Universitat Oberta de Catalunya

uoc.edu

Gravity Dimension

Carlos Vilarnau Lara
Máster en Diseño y desarrollo de videojuegos
Trabajo Final de Máster

Heliodoro Tejedor Navarro, Jordi Duch Gavaldà
Javier Luis Cánovas Izquierdo

29/06/19

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Gravity Dimension*
Nombre del autor:	*Carlos Vilarnau Lara*
Nombre del consultor/a:	*Heliodoro Tejedor Navarro, Jordi Duch Gavaldà*
Nombre del PRA:	*Javier Luis Cánovas Izquierdo*
Fecha de entrega (mm/aaaa):	06/2019
Titulación::	*Máster en Diseño y desarrollo de videojuegos*
Área del Trabajo Final:	*Trabajo Final de Máster*
Idioma del trabajo:	*Castellano*
Palabras clave	*Unity, Arcade, Plataformas*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Gravity Dimension es un videojuego del género plataformas destinado a dispositivos smartphone con sistemas Android. Este proyecto se ha realizado con el motor gráfico Unity, obteniendo como resultado una experiencia de juego completa, que incluye pantalla de título, selección de nivel, primer nivel completo jugable y pantalla final de créditos.

La jugabilidad de Gravity Dimension aprovecha elementos del smartphone como la pantalla táctil y el giroscopio para controlar al personaje principal, que es una nave espacial. El jugador debe avanzar con la nave realizando saltos entre los diferentes caminos que aparecen a su paso y evitar caer al vacío, además de aprovechar ciertos elementos facilitados para alcanzar el objetivo. El elemento diferenciador de Gravity Dimension, es la alta dificultad del juego, ya que hoy en día existe una gran demanda de videojuegos con una alta dificultad.

Como resultado, obtenemos una experiencia de juego arcade, resumida en un nivel jugable de alta dificultad, pero con un equilibrado inicial, que hace asequible el aprendizaje y la rápida adaptación al control por parte del jugador.

Abstract (in English, 250 words or less):

Gravity Dimension is a videogame of the genre platforms for smartphone devices with Android systems. This project has been done with the Unity graphic engine, obtaining as a result a complete game experience, which includes title screen, level selection, first playable level and final credits screen.

The gameplay of Gravity Dimension takes advantage of elements of the smartphone such as the touch screen and the gyroscope to control the main character, which is a spaceship. The player must go forward with the ship making jumps between the different roads that appear in its path and avoid falling into the void, in addition to taking advantage of certain elements facilitated to reach the objective. The differentiating element of Gravity Dimension, is the high difficult, since nowadays there is a high demand of high difficult videogames.

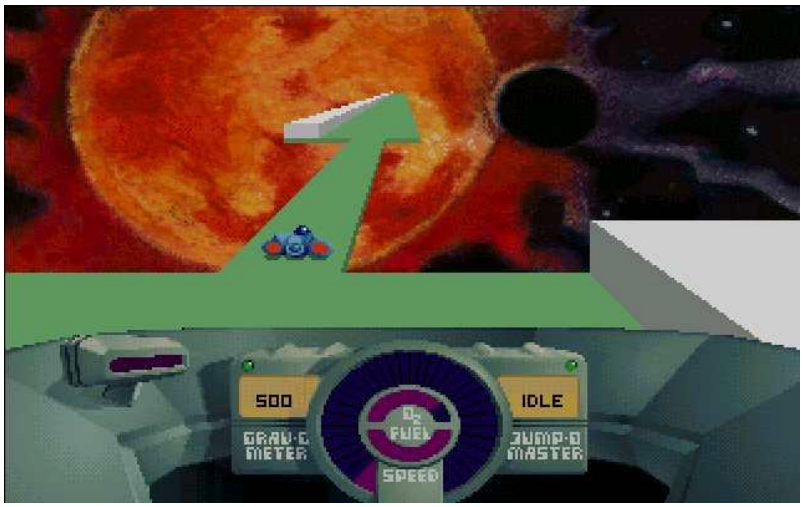
As a result, we get an arcade game experience, summarized in a high difficult playable level, but with an initial balance, which makes learning and the rapid adaptation to control by the player affordable.

Índice

1. Introducción

1.1 Contexto y justificación del Trabajo

La necesidad que se pretende cubrir con este proyecto, es la reinvención de un juego clásico de principios de los 90. Esta reinvención, tiene como objetivo mejorar la experiencia jugable del juego original, además de una sustancial mejora gráfica. Otra de las finalidades es hacer llegar este juego a plataformas actuales como son los smartphones con sistema Android, ya que la plataforma original es MS-DOS, haciendo que sea accesible a día de hoy principalmente vía emulador.



Debido a la creciente tendencia de los llamados speedrunners, (*Disciplina dentro del mundo de los videojuegos dedicada a la finalización de determinados títulos del sector en el menor tiempo posible*)[1] diversos canales de YouTube dedicados a esta disciplina, han empezado a utilizar el juego SkyRoads para publicar sus tiempos de finalización. Esta tendencia se debe a la alta demanda de juegos tipo arcade y/o plataformas con una moderada dificultad que este sector utiliza como reto.

Llegados a esta conclusión, la idea del proyecto ha sido crear un juego nuevo pero utilizando la idea de SkyRoads. La inclusión de ciertas dificultades como limitar la acción de salto y obligar al jugador a obtener energía de salto con varios elementos distribuidos en los diferentes niveles, puede fortalecer la experiencia del juego incluso para un público no tan exigente, que quizás busque algo más que acabar un juego lo antes posible. Por otro lado, ya que no es un juego que requiera una gran potencia gráfica, puede ser ejecutado en la gran mayoría de terminales smartphone actuales.

Como añadido, el control es muy simple, e incluso se puede manejar con una mano. La duración no muy extensa de los niveles lo convierte en un juego arcade con el que poder jugar a cualquier momento sin requerir largas sesiones de juego. Además, hace uso de elementos del smartphone como el giroscopio para controlar el movimiento horizontal de la nave protagonista del juego,

haciendo una experiencia más inmersiva que con los controles básicos en pantalla.

1.2 Objetivos del Trabajo

- Llevar a cabo un proyecto para plataformas móviles haciendo uso de la pantalla multitáctil y el acelerómetro utilizando el motor Unity.
- Crear una experiencia de juego tipo arcade con dificultad suficiente para llegar a los “speedrunners” y retar a los jugadores “casual”.
- Cerrar un proyecto con calidad suficiente para publicar de forma gratuita en la plataforma Play Store de Google.
- Explotar dentro de Unity los aspectos más interesantes y eficientes de cara a la publicación de un juego, teniendo en cuenta el tiempo y los recursos para la ejecución del proyecto.
- Composición y creación de las pistas musicales para el juego, tanto para menús como para los diferentes niveles.

1.3 Enfoque y método seguido

En lo que respecta a ideas y creatividad, las posibles estrategias para llevar a cabo el proyecto, pueden ser la realización del juego a partir de una nueva idea, aportando de este modo algo innovador al mercado, o la adaptación de una idea ya existente, permitiendo reinventar partes de la misma para cubrir necesidades que no se hayan tenido en cuenta en la idea original.

Para la realización de este trabajo, se ha optado por adaptar una idea existente en el mercado, cuyo producto es de 1993. Actualmente existe un colectivo de jugadores que siguen jugando a este mismo juego de 1993, por lo que indica una demanda sobre las características de su jugabilidad.

Se ha creído conveniente la reinvención de una idea de hace décadas frente a la invención de algo nuevo, principalmente por la corta duración del proyecto. Pese a que el proyecto se ha iniciado desde cero, como un producto nuevo, pero teniendo ya claras las ideas sobre la jugabilidad y la temática, llevar una nueva idea a cabo y poder ejecutarla, requiere un tiempo extra para asentar y dar sentido a esa idea, además de conseguir que sea un reto o una experiencia satisfactoria. Además es posible que exista una dificultad añadida a la hora de encontrar recursos o material sobre una nueva idea respecto a otras ideas ya existentes, con lo que puede llegar a obligar a crear algunos recursos desde cero como pueden ser ciertos personajes y elementos particulares exclusivos de esa idea.

