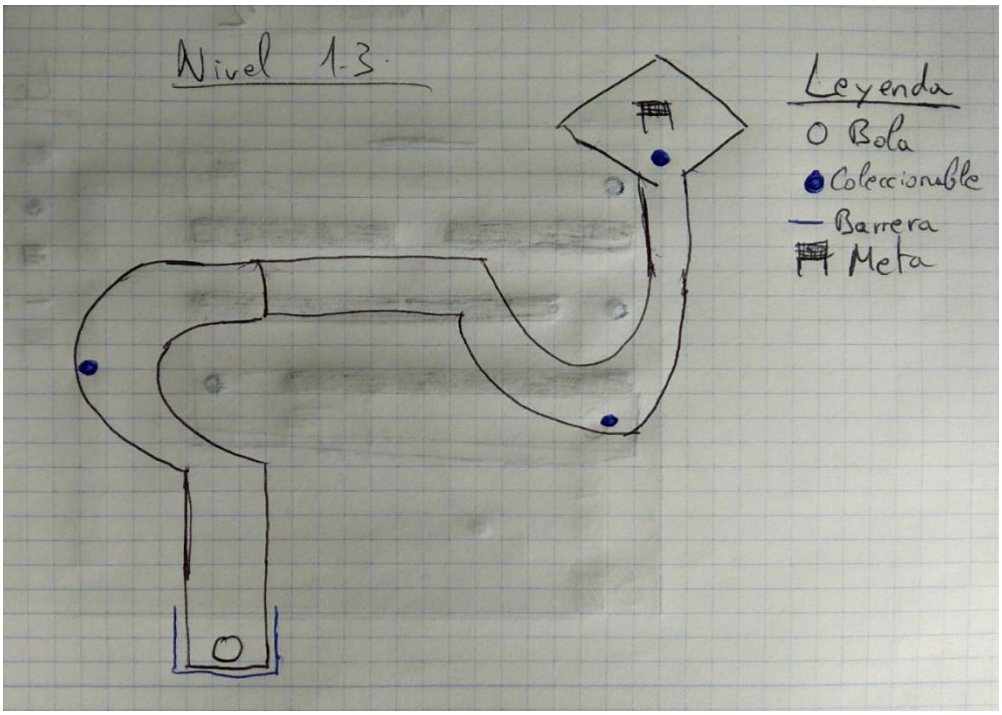


Ilustración 13.3 Boceto nivel 1-3

13.2.4 - Nivel 1-4

Trazado algo laberíntico que obligará al jugador a recorrer una espiral con cada vez menos barreras.

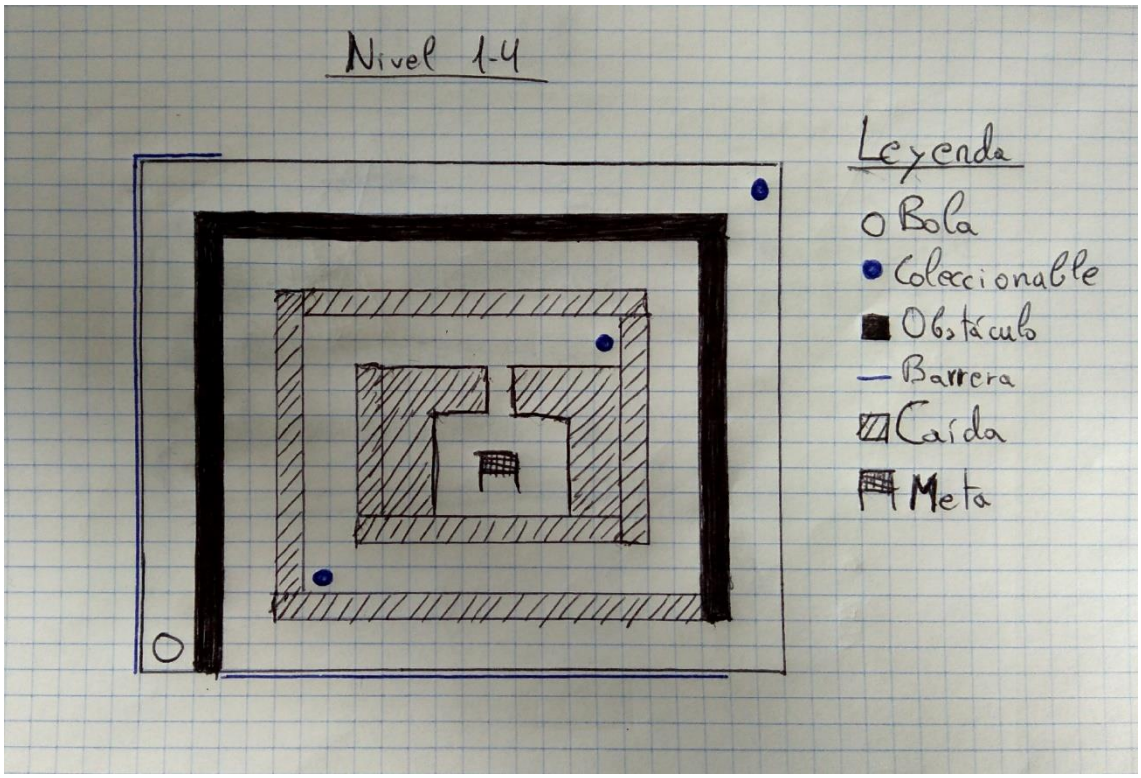


Ilustración 13.4 Boceto nivel 1-4

13.2.5 - Nivel 1-5

Nivel con una serie de giros marcados que incluye una pequeña sección de recorrido opcional para recoger los recolectables y que conduce nuevamente al usuario al trazado principal.

