Universidad del País Vasco
Euskal Herriko Unibertsitatea

Facultad de Informática

Grado de Ingeniería Informática

▪ Proyecto Fin de Grado ▪

Computación

# Creación del Entorno Virtual para un videojuego

Egoitz García Alonso

Septiembre 2016

informatika fakultatea
facultad de informática

# Resumen

Proyecto de Fin de Grado de la especialidad de Computación. Se ha creado un prototipo de videojuego llamado *Just 1 More Time*. Se trata de un videojuego en 2D, desarrollado en Unity para su uso en PC. Los *Scripts* se han realizado en el lenguaje C# y se ha utilizado Photoshop para la creación del apartado gráfico.

En este juego el protagonista debe superar las 11 fases que lo componen para superarlo, si fracasa en alguna de ellas se verá obligado a comenzar de nuevo desde el principio. El protagonista es capaz de correr, saltar, y mantenerse aferrado a paredes.

# Índice


***CAPÍTULO 1: INTRODUCCIÓN ..... 1***

1.1 CONTEXTO ..... 1
1.2 PROPUESTA ..... 1
1.3 MOTIVACIÓN PERSONAL ..... 2

***CAPÍTULO 2: OBJETIVOS Y ANTECEDENTES ..... 3***

2.1. TAREAS DE INVESTIGACIÓN ..... 3
2.1.1. GÉNEROS ..... 3
2.1.1.1. BEAT 'EM UP ..... 3
2.1.1.2. LUCHA ..... 4
2.1.1.3. ACCIÓN EN PRIMERA PERSONA ..... 4
2.1.1.4. ACCIÓN EN TERCERA PERSONA ..... 4
2.1.1.5. INFILTRACIÓN ..... 5
2.1.1.6. PLATAFORMAS ..... 5
2.1.1.7. ARCADE ..... 5
2.1.1.8. SPORT ..... 5
2.1.1.9. CARRERAS ..... 6
2.1.1.10. AGILIDAD MENTAL ..... 6
2.1.1.11. MUSICALES ..... 6
2.1.1.12. PARTY GAMES ..... 6
2.1.1.13. JUEGOS ON-LINE ..... 7
2.1.2. ELECCIÓN DE GÉNERO ..... 7
2.2. TAREAS DE DISEÑO ..... 7
2.2.1. DISEÑO DE NIVEL ..... 7
2.2.2. DISEÑO DE GRÁFICOS ..... 8
2.2.3. ELECCIÓN DEL ESTILO ARTÍSTICO ..... 8
2.3. OBJETIVOS DEL PROYECTO (ALCANCE Y EXCLUSIONES) ..... 8
2.4. HERRAMIENTAS UTILIZADAS ..... 9
2.4.1. ANÁLISIS DE *GAME ENGINES* ..... 9
2.4.1.1. UNITY3D ..... 9
2.4.1.2. UNREAL ENGINE 4 ..... 11
2.4.1.3. ELECCIÓN DE *GAME ENGINE* ..... 13
2.4.2. HERRAMIENTAS DE CREACIÓN DE GRÁFICOS ..... 14
2.5. APRENDIZAJE ..... 14
2.5.1. APRENDIZAJE DE UNITY ..... 14
2.5.1.1. ROLL-A-BALL TUTORIAL ..... 14
2.5.1.2. 2D UFO TUTORIAL ..... 15
2.5.1.3. TANKS TUTORIAL ..... 15
2.6. ANÁLISIS DE RIESGOS ..... 16
2.6.1. TIEMPO ..... 16
2.6.2. PÉRDIDA DE DATOS ..... 17
2.6.3. MEMORIA ..... 17
2.7. ANÁLISIS DE FACTIBILIDAD ..... 17

***CAPÍTULO 3: GESTIÓN DEL PROYECTO ..... 19***

3.1. PROTOTIPO INICIAL ..... 19
3.1.1. DISEÑO DE NIVEL ..... 19
1.4 GRÁFICOS ..... 19
3.1.2. COMENTARIOS TÉCNICOS ..... 20
3.2. VERSIÓN *ALPHA* ..... 21
3.2.1. DISEÑO DE NIVEL ..... 21
3.2.2. GRÁFICOS ..... 21
3.2.3. COMENTARIOS TÉCNICOS ..... 22
3.3. VERSIÓN BETA ..... 25
3.3.1. DISEÑO DE NIVEL ..... 25
3.3.2. GRÁFICOS ..... 28
3.3.3. COMENTARIOS TÉCNICOS ..... 29