1.4 Planificación del Trabajo

Los recursos para la realización de este trabajo, se basan principalmente en el motor gráfico Unity. En él se trabajan todos los aspectos relacionados con el juego mediante un proyecto definido partiendo de escenas. Estas escenas corresponden a los entornos que va a tener el juego, como por ejemplo el menú inicial, o un nivel del juego en concreto.

Uno de los recursos clave para la creación de este proyecto, es una herramienta que se incluye en Unity, pero también es accesible vía web. Hablamos de la ***Unity Asset Store***[2]. Esta herramienta permite tanto la búsqueda e importación de contenido ya sea de forma gratuita o de pago, como la publicación de contenido propio destinado al uso de la comunidad. En el caso de la publicación de contenido, son los desarrolladores los encargados de poner el precio a sus Assets en caso de buscar una remuneración por ese trabajo, ya que muchos otros desarrolladores publican contenido sin ánimo de lucro.

Todos los recursos externos generados, se integran en las escenas correspondientes de Unity para poder trabajarlos e integrarlos en el juego. Estos recursos externos se basan principalmente en imágenes descargadas de la red y posteriormente editadas con el software Adobe Photoshop, además de sonidos de audio y música, que se ha compuesto y editado con el software Ableton Live 9.

Tareas a realizar:

- Obtención de Assets y recursos (Unity Store, web)
- Creación de escenas y menús (Unity)
- Desarrollo de los scripts del juego y Gameobjects (Unity, Adobe Photoshop)
- Creación de niveles (Unity)
- Composición de la música (Ableton Live 9)
- Ejecución de música y efectos en las escenas (Unity, Adobe Audition)
- Testeo y equilibrado de la dificultad (Unity)

Planificación inicial del trabajo:

	Tiempo (Semanas)														
Objetivos	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Obtención de assets y recursos															
Creación escenas y menús															
Desarrollo scripts juego															
Desarrollo elementos y partículas															
Creación diferentes niveles															
Composición de la música															
Aplicación música + fx															
Testeo y equilibrado dificultad de niveles															

Este diagrama representa la planificación inicial, la cual se ha ido modificando levemente a medida que el desarrollo ha ido avanzando. Los motivos de los cambios se especificarán en el **capítulo 7** (Conclusiones).

Planificación final del trabajo:

PEC1
PEC2
PEC3

	Tiempo (Semanas)														
Objetivos	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Obtención de assets y recursos															
Creación escenas y menús															
Desarrollo scripts juego															
Creación nivel															
Desarrollo/Obtención ítems y partículas															
Composición Música															
Aplicación música + Fx															
Testeo y equilibrado nivel															

1.5 Breve sumario de productos obtenidos

El resultado de este trabajo se concentra dentro de un proyecto, donde se unen partes creadas u obtenidas con la finalidad de conseguir un videojuego para móviles en formato **.apk** ejecutable en sistemas Android.

Estas partes son:

- Pistas de audio creadas exclusivamente para el juego.
- Assets de Unity Store utilizados como Gameobjects.
- Tipografía para los textos del juego.
- Fondos de pantalla con temática espacial para cada nivel del juego.

El resultado final:

- Videojuego para plataformas Android llamado Gravity Dimension.
- Proyecto de Unity con Gravity Dimension con el cual poder exportarlo a otras plataformas como videoconsolas o PC.

1.6 Breve descripción de los otros capítulos de la memoria

- **Estado del arte**: Una revisión sobre el género y la tecnología elegidos para el desarrollo del trabajo. En este capítulo también se comparan los diferentes motores gráficos más utilizados en la actualidad, haciendo un repaso sobre las ventajas e inconvenientes de cada herramienta y para qué tipo de juegos o géneros se recomiendan.

- **Definición del juego**: Apartado donde se definen en detalle los aspectos del juego relacionados con la trama, además de la finalidad que tiene el jugador dentro del mismo, la temática y subgéneros.

- **Diseño técnico**: Si bien en el Estado del arte hablamos sobre las distintas tecnologías de desarrollo más utilizadas a día de hoy, en Diseño técnico empezaremos explicando qué herramienta hemos elegido para llevar a cabo el trabajo y porqué. También profundizaremos en los requerimientos técnicos para la ejecución del motor gráfico utilizado y otras herramientas necesarias como software de edición de imágenes y sonido. Los Assets de Unity también juegan un papel fundamental para dar vida a los objetos utilizados en el proyecto, por lo que haremos repaso por los diferentes paquetes utilizados.

- **Diseño de niveles**: Una explicación detallada del diseño del nivel, los elementos que contiene y los puntos más destacados. Se analizará mediante un mapa completo del nivel.

- **Manual de usuario**: Manual que repasa todas las acciones a realizar por el jugador para llevar a cabo los objetivos planteados por el juego.

- **Conclusiones**: Repaso personal sobre todo lo aprendido durante el trabajo y las mejoras que se han podido realizar. En cierta forma se sintetiza lo analizado anteriormente.

- **Pruebas con usuarios**: Se expone el feedback realizado con las pruebas de diferentes tipos de usuarios, ya sean más afines a los videojuegos o menos.

2. Estado del arte

2.1 Una revisión sobre el género de Plataformas.

El género conocido como plataformas es un género donde la jugabilidad tiene un gran papel. Esta jugabilidad consta de acciones como moverse y saltar entre obstáculos y objetos similares a plataformas. El género en sí no puede ser considerado exclusivamente como un género independiente, ya que a menudo se combina con otros géneros de juegos como Shooters y juegos de Rol.

El género de plataformas se originó en la escena arcade a principios de los 80, Donkey Kong (Nintendo, 1981).

Durante la siguiente década el género de plataformas produce algunos de los juegos más influyentes hasta la fecha con títulos como Super Mario Bros. (Nintendo, 1985), que sostuvo el título de ser el juego más vendido de todos los tiempos durante tres décadas, Sonic the Hedgehog (Sega, 1991) y Super Mario Bros. 3 (Nintendo, 1993).

Aunque el género siguió el desarrollo 3D en el siglo XXI con juegos como Super Mario Sunshine (Nintendo, 2002) y Rayman 3 (Ubisoft, 2003), posteriormente, comenzó a mostrar una popularidad decreciente en el mercado.

Daniel Boutros[3] afirmó debido a que el género disfrutó del 2% del mercado de videojuegos en 2002 en comparación con el 15% en 1998:

“Creemos que no es un tema de género, sino un problema de pérdida de principios efectivos en del diseño”.

Esta declaración, que se hizo en 2006, fué bastante precisa, ya que actualmente estamos presenciando un renovado interés en el género de plataformas.

Juegos como New Super Mario Bros 1 y 2 (Nintendo 2006, 2012), Donkey Kong Country Tropical Freeze (Nintendo 2014) y Rayman Origins (Ubisoft, 2011), que han sido éxitos recientes, básicamente, son remakes de algunos de los juegos de plataformas 2D más vendidos.

Esto fortalece el razonamiento de Boutros sobre los componentes importantes relacionados al diseño. Ya que estos éxitos de ventas, rescatan en cierta manera, la esencia de los originales en jugabilidad y diseño.

Daniel Boutros[3], afirmaba que los desarrolladores actuales buscan ampliar el mercado, haciendo más accesibles los juegos de plataformas, haciendo más fácil el juego. Por su contra, esto causa un efecto de falta de similitud con la esencia de los éxitos del género.

Como resultado, tenemos que la dificultad en el diseño de los juegos de género plataformas tiene un papel esencial en el éxito de los mismos.

Casos como Flappy Bird (Dong Nguyen, 2014), que fué un videojuego para dispositivos móviles donde su éxito residía en su extremada dificultad, resultó siendo un éxito inimaginable entre la comunidad de jugadores.