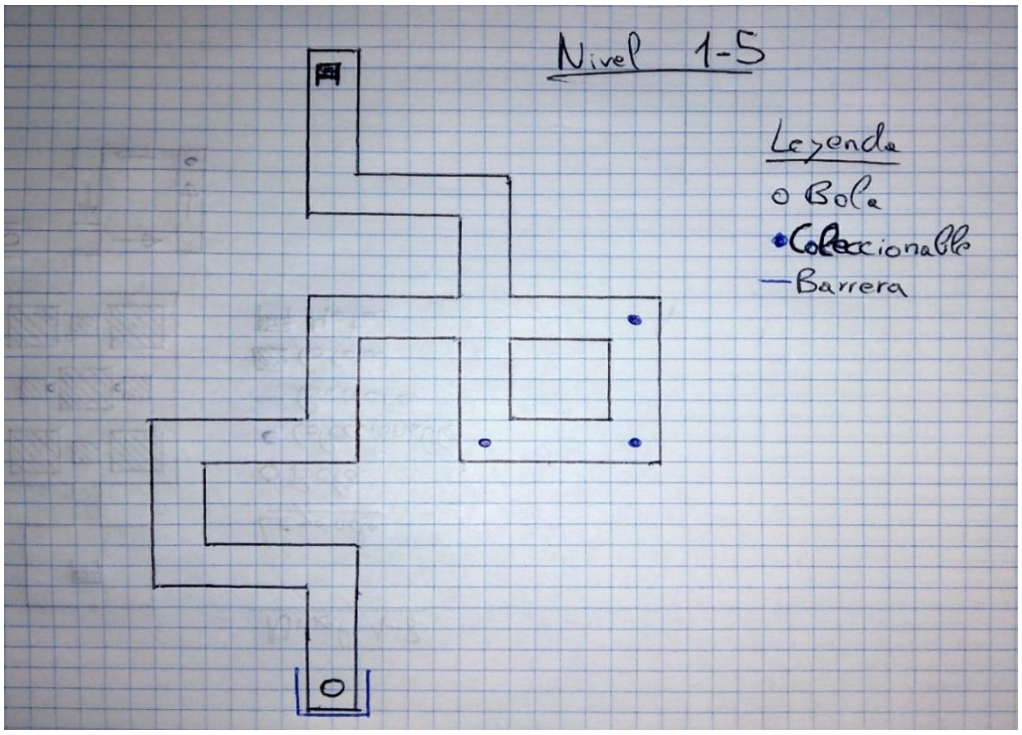


Ilustración 13.5 Boceto nivel 1-5

13.2.6 - Nivel 1-6

Un nivel laberíntico, en el que la cámara se ubicará más próxima al jugador para reducir así su campo de visión y hacer que el laberinto sea algo más complicado de recorrer.

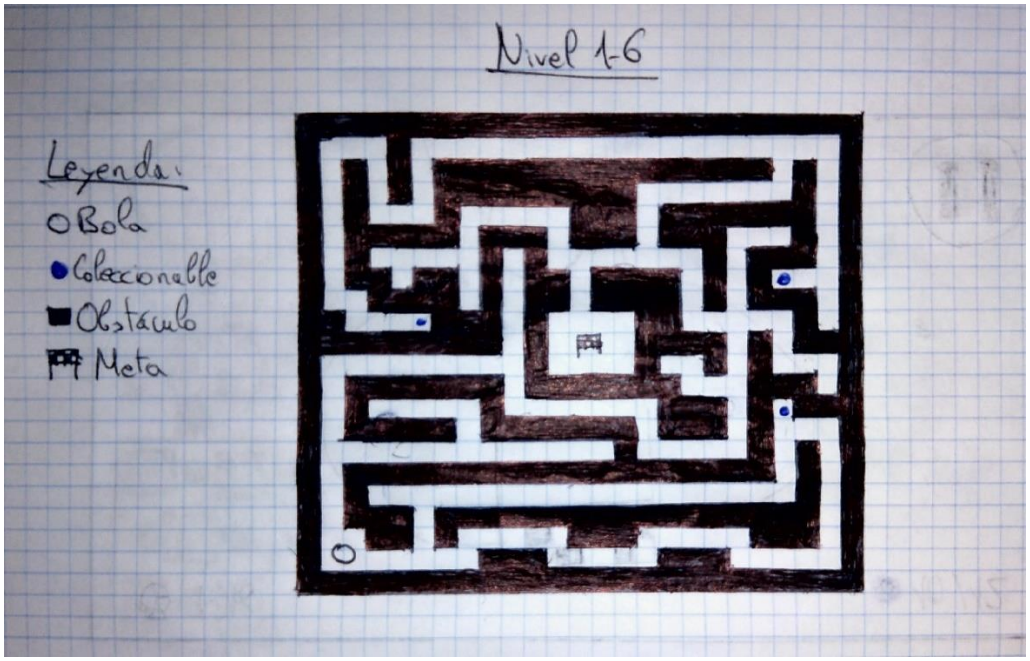


Ilustración 13.6 Boceto nivel 1-6

13.2.7 - Nivel 1-7

En este nivel se introducen las plataformas a distintas alturas, empezando en el nivel superior. El jugador debe cruzar un puente cilíndrico para llegar a otra plataforma, desde la que podrá descender mediante rampas al nivel inferior. En el nivel inferior deberá volver a cruzar el puente, pero en esta ocasión lo atraviesa por dentro para llegar a la plataforma en la que está la meta. Para conseguir batir el récord de tiempo será necesario dejarse caer desde el puente al nivel inferior en lugar de bajar por las rampas. Esto supondrá un reto para el jugador tanto en la concepción de la idea como en su ejecución que será recompensado con la estrella de batir el récord de tiempo del nivel.