***CAPÍTULO 4: DISEÑO ..... 35***

4.1. CONCEPTO GENERAL ..... 35
4.2. DISEÑO DE NIVEL ..... 35
4.2.1. NIVEL 1 ..... 35
4.2.1.1. DEFINICIÓN ..... 35
4.2.1.2. INFORMACIÓN ADICIONAL ..... 36
4.2.2. NIVEL 2 ..... 36
4.2.2.1. DEFINICIÓN ..... 36
4.2.2.2. INFORMACIÓN ADICIONAL ..... 37
4.2.3. NIVEL 3 ..... 37
4.2.3.1. DEFINICIÓN ..... 37
4.2.3.2. INFORMACIÓN ADICIONAL ..... 38
4.2.4. NIVEL4 ..... 38
4.2.4.1. DEFINICIÓN ..... 38
4.2.4.2. INFORMACIÓN ADICIONAL ..... 39
4.2.5. NIVEL 5 ..... 39
4.2.5.1. DEFINICIÓN ..... 39
4.2.5.2. INFORMACIÓN ADICIONAL ..... 40
4.2.6. NIVEL 6 ..... 40
4.2.6.1. DEFINICIÓN ..... 40
4.2.6.2. INFORMACIÓN ADICIONAL ..... 41
4.2.7. NIVEL 7 ..... 41
4.2.7.1. DEFINICIÓN ..... 41
4.2.7.2. INFORMACIÓN ADICIONAL ..... 42
4.2.8. NIVEL 8 ..... 43
4.2.8.1. DEFINICIÓN ..... 43
4.2.8.2. INFORMACIÓN ADICIONAL ..... 43
4.2.9. NIVEL 9 ..... 44
4.2.9.1. DEFINICIÓN ..... 44
4.2.9.2. INFORMACIÓN ADICIONAL ..... 44
4.2.10. NIVEL 10 ..... 44
4.2.10.1. DEFINICIÓN ..... 44
4.2.10.2. INFORMACIÓN ADICIONAL ..... 45
4.2.11. NIVEL 11 ..... 46

4.2.11.1. Definición ..... 46
4.2.11.2. Información adicional ..... 49
4.3. Gráficos estáticos ..... 49
4.3.1. Gráficos del nivel ..... 49
4.3.2. Gráficos de elementos del juego ..... 50
4.4. Animaciones Personaje ..... 51
4.4.1. Reposo ..... 51
4.4.1.1. Fotogramas ..... 51
4.4.1.2. Condición de activación ..... 51
4.4.2. Correr ..... 51
4.4.2.1. Fotogramas ..... 52
4.4.2.2. Condición de activación ..... 52
4.4.3. Saltar ..... 52
4.4.3.1. Fotogramas ..... 52
4.4.3.2. Condición de activación ..... 53
4.5. Interfaz de usuario (UI) ..... 53
*4.6. Scripts* ..... 53
4.6.1. *Scripts* de cámara ..... 53
4.6.2. *Scripts* de Personaje ..... 54
4.6.3. *Scripts* de entorno ..... 58
***CAPÍTULO 5: RESULTADOS ..... 63***
***CAPÍTULO 6: CONCLUSIONES Y POSIBILIDADES DE MEJORA ..... 65***
***CAPÍTULO 7: REFERENCIAS ..... 67***
***CAPÍTULO 8: ANEXOS ..... 69***
8.1. Manual de instrucciones ..... 69