Por otro lado, el juego en el que este trabajo se ha basado, SkyRoads (Bluemoon, 1993), también tuvo un éxito arraigado a su dificultad. Por este mismo motivo, a día de hoy, algunos Speedrunners siguen haciendo uso de SkyRoads y su dificultad para compartir sus habilidades con la comunidad.

2.2 Una revisión sobre la tecnología.

Existen multitud de entornos de desarrollo integrados (IDE) dedicados a la creación de videojuegos, comúnmente llamados motores gráficos. La gran mayoría se distingue por su complejidad y posibilidades respecto a los resultados que se pueden obtener, ya sea potencia gráfica, herramientas de gestión de elementos y rendimiento. Otro de los puntos clave respecto a los motores gráficos es para qué plataformas se puede desarrollar.

A continuación analizaremos las opciones más utilizadas a día de hoy:

Unity

Unity es un entorno muy popular de desarrollo para juegos 3D, 2D, video y otros contenidos interactivos. Los juegos desarrollados en Unity se pueden ejecutar en gran variedad de consolas y sistemas operativos tales como Windows, Mac, Xbox One, PlayStation 4, Nintendo Switch, iOS, Android, Chrome, Flash y Linux.

Consta de dos elementos principales: un editor para el desarrollo/diseño de contenidos y un motor de juego. Ambos están estrechamente integrados, lo cual permite que desde el mismo editor se puedan realizar acciones que invocan al motor de juego. Un ejemplo de esto es que desde el mismo editor puedes ejecutar y probar en vivo el proyecto que se está desarrollando. El código para programar con Unity puede ser mediante scripts con el lenguaje C# o con Javascript.

En general, podemos decir que Unity es un entorno de desarrollo de juegos que destaca por soportar múltiples plataformas para ejecutar los juegos desarrollados, así como por la facilidad de uso y productividad de su editor.

Unity está disponible en distintas versiones y precios. Aunque existe una versión gratuita llamada **Personal**, esta no contiene toda la funcionalidad de la edición **Plus** (25$ al mes con prepago de un año) ni la **Pro** (125$ al mes). Con cualquier versión se pueden publicar juegos en las diferentes plataformas de desarrollo, siempre y cuando, para la versión personal no superen los 100.000$ de beneficio por año. En caso de superar esa cuantía, estaremos obligados a adquirir la versión Plus. La versión Plus añade más recursos técnicos como almacenaje en la nube de Unity y diversas herramientas como asesoramiento directo, cursos de aprendizaje, además de herramientas de feedback sobre los usuarios del juego desarrollado. La versión Pro, sin embargo, no tiene límite de beneficios e incluye todas las ventajas de la versión Plus, además de funciones avanzadas para gestión de equipos de desarrollo, estadísticas de uso y acceso gratuito a contenido artístico producido por profesionales.

Unreal Engine

El Unreal Engine, actualmente en su versión 4, es un motor creado por Epic Games, y que ha sido utilizado para desarrollar varios de los videojuegos para consolas más populares en el mercado, como la serie Gears of War, Assassin's Creed, Bioshock o Infinity Blade para iOS. Durante los primeros años, Unreal Engine solo estuvo disponible para estudios de videojuegos profesionales, sin embargo en noviembre del 2009 Epic liberó el Unreal Development Kit (UDK), que es una versión del Unreal Engine disponible al público en general con el que se puede crear juegos para distintas plataformas incluyendo las consolas Playstation 4, Xbox One, Nintendo Switch y otras plataformas como PC, Mac, Android, iOS, Oculus y Steam VR.

Este motor destaca por sus capacidades avanzadas para el renderizado de gráficos, incluyendo técnicas como iluminación por píxel, sombras dinámicas y high dynamic range rendering (HDRR). También cuenta con su propio entorno de edición (Unreal Editor). Los scripts se crean en un lenguaje de programación propietario (UnrealScript) y se conectan entre sí usando el editor visual Unreal Blueprints.

Unreal Engine es un software gratuito a día de hoy, adoptando un modelo de negocio similar al de Unity, es decir, si el proyecto desarrollado no supera una cantidad de ingresos de 3000$. En el caso de superar esa cantidad como beneficio, deberemos abonar a Epic un 5% de las ganancias brutas obtenidas.

Cocos 2D

Cocos2d es un framework open source escrito en Python para crear juegos 2D. Además de a versión original de Python, existen diversos ports a otros lenguajes y plataformas como iOS, Android, HTML5 y XNA.

Entre las ventajas de cocos2d destacan las siguientes:

- Fácil de usar. Su API es sencilla e incluye una gran variedad de ejemplos. Provee abstracciones de alto nivel para las tareas más comunes.
- Rápido. Cocos2d utiliza las mejores prácticas de OpenGL ES y estructuras de datos optimizadas.

- Es software libre. Cocos2d está bajo licencia MIT, una licencia muy flexible que permite utilizarlo tanto para hacer juegos de código abierto como cerrado. Además puedes extenderlo e integrarlo con bibliotecas de terceros.
- Comunidad activa. La comunidad de cocos2d es grande y activa, en los foros se puede obtener respuestas rápidamente.

Cocos2d-Swift para iOS utiliza las herramientas y lenguajes de esta plataforma (Objective-C, XCode), lo cual puede ser una ventaja si ya estás familiarizado con ellos o una desventaja si no te son familiares. Otra desventaja de Cocos2d es que a diferencia de las herramientas comerciales como Unity o Unreal, no posee un editor gráfico para animaciones o escenas. Además, tampoco posee la versatilidad de exportación multiplataforma que poseen otras plataformas como Unity o Unreal.

CryEngine

Motor desarrollado por la empresa CryTek y posteriormente adquirido por Ubisoft. Originalmente fue destinado a ser un motor de demostración para las tarjetas gráficas de Nvidia, pero debido a su gran potencial, se utilizó en el desarrollo del primer FarCry.

Es uno de los motores más potentes a nivel gráfico, sin embargo, no es un motor de fácil uso u orientado al aprendizaje o perfiles menos experimentados en motores gráficos. Utiliza lenguaje de programación C++ y puntualmente se puede utilizar Java.

Se podría denominar a este motor, como de nicho, ya que en su versión más reciente, la 5, se suele utilizar para desarrollos muy específicos de alta complejidad.

Además de los motores anteriormente comentados, debemos nombrar el de la prestigiosa Valve, el motor **Source**.

Source, fue un motor revolucionario en el desarrollo de los primeros Half Life y Counter Strike. Actualmente se encuentra en una fase muy desactualizada. Pero la empresa Valve ha anunciado para este 2019 el lanzamiento de **Source 2** que promete estar a la altura de los motores más conocidos, tanto en potencia como en modelo de negocio.

3. Definición del juego

Gravity Dimension es un videojuego de plataformas 3D con vista trasera y desplazamiento automático, donde el personaje principal, que es una nave espacial, deberá evitar caer al vacío saltando entre plataformas distribuidas en tres caminos paralelos. El objetivo que se plantea al jugador es conseguir llegar hasta el final sin morir ya sea cayendo al vacío o colisionando contra algún túnel. Para ello, el jugador dispondrá de un movimiento horizontal haciendo uso del giroscopio del smartphone y un movimiento de salto tocando la pantalla multitáctil.

La dificultad del juego, radica en la necesidad de realizar el menor número de saltos entre plataformas para evitar caer al vacío, ya que la nave protagonista posee una barra indicadora de *Fuel* que se irá agotando con cada salto realizado. Para evitar vaciar la barra de combustible, se han diseñado dos opciones:

- Utilizar los caminos que tengan las plataformas más largas o los que tengan rampas para aprovechar el salto de la rampa. De este modo podremos desplazarnos el máximo tiempo posible por el nivel consumiendo el mínimo combustible.
- Recoger los ítems de *Fuel*. Se han diseñado barriles repartidos por los diferentes niveles para que la nave protagonista pueda usarlos, de manera que si pasa sobre alguno de ellos, la barra indicadora de fuel se volverá a llenar, permitiendo poder realizar más saltos.