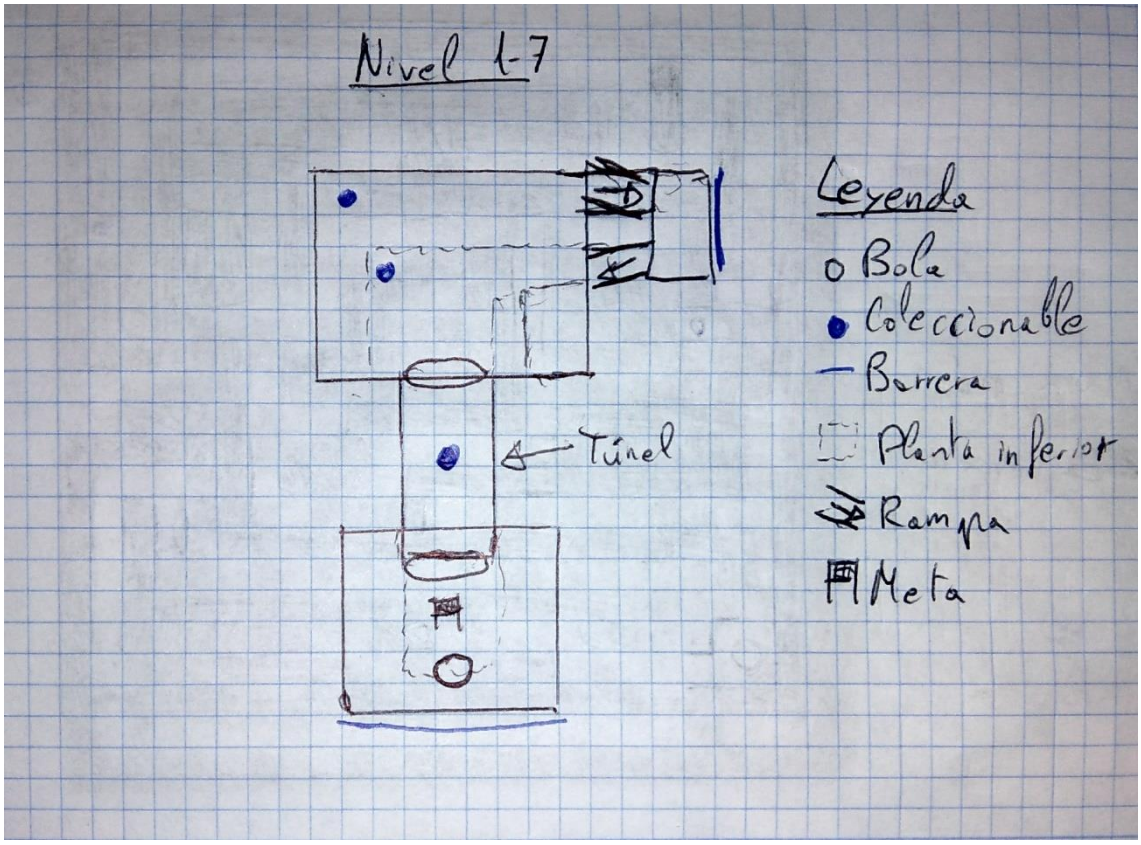


Ilustración 13.7 Boceto nivel 1-7

13.2.8 - Nivel 1-8

En este nivel se vuelven a usar plataformas a distintas alturas. El jugador parte desde una posición elevada y para descender podrá optar por tirarse desde un hueco existente entre las barreras o bajar por las rampas, que esta vez no tendrán barreras

de apoyo. Al descender se llegará a una plataforma plagada de agujeros con múltiples caminos por los que recorrerla.

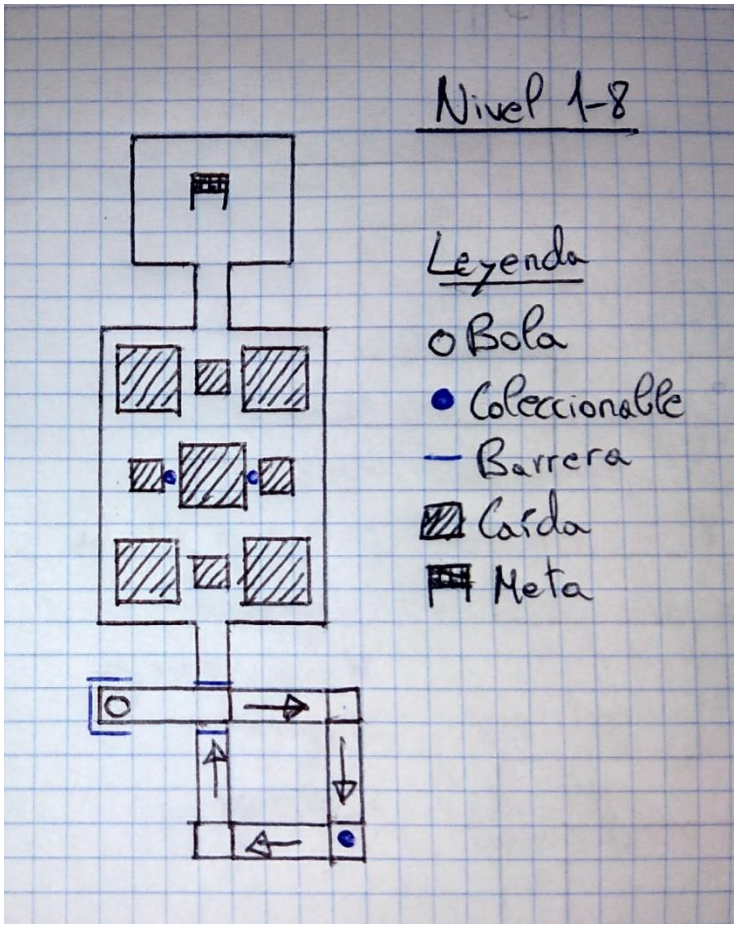


Ilustración 13.8 Boceto nivel 1-8

13.2.9 - Nivel 2-1

Este nivel se basa en una serie de plataformas inclinadas, pero a diferencia de los niveles anteriores la inclinación no tiene la misma orientación que la dirección del movimiento de la bola, sino que las plataformas están inclinadas hacia un lado de forma horizontal.