Otro aspecto importante sobre la dificultad es el aumento de la velocidad a medida que se avanzan niveles. Cada nivel tiene unos valores específicos de desplazamiento de plataformas, por lo que a medida que el jugador consigue avanzar un nivel, este notará un sensible aumento en la velocidad. De este modo se acentúa la curva de dificultad en cada nivel.

La trama transcurre en una dimensión espacial donde la nave protagonista ha llegado por error. Esta dimensión posee una fuerza gravitacional excepcional de la que el protagonista deberá escapar. Cada nivel se identifica por un fondo de pantalla diferente haciendo referencia a distintos lugares de la dimensión. Por lo que la temática general del videojuego y cada uno de sus niveles se sitúan en el espacio exterior.

Los subgéneros que abarca Gravity Dimension son Arcade (el término *arcade* se refiere a los videojuegos clásicos o que recuerdan a las máquinas del mismo nombre. También se usa para diferenciar a los simuladores. *Arcade*, en este sentido, suele referirse a los juegos relativamente fáciles de jugar o que no responden fielmente a la gravedad y otras fuerzas.)[4], Endless runner (Género donde el jugador debe avanzar de manera irremediable en una misma dirección, generalmente escapando de algún enemigo o peligro, y cuyo objetivo es avanzar lo máximo posible antes de morir.)[5] y Plataformas (género de videojuegos que se caracterizan por tener que caminar, correr, saltar o escalar sobre una serie de plataformas y acantilados, con enemigos, mientras se recogen objetos para poder completar el juego.)[6].

4. Diseño técnico

4.1 Entorno de desarrollo

Para poder llevar a cabo este trabajo, se ha optado por el motor Unity. El motivo principal es el conocimiento adquirido durante estos dos años de Máster. Por otro lado, Unity es un motor que tiene ciertas ventajas respecto a sus competidores, sobre todo a la hora de desarrollar juegos para smartphones como es el caso. Unity a diferencia de Unreal, su competidor directo, posee un editor más sencillo, por lo que es un motor más indicado para programadores que se están iniciando en el desarrollo de videojuegos. Además, Unity posee un bajo consumo de recursos del PC en el que se está ejecutando respecto a Unreal, que tiene un consumo de recursos mucho mayor, por lo que no requerirá tener un PC extremadamente potente, con los gastos económicos que eso conlleva.

Los juegos que normalmente se desarrollan para ambos también tienen una diferencia respecto a la potencia, los recursos necesarios para ejecutar los juegos resultantes suelen ser mayores para los juegos de Unreal, ya que está orientado para desarrollos más potentes que Unity. Por ese mismo motivo, Unreal se utiliza para desarrollar costosos juegos destinados a plataformas como Playstation 4, Xbox One X o PC.

Por lo que a modelo de negocio se refiere, Unity ofrece la versión *Personal* de forma gratuita. Esta versión es lo suficientemente completa para poder realizar proyectos individuales de bajo coste que no superen los 100.000$ de beneficio. En tal caso, deberemos adquirir una licencia Plus.

Llegados a este punto, Unity se posiciona claramente como la herramienta indicada para el desarrollo de Gravity Dimension. Juego de bajos recursos y orientado a plataformas móviles con sistema Android.

4.2 Requerimientos técnicos del entorno de desarrollo.

Requerimientos técnicos para el entorno de desarrollo Unity

OS: Windows 7 SP1+, 8, 10, solamente versiones 64-bit; macOS 10.12+.

No se han probado las versiones de servidor de Windows & OS X.

CPU: Soporte para el conjunto de instrucciones SSE2.

GPU: Tarjeta de video con capacidad para DX10 (modelo de shader 4.0).

El resto depende principalmente de la complejidad de sus proyectos.

Requisitos adicionales para el desarrollo de plataformas:

- iOS: Ordenador Mac ejecutando como mínimo macOS 10.12.6 y Xcode 9.4 o superior.
- Android: Kit de desarrollo Android SDK y Java (JDK); el IL2CPP scripting backend requiere Android NDK.
- Plataforma Windows universal: Windows 10 (64 bits), Visual Studio 2015 con componente C++ Tools o posterior y SDK para Windows 10.

Requerimientos técnicos para ejecutar juegos de Unity

Por lo general, el contenido desarrollado con Unity puede ejecutarse bastante bien en todas partes. Qué tan bien se ejecuta depende de la complejidad de su proyecto.

Requisitos más detallados:

- Escritorio:
  - OS: Windows 7 SP1+, macOS 10.12+, Ubuntu 16.04+.
  - Tarjeta de video con capacidad para DX10 (modelo de shader 4.0).
  - CPU: compatible con el conjunto de instrucciones SSE2.
- El reproductor de iOS requiere iOS 9.0 o superior.
- Android: OS 4.1 o posterior; ARMv7 CPU con soporte NEON o CPU Atom; OpenGL ES 2.0 o posterior.
- WebGL: Cualquier versión de escritorio reciente de Firefox, Chrome, Edge o Safari.
- Plataforma Windows universal: Windows 10 y una tarjeta gráfica con capacidades DX10 (modelo de shader 4.0).

4.3 Inventario de herramientas empleadas.

Listado de las herramientas necesarias para la realización de este proyecto:

- **Unity.** Motor gráfico y nexo principal para el desarrollo del videojuego. Es el encargado de poner en funcionamiento todos los elementos programados en cada escena del juego. En él se incluyen partes del desarrollo como los scripts en C# encargados del control de cada elemento programado, las escenas del juego y todos los elementos externos que se han introducido en cada una de ellas.

- **Unity Asset Store**. Herramienta de recursos 3D integrada en Unity para la obtención de objetos creados por terceras personas como la nave protagonista, los túneles, las rampas de cada nivel y los efectos de partículas. Todos estos elementos se importan automáticamente en el proyecto de Unity y quedan a disposición del programador para su posterior uso.

- **Visual Studio**. Entorno de desarrollo compatible con C# incluido en la instalación de Unity para Windows. Es la herramienta con la que crear y editar todos los scripts que se incluyen en un proyecto de Unity.

- **Adobe Photoshop**. Software de edición de imágenes utilizado para procesos de recorte y suavizado de bordes de algunos elementos en formato imágen incluidos en el proyecto.

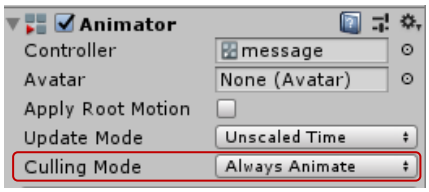
- **Ableton Live 9**. Secuenciador de audio o DAW (Digital Audio Workstation) diseñado para composición musical. Este software, incluyendo algunas herramientas de sintetización de terceros, ha sido el utilizado para la creación y edición de todas las pistas de audio del juego. Estas pistas incluyen la música principal del menú, la música para cada nivel y los efectos de audio del juego.

4.4 Inventario y descripción de assets y recursos

Animaciones:

- Blink (parpadeo): Animación generada con la herramienta **Animation** de Unity. La finalidad de esta animación, es dotar de un efecto parpadeante a los textos que aparecen a modo de tutorial en la escena del nivel jugable.

  Para la creación de este efecto, se ha utilizado la automatización de la transparencia de color del texto. Gracias a la herramienta Animation, se ha creado una automatización del parámetro “Color.a” de Text.Color. Una vez automatizado el efecto, aplicando transparencia y volviéndola a aplicar, se utiliza la animación en modo bucle para que permanentemente aparezca el efecto de parpadeo.



DigitalKonstrukt:

- Pack de assets obtenido de Asset Store, con elementos de construcción para ciudades. Este pack se ha aprovechado para la utilización de los túneles del juego.

Fonts:

- Conthrax-sb es un estilo de fuente descargado de la web de recursos **DaFont**, utilizado para dar el estilo de los textos a todo el juego.

Free_Barrel:

- Asset descargado de Asset Store que contiene el objeto del Barril de fuel y el charco representativo con dicho material.

Images:

- Fondos de pantalla: Imágenes utilizadas como fondos del espacio para cada escena del juego. Estas imágenes se han descargado de diversas páginas web de fondos de pantalla en alta resolución de forma gratuita.