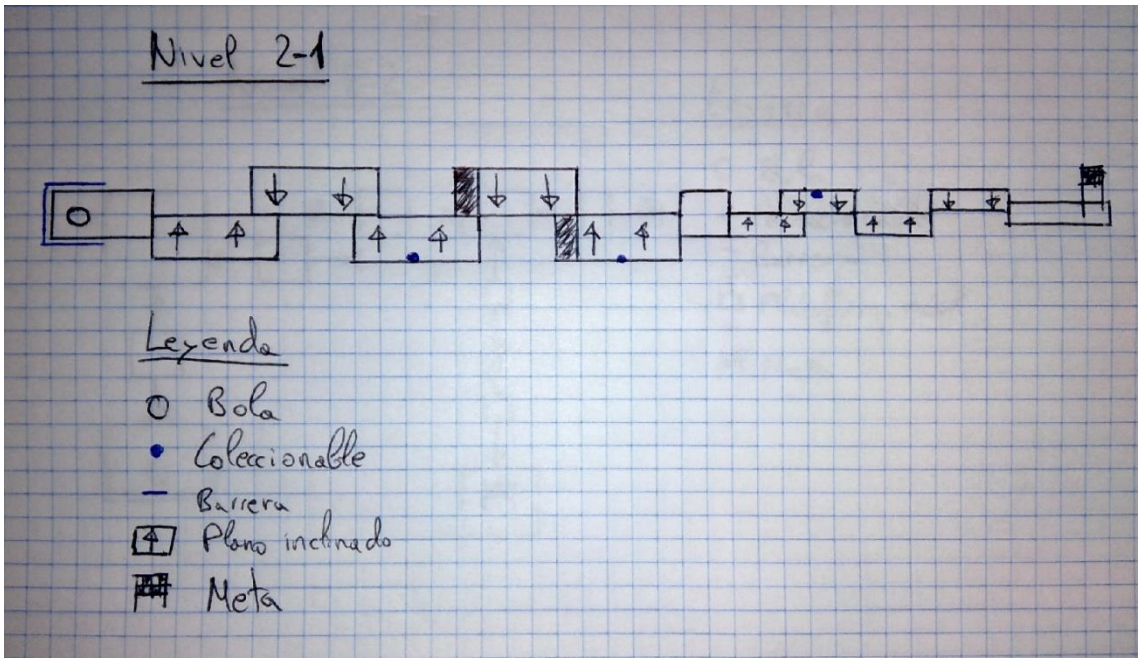


Ilustración 13.9 Boceto nivel 2-1

13.2.10 - Nivel 2-2

En este nivel se introducen las plataformas móviles. Estas plataformas se mueven siguiendo un recorrido prefijado. Al llegar al final del recorrido se espera un tiempo determinado y después vuelve siguiendo el mismo recorrido. Cuando la bola está subida sobre una plataforma, la bola adquiere el movimiento de la plataforma, es decir, no será necesario moverse para mantenerse dentro de la plataforma.

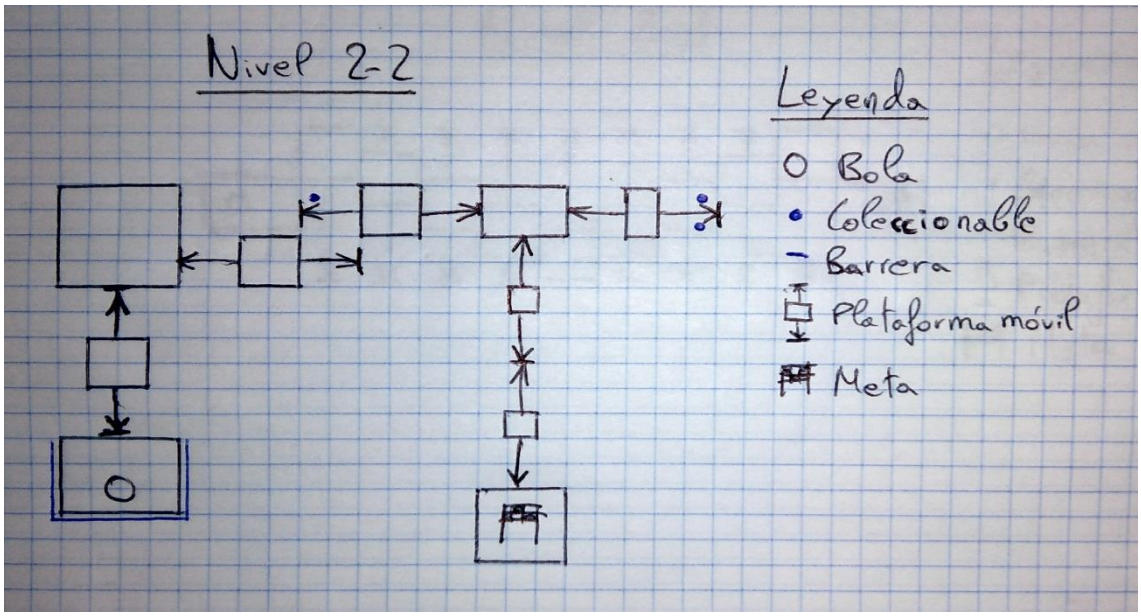


Ilustración 13.10 Boceto nivel 2-2

13.2.11 - Nivel 2-3

En este nivel hay que atravesar una serie de plataformas que se mueven de forma perpendicular a la dirección que debe seguir la bola para llegar a la meta. En este caso las plataformas no se paran al llegar al fin de su recorrido, sino que inician inmediatamente el camino de regreso. Los recolectables están posicionados estratégicamente para marcar la ruta óptima a seguir. Sin esta guía o una ubicación de los recolectables más exigente puede suponer un incremento considerable de la dificultad del nivel.