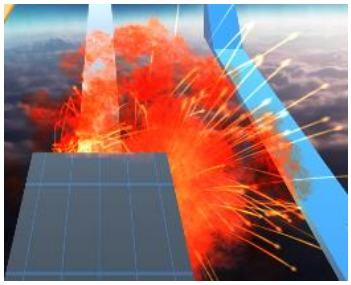
- Elementos en formato png utilizados como Sprites 2D para dar forma a objetos como el botón de pausa, la barra de fuel y las indicaciones de algunos mensajes del tutorial.

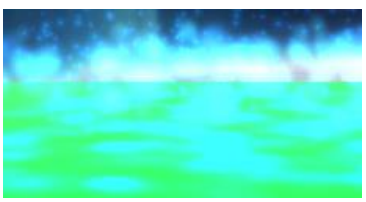
Materials:

- Materiales creados específicamente para aplicar color a las texturas de las plataformas del juego. Se aplica tanto a las plataformas donde avanza la nave, como a las rampas de salto.

Particle Ingredient Pack:

- Paquete de partículas descargadas de Asset Store. En concreto de este paquete, se utilizan tres efectos de partículas. La explosión de la nave cuando colisiona contra un túnel, el efecto propulsor de la nave y el efecto brillante de la plataforma que representa la meta del nivel.





Scenes:

- **MainScreen**: Escena inicial del juego. Es la primera escena que se carga en Gravity Dimension. Esta escena es a modo de “título” donde el usuario decide si jugar al juego o salir de él.

- **LevelSelection**: En caso de que el jugador seleccione jugar al juego en la escena MainScreen, aparecerá en esta escena. Esta escena es la encargada de listar los niveles del juego disponibles. En este caso, solamente está disponible el nivel 1, pero la intención es ir ampliando con diferentes niveles de manera que no se puedan seleccionar a no ser que el jugador haya superado el nivel anterior.
- **Level1Tuto**: Escena que contiene el primer nivel del juego, pero además, contiene las indicaciones a modo de tutorial para la primera vez que se ejecuta el juego. Cuando el usuario selecciona el primer nivel, se lanza esta escena solamente una vez. El resto de veces, ya sea por reinicio debido a una muerte o por reinicio del modo pausa, el juego saltará a la escena Level1.

- **Level1**: Es la misma escena Level1Tuto, pero sin el contenido de tutorial, de este modo, evitamos indicar cada vez que el jugador muere las indicaciones de cómo jugar.

- **Credits**: Escena final del juego. Una vez el jugador, haya finalizado el nivel, tras unos segundos, saltará esta escena que contiene un mensaje de agradecimiento y los créditos del autor del videojuego, además de un tipo de música diferente para el final.

Scripts:

- **ButtonManager**: Script que contiene las funciones necesarias para los botones que el jugador puede usar dentro de las escenas **MainScreen** y **LevelSelection**. Estas funciones equivalen a saltar a la siguiente escena, salir del juego o seleccionar el nivel correspondiente.

- **cameraFollow**: Este script se ha creado específicamente para que la cámara de las escenas del nivel jugable, mantenga una distancia óptima detrás de la nave. Debido a la fuerza de fricción en las rampas de salto, la distancia de la nave respecto a la cámara se iba acortando. De este modo, forzamos la distancia de la cámara para que ésta sea durante toda la partida la misma.

- **cameraZoomMenu**: Script encargado de hacer un efecto de Zoom puramente estético respecto al fondo de pantalla en la escena inicial **MainScreen**.

- **dontDestroyOnLoad**: Código que se encarga de mantener sin interrupción la música que suena durante el nivel jugable. Es decir, que cuando el protagonista muera, o el jugador decida reiniciar la partida, seguirá sonando la música sin cortes ni reinicios. De este modo evitaremos una sensación repetitiva sobre la música, ya que este juego posee una elevada dificultad y se requieren muchos intentos para superarlo.

- **EndScript**: Pequeña función que se invoca tras los 6 primeros segundos de mostrar la pantalla de créditos una vez finalizado el nivel, para posteriormente volver a la pantalla inicial del juego.

- **itemsManager**: Script encargado de la gestión de los barriles de fuel. Este código se encarga de la rotación y del movimiento hacia arriba y abajo del barril, además de volver a cargar al 100% la barra de fuel una vez cogido por la nave.

- **itemsManagerNoTuto**: Este script es una copia del anterior, pero levemente modificado y adaptado a la escena del nivel sin el tutorial para el jugador. Estas modificaciones se deben a que cada una de las escenas (con tutorial o sin) tienen algunos scripts con nombre diferentes, por lo que en determinados casos, algunas referencias han tenido que ser modificadas.

- **musicMenu**: Pequeño script encargado de que la música de la escena inicial **MainScene**, se siga reproduciendo en la escena siguiente de selección de nivel sin corte alguno.

- **PlatformMovement**: Script esencial como “motor” del juego, ya que es el encargado de mover las plataformas por las que debe avanzar la nave. El planteamiento de este juego se basa en que la nave no avanza, sino que lo que realmente se mueve son las plataformas, creando el efecto de que es la nave la que avanza. Este planteamiento nos permite de manera más sencilla trabajar el resto de elementos del juego, como el fondo y los elementos de la interfaz, de modo que todo está estático excepto las plataformas y no hay que aplicar movimiento respecto a la nave o la cámara.

- **ShipMovement**: Este es el script más importante del juego, junto con **PlatformMovement**, ya que es el encargado de

aplicar el movimiento horizontal, salto, explosión, muerte de la nave e interacción con el menú de pausa, junto con el control del giroscopio del smartphone, además de controlar la barra de fuel y los mensajes a modo de tutorial.

- **ShipMovementNoTuto**: Versión del script **ShipMovement**, pero adaptado a la escena Level1 (que no posee tutorial). En este caso, la diferencia radica en que se han eliminado los mensajes de tutorial.

Sounds:

- **menutrack**: Canción creada para los menús iniciales de MainScreen y LevelSelection.

- **lvl1track**: Esta canción, también creada de cero, se utiliza en las escenas del nivel jugable.