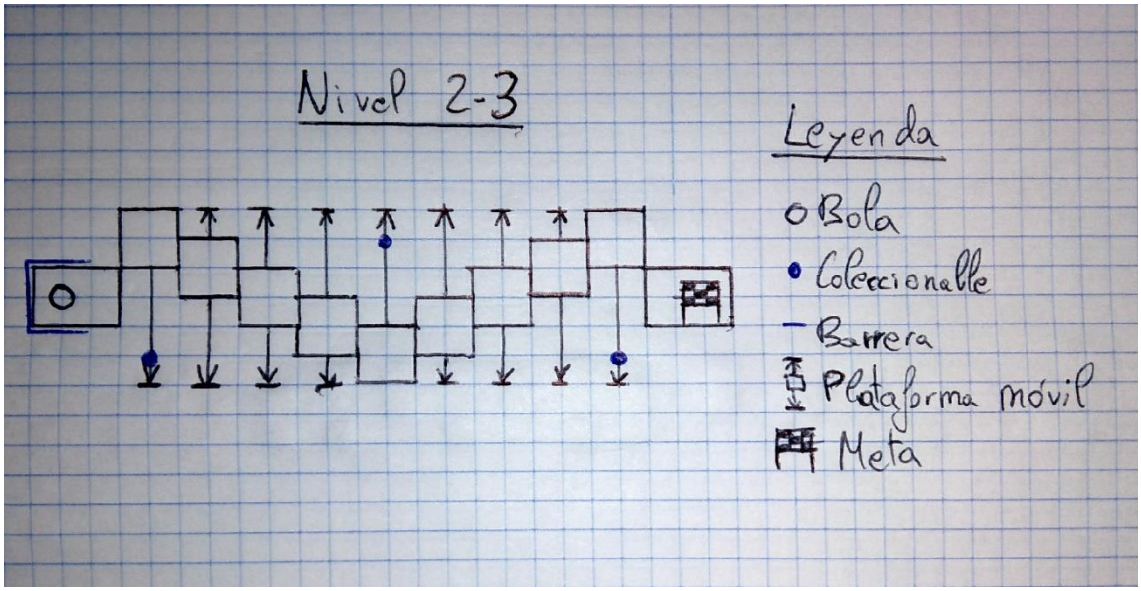


Ilustración 13.11 Boceto nivel 2-3

13.2.12 - Nivel 2-4

En este nivel se sigue explorando la mecánica de la plataforma móvil, pero con una única plataforma que sigue un recorrido más largo y que a lo largo del mismo presenta numerosos obstáculos que el jugador deberá esquivar para llegar a la meta. La ubicación de los recolectables obliga a realizar el recorrido un total de tres veces (ida, vuelta y de nuevo ida) para aquellos jugadores que quieran obtener las tres estrellas en este nivel.

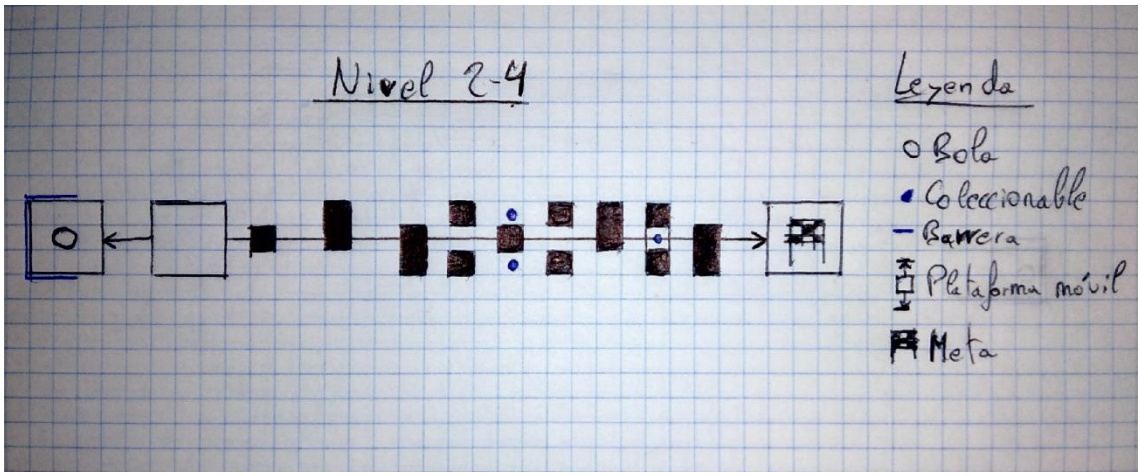


Ilustración 13.12 Boceto nivel 2-4

13.2.13 - Nivel 2-5

En este nivel se introducen dos nuevas mecánicas que se explorarán más en otros niveles. Estas son los elementos giratorios respecto a un punto de referencia y los objetos que rebotan. Hasta ahora, los obstáculos que se han presentado absorbían los impactos de la bola e impedían que esta rebotase. Pero con este nuevo obstáculo la bola sale rebotada hacia atrás unos metros debido al impacto. El jugador deberá superar dos ruletas de objetos rebotadores que giran a alta velocidad y en distintos sentidos para llegar a la meta.

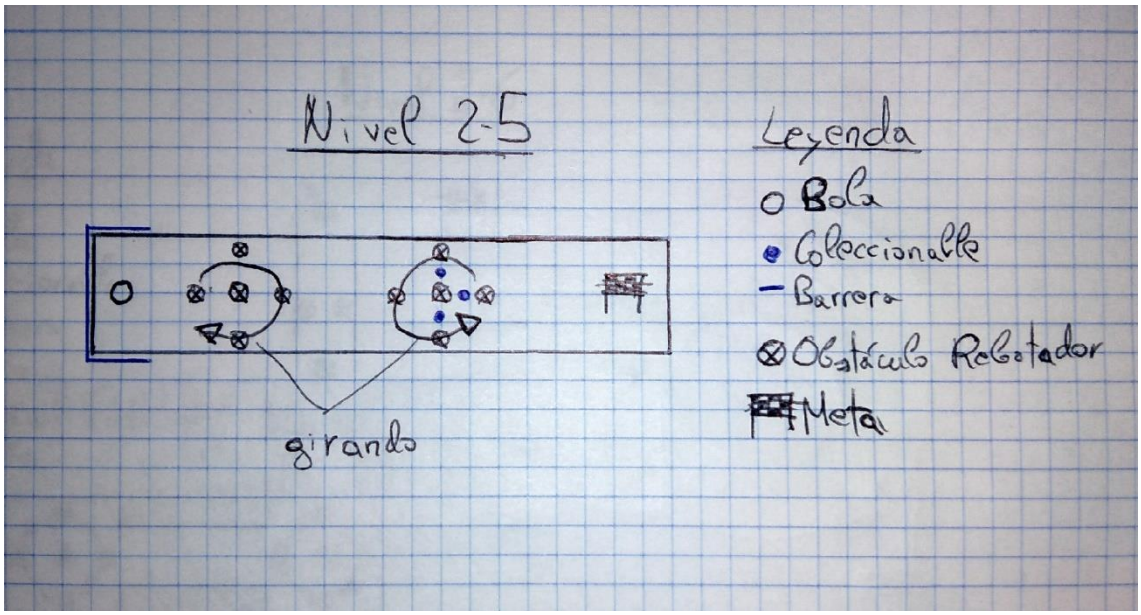


Ilustración 13.13 Boceto nivel 2-5

13.2.14 - Nivel 2-6

En este nivel se continúan explorando las mecánicas introducidas en el nivel anterior. En el inicio del nivel hay dos ruletas como las vistas en el nivel anterior, pero ahora giran más rápido y el jugador debe tomar una curva, por lo que se amplía el tiempo que el jugador debe permanecer al alcance de los objetos rebotadores. Una vez superadas estas dos ruletas se presenta un pasillo con objetos rebotadores estáticos.

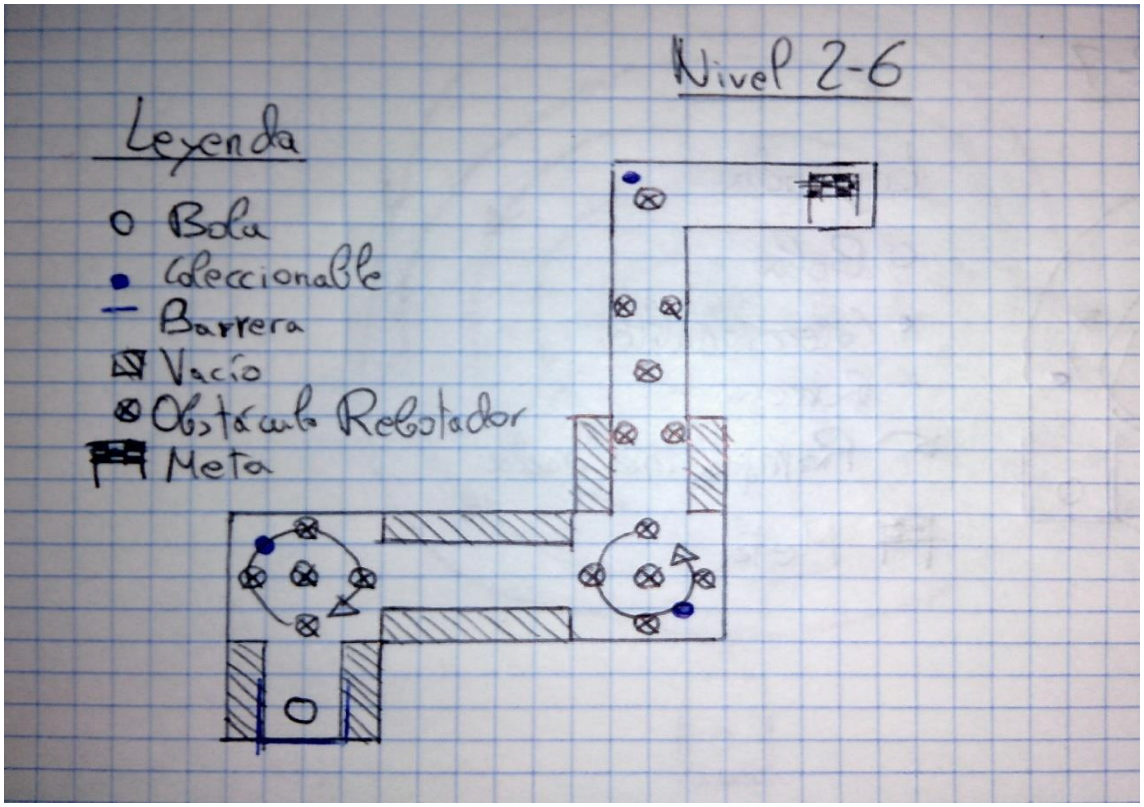


Ilustración 13.14 Boceto nivel 2-6

13.2.15 - Nivel 2-7

Este nivel es algo distinto a lo visto hasta ahora. Se trata de una escalera de caracol que en total da un giro de 540 grados y que el jugador deberá descender para llegar a la meta. Sin embargo, para poder batir el tiempo récord del nivel y ser así galardonado con una estrella, el jugador deberá dejarse caer de la plataforma en el punto adecuado. Ese punto ha sido marcado con un recolectable.

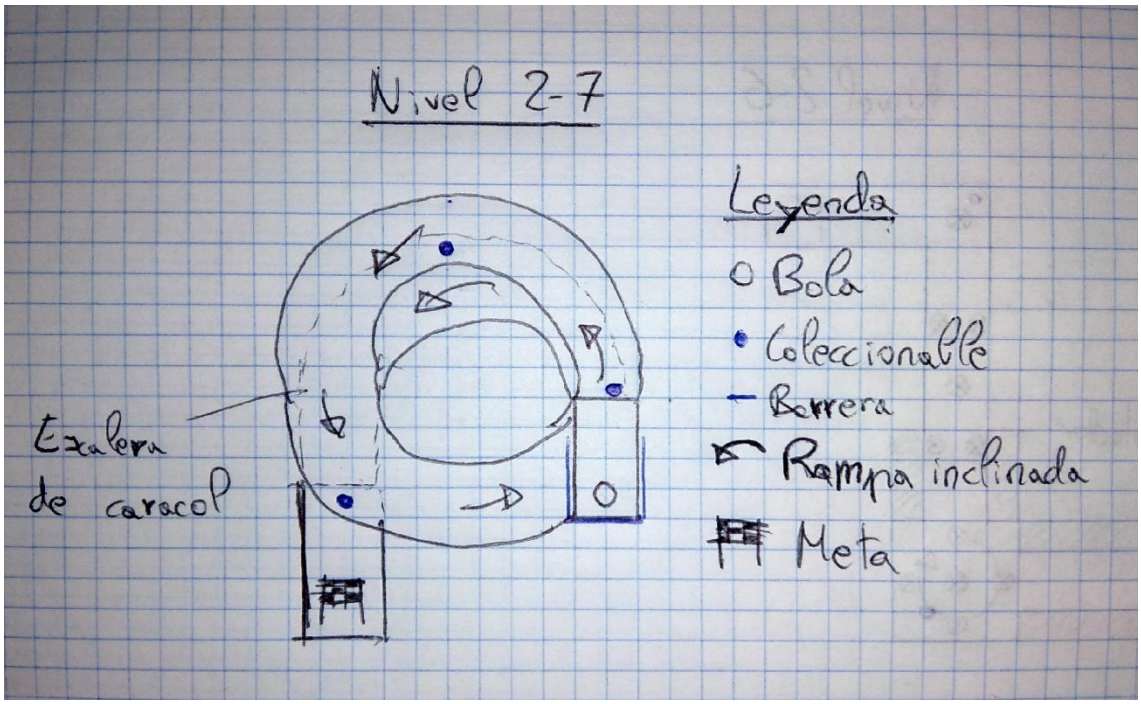


Ilustración 13.15 Boceto nivel 2-7

13.2.16 - Nivel 2-8

Este nivel cuenta con una serie de plataformas que se mueven en círculos concéntricos y el objetivo del jugador es llegar hasta el centro partiendo desde el exterior. A diferencia de lo que pueda parecer, las plataformas de este nivel no son plataformas móviles como las descritas en niveles anteriores, ya que no van de un punto a otro y luego vuelven, sino que giran siempre en el mismo sentido. Siguen, por tanto, el comportamiento de los elementos giratorios respecto a un punto de referencia. El jugador deberá de cruzar un total de cinco plataformas que se mueven a diferentes velocidades y en el sentido opuesto a sus plataformas adyacentes.