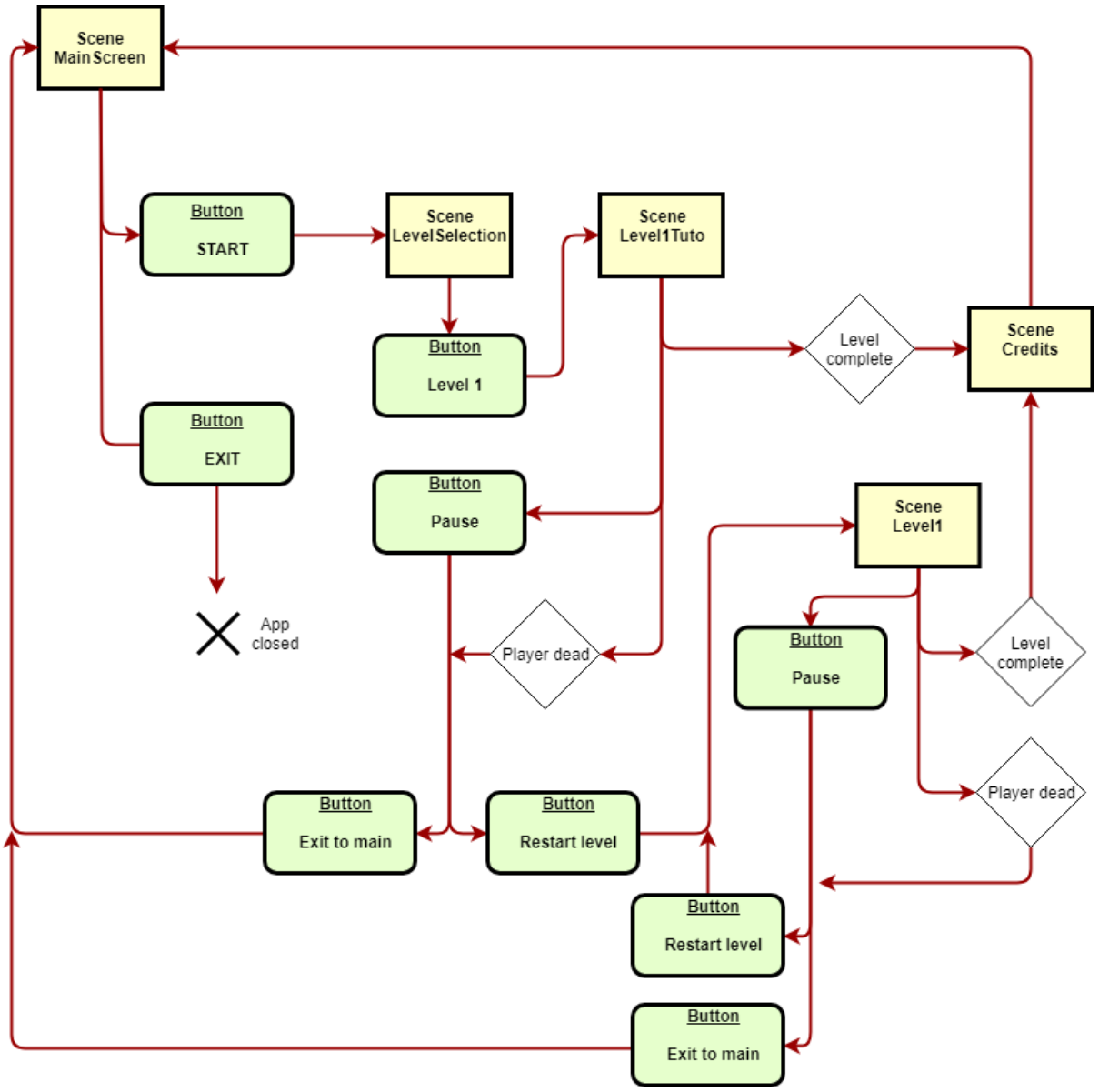
- **finishtrack**: Pequeño fragmento de una canción ya existente, esta vez utilizada para la escena de los créditos de seis segundos de duración.

Las canciones se han creado mediante el software de creación musical **Ableton Live 9.**

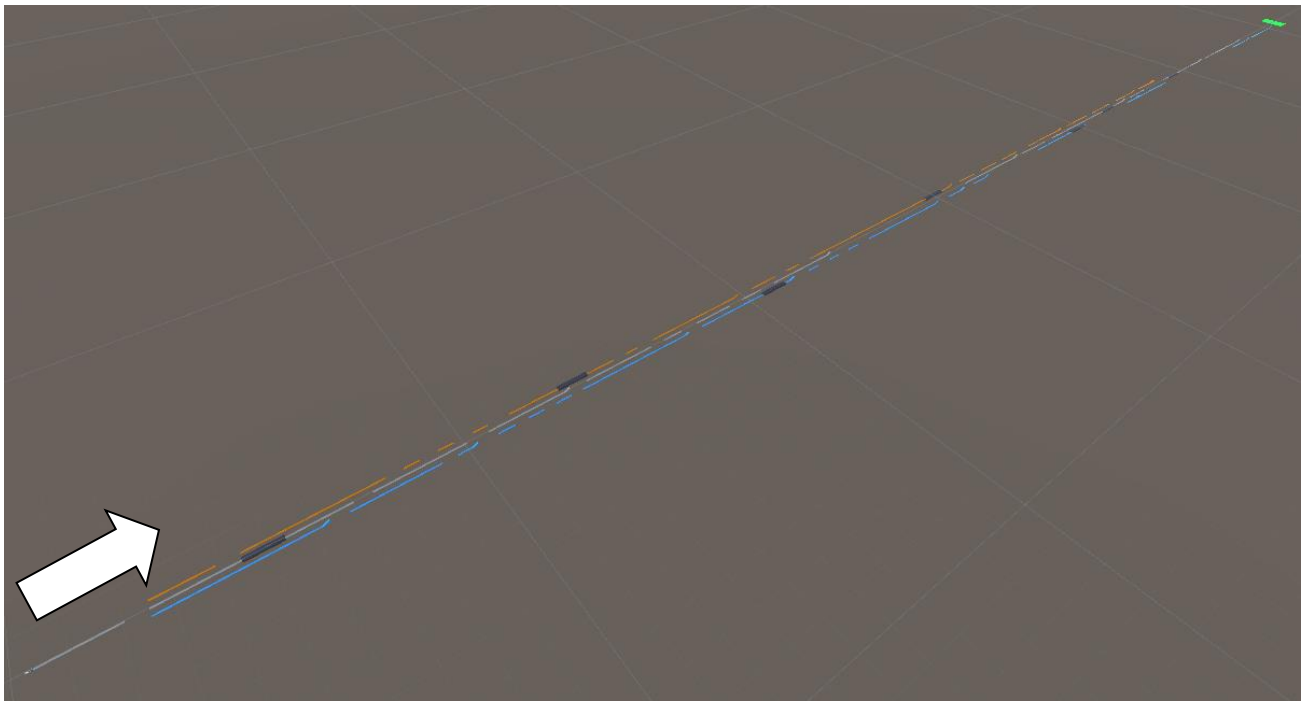
StarSparrow:

- Conjunto de naves espaciales descargado de Asset Store de Unity. Este pack contiene diferentes modelos de naves espaciales, entre ellos, la nave protagonista de Gravity Dimension.

Esquema de arquitectura de Gravity Dimension:



5. Diseño del nivel

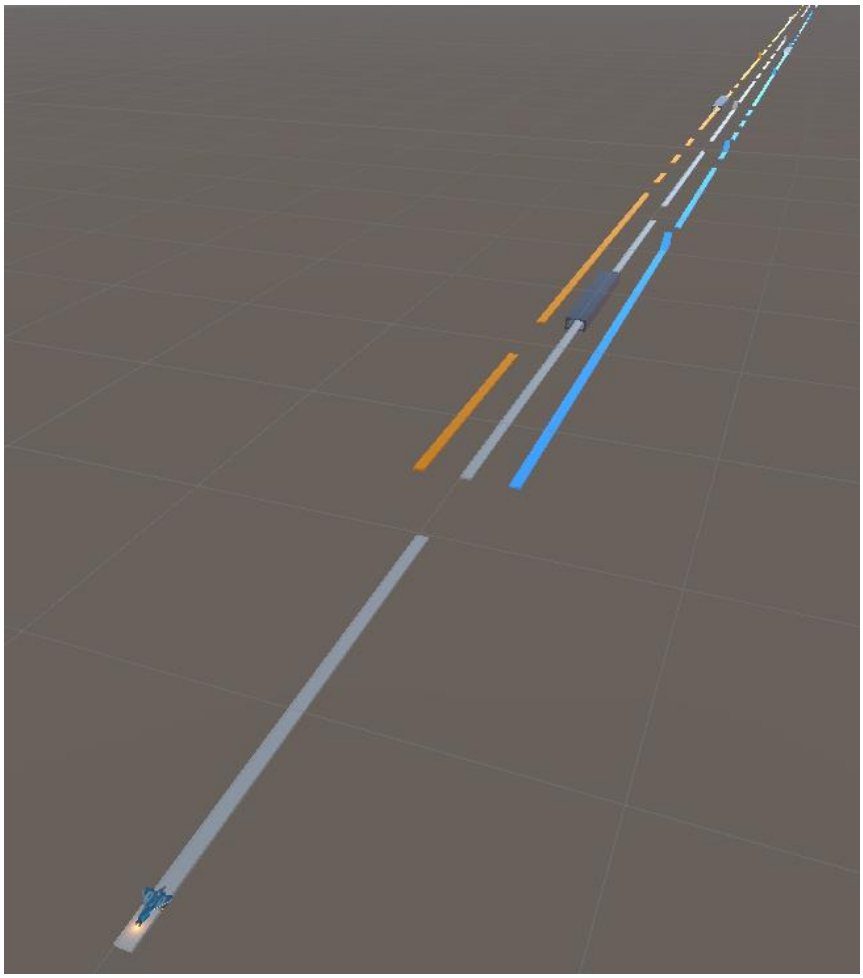


Cuando hablamos del diseño de Gravity Dimension, hablamos del diseño de un juego de plataformas asociado a una cierta dificultad. Para el diseño del nivel, el elemento fundamental y el que más abunda durante la partida, son las plataformas.