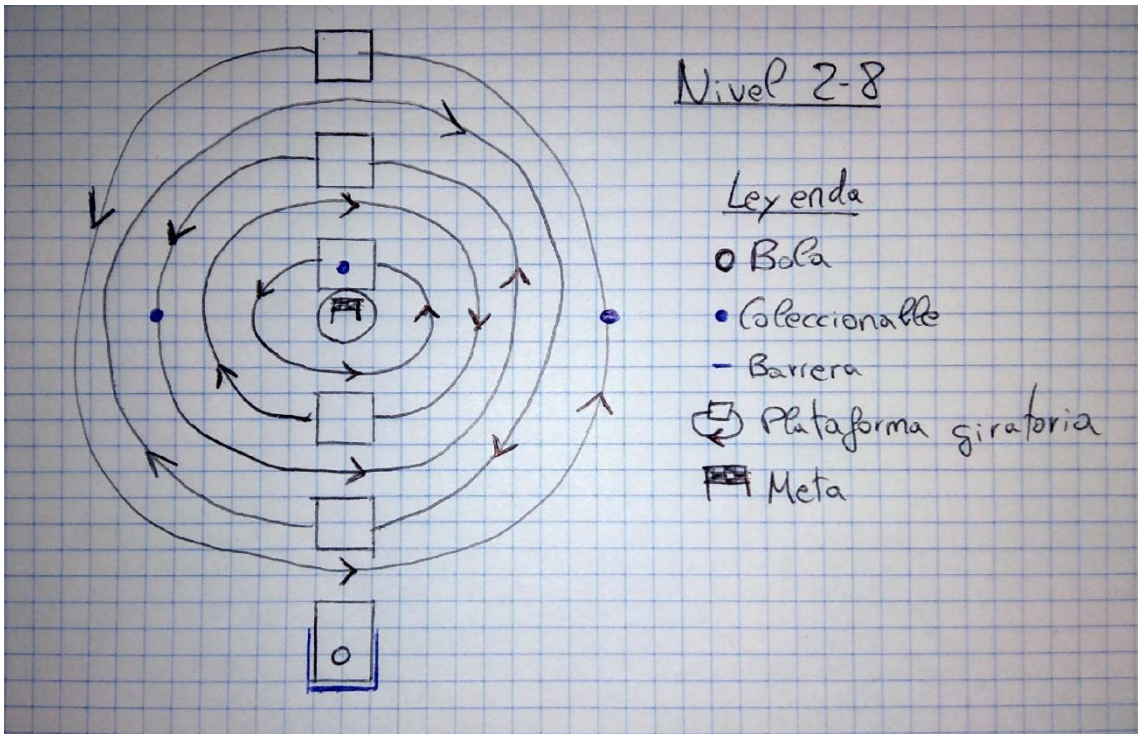


Ilustración 13.16 Boceto nivel 2-8

14 - BIBLIOGRAFÍA

[1] Newzoo, <<Mobile Revenues Account for More Than 50% of the Global Games Market as it Reaches $137.9 Billion in 2018,>> 30 Abril 2018. [En línea]. Available: https://newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/. [Último acceso: 13 Abril 2019].

[2] AppBrain, <<Google Play stats: Top categories,>> [En línea]. Available: https://www.appbrain.com/stats/android-market-app-categories. [Último acceso: 13 Abril 2019].

[3] AppBrain, <<Google Play stats: Free versus paid apps,>> [En línea]. Available: https://www.appbrain.com/stats/free-and-paid-android-applications. [Último acceso: 13 Abril 2019].

[4] Minecraft, <<Minecraft,>> [En línea]. Available: https://www.minecraft.net/es-es/. [Último acceso: 19 Junio 2019].

[5] Google Play, <<Top juegos,>> [En línea]. Available: https://play.google.com/store/apps/collection/cluster?clp=ChwKGgoUdG9wc2VsbGluZ19mcmVlX0dBTUUQBxgD:S:ANO1ljJH_B0&gsr=Ch4KHAoaChR0b3BzZWxsaW5nX2ZyZWVfR0FNRRAHGAM%3D:S:ANO1ljLEXvI. [Último acceso: 18 Junio 2019].

[6] Google Play, <<Juegos top en ingresos,>> [En línea]. Available: https://play.google.com/store/apps/collection/cluster?clp=ChgKFgoQdG9wZ3Jvc3NpbmdfR0FNRRAHGAM%3D:S:ANO1ljKVCGg&gsr=ChoKGAoWChB0b3Bncm9zc2luZ19HQU1FEAcYAw%3D%3D:S:ANO1ljI_yc8. [Último acceso: 18 Junio 2019].

[7] Apple Inc., <<Top Grossing Apps,>> [En línea]. Available: https://www.apple.com/itunes/charts/top-grossing-apps/. [Último acceso: 18 Junio 2019].

[8] PEGI, <<Pan European Game Information,>> [En línea]. Available: https://pegi.info/es. [Último acceso 2 Septiembre 2019].

[9] Sega, <<Super Monkey Ball,>> [En línea]. Available: https://games.sega.com/supermonkeyball/. [Último acceso: 22 Julio 2019].

[10] Wikipedia, <<Marble Madness,>> [En línea]. Available: https://es.wikipedia.org/wiki/Marble_Madness. [Último acceso: 22 Julio 2019].

[11] Unity Technologies, <<Unity,>> [En línea]. Available: https://unity3d.com/es/unity. [Último acceso: 26 Julio 2019].

[12] Epic Games, <<Unreal Engine,>> [En línea]. Available: https://www.unrealengine.com/en-US/. [Útimo acceso: 26 Julio 2019].

[13] Crytek, <<CryEngine,>> [En línea]. Available: https://www.cryengine.com/. [Último acceso: 26 Julio 2019].

[14] Unity Technologies, <<Unity Asset Store,>> [En línea]. Available: https://assetstore.unity.com/. [Último acceso: 2 Septiembre 2019].