Estas plataformas se basan en tres diferentes caminos, los cuales se pueden acceder de dos formas, saltando con la nave, o utilizando las rampas de salto para no gastar combustible.

Cada camino está diferenciado por un color, para que el usuario tenga una referencia a la hora de recordar qué camino es más sencillo en ciertas partes de la partida, ya que esa es la finalidad del juego, poder avanzar por los caminos más sencillos y gastar el mínimo combustible posible.

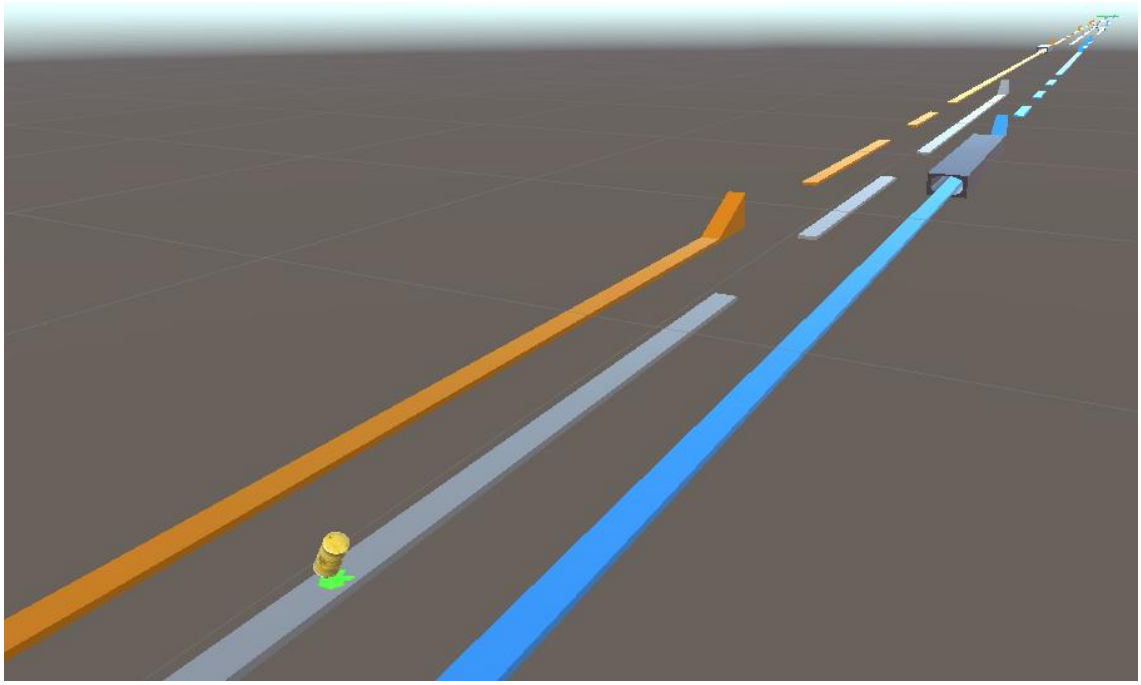
Para el desarrollo del nivel, se han empleado muchas horas en equilibrar la dificultad, de manera que si el jugador utiliza los caminos correctos, ya que con un vistazo se puede intuir cuales son, siempre puede llegar a alcanzar un barril de combustible.



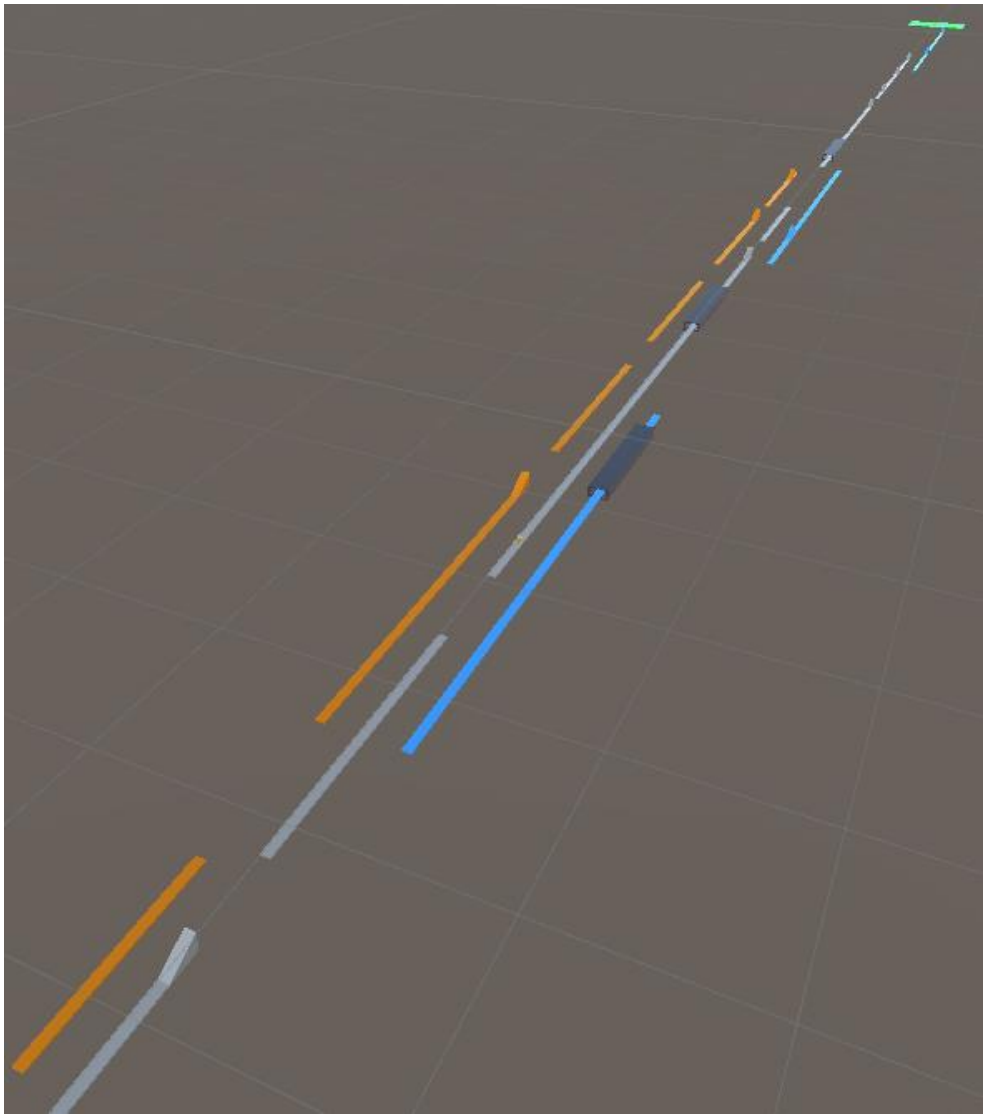
Para el primer tramo del nivel, hasta el primer barril de combustible, hay una clara diferenciación en los distintos caminos, de modo que el naranja ya aparenta cierta dificultad comparado con el azul o el gris. Ya que es el principio del nivel, para facilitar la asimilación, disponemos que tanto el gris como el azul son buenas alternativas para dar los primeros saltos. Si usamos el gris, podremos pasar a través de un túnel, lo que nos servirá para dar a entender al jugador que se puede pasar sin problema, siempre y cuando no colisionemos contra él. Si saltamos al azul, el jugador aprenderá a usar la rampa de salto que está ubicada al final de este primer tramo. Si por el contrario, decide usar el naranja y no cambiar de carril, el jugador aprenderá que el fuel se le acabará pronto y no podrá avanzar demasiado por ese carril.