[15] Epic Games, <<Epic Games,>> [En línea]. Available: https://www.epicgames.com/store/es-ES/. [Último acceso: 2 Septiembre 2019].

[16] Crytek, <<Crytek,>> [En línea]. Available: https://www.crytek.com/. [Último acceso: 2 Septiembre 2019].

[17] Crytek, <<Crysis,>> [En línea]. Available: https://www.crytek.com/games/crysis. [Último acceso: 26 Julio 2019].

[18] Adobe, <<Adobe Photoshop,>> [En línea]. Available: https://www.adobe.com/es/products/photoshop.html. [Último acceso: 27 Julio 2019].

[19] Microsoft, <<Paint 3D,>> [En línea]. Available: https://www.microsoft.com/es-es/p/paint-3d/9nblggh5fv99. [Último acceso: 27 Julio 2019].

[20] Microsoft, <<Visual Studio,>> [En línea]. Available: https://visualstudio.microsoft.com/es/vs/. [Último acceso: 27 Julio 2019].

[21] Microsoft, <<Microsoft Word,>> [En línea]. Available: https://products.office.com/es-es/word. [Último acceso: 28 Julio 2019].

[22] <<Audacity,>> [En línea]. Available: https://audacity.es/. [Último acceso: 28 Julio 2019].

[23] Unity Technologies, <<Input: Mobile Device Input,>> [En línea]. Available: https://docs.unity3d.com/Manual/MobileInput.html. [Último acceso: 18 Junio 2019].

[24] Ecosistemas Digitales de Negocio, SL, <<Cuentica – Tabla de años y porcentajes de amortización para sociedades,>> [En línea]. Available: https://cuentica.com/asesoria/tabla-anos-y-porcentajes-de-amortizacion-sociedades-a-partir-de-2015/. [Útimo acceso: 28 Julio 2019].

[25] Boletín Oficial del Estado, <<Resolución de 18 de marzo de 2019, de la Dirección General de Trabajo, por la que se registra y publica el XVI Convenio

colectivo estatal de empresas de consultoría y estudios de mercado y de la opinion pública,>> [En línea]. Available: https://www.boe.es/boe/dias/2009/04/04/pdfs/BOE-A-2009-5688.pdf. [Último acceso: 28 Julio 2019].

[26] Seguridad Social, <<Bases y tipos de cotización 2019,>> [En línea]. Available: http://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537. [Último acceso: 28 Julio 2019].

[27] <<FlatIcon,>> [En línea]. Available: https://www.flaticon.com/. [Último acceso: 18 Junio 2019].

[28] Unity Technologies, <<Manual: Physic Material,>> [En línea]. Available: https://docs.unity3d.com/2018.3/Documentation/Manual/class-PhysicMaterial.html. [Último acceso: 3 Septiembre 2019].

[29] Unity Technologies, <<Scripting API: Quaternion,>> [En línea]. Available: https://docs.unity3d.com/ScriptReference/Quaternion.html. [Último acceso: 3 Septiembre 2019].

[30] E. Freeman, E. Freeman, K. Sierra y B. Bates, Head First Design Patterns, 1 ed., O'Reilly Media, Inc, 2004.

[31] MarkLight, <<MarkLight,>> [En línea]. Available: https://koyoki.github.io/marklight/. [Último acceso: 25 Julio 2019].

[32] Unity Technologies, <<Manual: The XUML format,>> [En línea]. Available: https://docs.unity3d.com/Manual/UIE-UXML.html. [Último acceso: 3 Septiembre 2019].

[33] Unity Technologies, <<Manual: Prefabs,>> [En línea]. Available: https://docs.unity3d.com/es/2019.1/Manual/Prefabs.html. [Último acceso: 3 Septiembre 2019].

[34] Wikipedia, <<Back-face culling,>> [En línea]. Available: https://en.wikipedia.org/wiki/Back-face_culling. [Último acceso: 3 Septiembre 2019].

[35] Wikipedia, <<Tiling,>> [En línea]. Available: https://es.wikipedia.org/wiki/Tiling. [Último acceso: 3 Septiembre 2019].

[36] TurboSquid, <<Animated Flag,>> [En línea]. Available: https://www.turbosquid.com/3d-models/flag-x-free/1084430. [Último acceso: 3 Septiembre 2019].

[37] <<icon-icons,>> [En línea]. Available: https://icon-icons.com/es/icono/a-cuadros/54040. [Último acceso: 3 Septiembre 2019].

[38] Unity Technologies, <<Manual: Skybox,>> [En línea]. Availble: https://docs.unity3d.com/es/current/Manual/class-Skybox.html. [Último acceso: 3 Septiembre 2019].

[39] <<HDRI-Hub.com,>> [En línea]. Available: https://www.hdri-hub.com/hdri-skies-aviation-aerospace. [Último acceso: 3 Septiembre 2019].

[40] Youtube, <<Música Hawaiana Sin Copyright (10 minutos),>> [En línea]. Available: https://www.hdri-hub.com/hdri-skies-aviation-aerospace. [Último acceso: 3 Septiembre 2019].

[41] SoundBible, <<Metal Pong From Pan And Pot Sound,>> [En línea]. Available: http://soundbible.com/597-Metal-Pong-From-Pan-And-Pot.html. [Último acceso: 3 Septiembre 2019].

[42] SoundBible, <<Metal Clack Pots Pans Sound,>> [En línea]. Available: http://soundbible.com/596-Metal-Clack-Pots-Pans.html. [Último acceso: 3 Septiembre 2019].

[43] SoundBible, <<Shooting Star Sound,>> [En línea]. Available: http://soundbible.com/1744-Shooting-Star.html. [Último acceso: 3 Septiembre 2019].

[44] Unity Technologies, <<Manual: Coroutines,>> [En línea]. Available: https://docs.unity3d.com/Manual/Coroutines.html. [Último acceso: 3 Septiembre 2019].

[45] Unity Technologies, <<Unity Ads,>> [En línea]. Available: https://unity.com/es/solutions/unity-ads. [Último acceso: 2 Septiembre 2019].

[46] <<How do I earn revenue from Unity Ads?,>> [En línea]. Available: https://support.unity3d.com/hc/en-us/articles/205263819-How-do-I-earn-revenue-from-Unity-Ads-. [Último acceso: 2 Septiembre 2019].