Esta metodología es la que se sigue aplicando el resto del nivel, aunque con diversas modificaciones que hacen aumentar un poco la dificultad, con cambios en los caminos no tan predecibles.

Tras el primer tramo, el jugador no tardará en asimilar el control del juego y lo que puede o no hacer con la nave. Ya que los caminos correctos durante los primeros segundos del juego, son tramos bastante largos.



Una vez hemos alcanzado el primer barril, la dificultad se incrementa con el aumento de tramos cortos que el jugador debe evitar para no gastar combustible. Para ello, se han añadido más rampas, que facilitan el salto entre carriles. Además se eliminan tramos de algunos caminos por un tiempo determinado, para que el jugador deba elegir rápidamente por cual continuar. El jugador que llega a esta fase, ya ha adquirido la habilidad suficiente como para poder superar el nivel sin muchos más problemas.



Para la parte final del nivel, que se sitúa cerca del segundo barril de fuel, los saltos empiezan a ser mayores, haciendo que el jugador deba medir mucho el salto para poder llegar. Además, el último elemento de dificultad son los túneles con el corte del final del camino. Estos tramos requieren buena habilidad por parte del jugador, ya que justo al acabar el túnel, deberá cambiar muy rápido de carril para no caer al vacío. También apreciamos otro incremento de rampas de salto para poder cambiar constantemente de carril. Este tramo, permite poder llegar al final del juego sin haber utilizado combustible, siempre y cuando el jugador sepa elegir bien el camino que le va a ayudar más. Sin embargo, ese combustible le vendrá bien al jugador que no vaya por el camino correcto, ya que debemos otorgar un margen de error, que en este caso son unos cinco saltos extra.

6. Manual de usuario

Requisitos para la ejecución de Gravity Dimension en formato APK detallados:

Android:

OS 4.1 o posterior;
ARMv7 CPU con soporte NEON o CPU Atom;
OpenGL ES 2.0 o posterior.

Instrucciones de uso:

Se usará la pantalla táctil del smartphone para seleccionar las opciones que aparecen en pantalla.

Controles:

Para controlar la nave, existen dos movimientos, el salto de la nave, el cual se accionará con un toque en cualquier parte de la pantalla, y el movimiento horizontal, el cual se ejecuta con el movimiento del smarphone usando el giroscopio de forma horizontal.

Botón de pausa:

En la esquina superior izquierda, el jugador puede hacer uso del modo pausa, el cual permite pausar el juego y retomarlo, volviendo a la partida, además de reiniciar el nivel desde el principio, o volver a la pantalla inicial del juego.

7. Conclusiones

La realización de este trabajo, ha servido principalmente para tener una visón del desarrollo completo de un videojuego desde cero, con una fecha límite. Aprender a gestionar el tiempo de cada fase, junto con los problemas que han aparecido que no se tenían en cuenta, ha servido como una gran experiencia de cara a futuros proyectos. La forma de organizar el trabajo, incluso el haber visto qué fases son más importantes, o a qué dedicarle más tiempo, ha sido un gran aporte.

Otro factor importante para el desarrollo de este juego, ha sido la búsqueda de recursos, los cuales han sido diversos, tanto para imágenes, tipografía, Asset Store.

También cabe mencionar el aprendizaje respecto al desarrollo para Android. Este aspecto ha servido para aprender cómo adaptar elementos a distintos tamaños de pantalla, además de las herramientas requeridas en Unity para exportar un archivo APK.

Las horas invertidas en convertir el nivel jugable de Gravity Dimension en algo equilibrado es otro factor a tener en cuenta de cara a futuros proyectos, sobretodo en este tipo de géneros de juego. Requiere un tiempo extra hacer pruebas y editar el nivel, además de probar con distintas personas.

Saber adaptar un estilo musical al juego, ha sido una satisfactoria experiencia. Para ello, ha sido imprescindible estudiar diferentes tipos de juegos similares, como Wipeout, y demás estilos arcade frenéticos.

Como crítica, cabe destacar la falta de más niveles para finalizar el proyecto tal y como se planeó. Debido a la alta demanda de tiempo y pruebas para equilibrar la dificultad del nivel, ha sido inviable poder realizar más de uno.

En lo que respecta a la planificación, durante el desarrollo, han surgido diversos cambios de lo planificado inicialmente. Esta modificación radica en la falta de experiencia en planificación de videojuegos y el desconocimiento de la dificultad de diferentes objetivos, además de no saber qué objetivos debían trabajarse juntos.

Por ejemplo, en la planificación inicial, se plantean diversas semanas para desarrollar scripts solamente y por otro lado, la creación del nivel más adelante con semanas exclusivas a esa tarea. Pero la realidad ha sido que los scripts y la creación del nivel van de la mano, alternando correcciones de ambos objetivos.

8. Pruebas con usuarios

Se han realizado pruebas con dos usuarios que poseen dos tipos de perfiles diferentes. Uno de ellos, jugador habitual de videojuegos con gran experiencia tanto en smartphones como en pc y videoconsolas. El otro usuario, con muy poca experiencia en videojuegos en general.

La idea de probar con estos dos perfiles tiene dos finalidades. La finalidad de probar con un jugador experimentado, es la de testear si el nivel tiene el suficiente reto para reintentarlo en caso de muerte o en su contra, demasiado asequible. Por otro lado, un usuario sin experiencia, para poder testear si la curva de aprendizaje permite un avance en jugadores sin experiencia y cuántos reintentos deben realizar para adaptarse.

En el caso del jugador experimentado, hubo diferentes reintentos al principio, pero el jugador se adaptó relativamente rápido al control. Una vez el control adaptado, hubieron otra serie de reintentos para tener en cuenta el gasto de combustible. Una vez cogido por la mano el control y las rampas, además de los caminos más o menos correctos, el jugador pudo finalizar el nivel correctamente.

En el caso del jugador no experimentado, la adaptación al control costó algunos reintentos, pero de forma muy similar al anterior. Lo que indica que el control con el giroscopio es fácil de asimilar. Lo que costó más fue evitar las colisiones en túneles y la gestión de combustible. Debido a esos factores, los reintentos fueron bastantes, aunque finalmente, y tras un tiempo bastante diferenciador, el jugador pudo finalizar el nivel.

El test, tuvo como resultado algo ya esperado, y es que es un videojuego enfocado a un público más harcore o de alto nivel, tanto por la velocidad de reacción que requiere como por el control milimetrado para ciertas acciones. Aun así, los dos jugadores se desenvolvieron bastante bien con el juego y lo más importante, disfrutaron superando el nivel.

9. Glosario

Arcade: El término *arcade* se refiere a los videojuegos clásicos o que recuerdan a las máquinas del mismo nombre. También se usa para diferenciar a los simuladores. *Arcade*, en este sentido, suele referirse a los juegos relativamente fáciles de jugar o que no responden fielmente a la gravedad y otras fuerzas.

Speedrunners: Disciplina dentro del mundo de los videojuegos dedicada a la finalización de determinados títulos del sector en el menor tiempo posible.

Giroscopio: Dispositivo integrado en la mayoría de smartphones con la finalidad de detectar cualquier tipo de rotación en cualquiera de sus ejes.

Assets: Recursos almacenados dentro de un proyecto de Unity con la finalidad de integrarlos en una escena de juego.

GameObjects: Elementos dentro de una escena de juego de Unity.

10. Bibliografía

1- https://www.lavozdeasturias.es/noticia/informacion/2018/04/24/speedrunner/000315245716382312516 72.htm (18/05/19)

2- https://www.assetstore.unity3d.com/ (25/05/19)

3- https://www.crunchbase.com/person/daniel-boutros#section-overview (02/06/19)

4- https://es.wikipedia.org/wiki/Arcade (08/06/19)

5- http://www.gamerdic.es/termino/endless-runner/ (08/06/19)

6- https://es.wikipedia.org/wiki/Videojuego_de_plataformas (08/06/19)