# Índice de Figuras

1: 3.1 Nivel prototipo pruebas ..... 19
2: 3.2 Nivel prototipo pruebas + jugador ..... 20
3: 3.3 *Sprites* jugador versión *Alpha* ..... 22
4: 3.4 Animación de salto versión *Alpha* ..... 22
5: *3.5 Collider* del jugador ..... 23
6: 3.6 Controlador de animaciones de jugador (*Alpha*) ..... 23
7: 3.7 Zonas de nivel Beta ..... 26
8: 3.8 Inicio nivel Beta ..... 26
9: 3.9 Zona central nivel Beta ..... 27
10: 3.10 Zona inferior nivel Beta ..... 27
11: 3.11 Zona superior nivel Beta ..... 27
12: 3.12 Zona final nivel Beta ..... 28
13: 3.13 Puerta versión Beta ..... 28
14: 3.14 Obstáculos versión Beta ..... 28
15: 3.15 Llaves versión Beta ..... 28
16: 3.16 Meta versión Beta ..... 29
17: 3.17 Coleccionables versión Beta ..... 29
18: 4.1 Nivel 1 ..... 36
19: 4.2 Nivel 2 ..... 37
20: 4.3 Nivel 3 ..... 38
21: 4.4 Nivel 4 ..... 39
22: 4.5 Nivel 5 ..... 40
23: 4.6 Nivel 6 ..... 41
24: 4.7 Nivel 7 (Posición 1) ..... 42
25: 4.8 Nivel 7 (Posición 2) ..... 42
26: 4.9 Nivel 8 ..... 43
27: 4.10 Nivel 9 ..... 44
28: 4.11 Nivel 10 ..... 45
29: 4.12 División nivel 11 ..... 47
30: 4.13 Nivel 11 (Zona superior) ..... 47
31: 4.14 Nivel 11 (Subida) ..... 48
32: 4.15 Nivel 11 (Zona Central) ..... 48
33: 4.16 Nivel 11 (Zona inferior) ..... 48
34: 4.17 Modelo suelos, plataformas y paredes ..... 49
35: 4.18 Modelo portal ..... 49
36: 4.19 Modelo *hazard* ..... 50
37: 4.20 Modelos puerta y llave ..... 50
38: 4.21 Fotogramas animación moneda ..... 50
39: 4.22 Fotogramas animación reposo ..... 51
40: 4.23 Fotogramas animación correr ..... 52
41: 4.24 Fotogramas animación salto ..... 52

# Índice de Códigos

*SCRIPT 3.1: PROTOTIPO SCRIPT JUGADOR* ..... **20**
*SCRIPT 3.2: ALPHA SCRIPT JUGADOR* ..... **23**
*SCRIPT 3.3: BETA SCRIPT JUGADOR* ..... **29**
*SCRIPT 3.4: BETA SCRIPT PUERTAS* ..... **31**
*SCRIPT 3.5: BETA SCRIPT PLATAFORMAS MÓVILES* ..... **32**
*SCRIPT 3.6: BETA SCRIPT CÁMARA* ..... **33**
*SCRIPT 4.1: SCRIPT CÁMARA FINAL* ..... **54**
*SCRIPT 4.2: SCRIPT JUGADOR FINAL* ..... **54**
*SCRIPT 4.3: SCRIPT ABRIR PUERTAS FINAL* ..... **59**
*SCRIPT 4.4: SCRIPT MECANISMO TRAMPA FINAL* ..... **59**
*SCRIPT 4.5: SCRIPT PLATAFORMA MÓVIL FINAL* ..... **60**
*SCRIPT 4.6: SCRIPT PLATAFORMA INTERMITENTE FINAL* ..... **61**

# Capítulo 1: Introducción

## 1.1 Contexto

Entretenimiento. Una palabra tan simple, pero que a la vez tantas cosas mueve en nuestra sociedad. Desde siempre, el ser humano ha buscado entretenerse de una forma u otra a lo largo de las épocas, ya fuese mediante deportes, estudio, pintura, escultura, escritura, leer, o incluso simplemente pasear por el campo o la ciudad contemplando el paisaje.

Sin embargo, el tiempo pasa. Actos que antaño resultaban entretenidos para la gente hoy en día no son más que vagos recuerdos de historias que nos contaban. Los tiempos cambian, y el formato del entretenimiento también, especialmente desde que la tecnología entró en nuestras vidas de una forma tan estrecha como ocurre actualmente.

En pleno siglo XXI en el que nos encontramos, raro es ver a una persona que no tenga contacto con dispositivos electrónicos de entretenimiento, ya sea un ordenador, un móvil, o una videoconsola. Los dispositivos de entretenimiento son varios, pero todos tienen algo en común: Videojuegos.

Cuando comenzó la época del cine, la gente miraba maravillada esa nueva forma de entretenimiento, les permitía ver lugares y situaciones variadas que de otra forma jamás habrían llegado experimentar. Cuando llegaron los videojuegos, esa sensación de inmersión renació con más fuerza, ya que ahora podíamos ser nosotros mismos los protagonistas de esas historias que leíamos, de esas películas que veíamos, podíamos transformarnos en los héroes que de niños siempre hemos querido ser.

Desde que comenzó la era de los videojuegos hasta hoy en día, el mercado ha ido creciendo cada vez a mayor escala, si bien antes para poder jugar a un videojuego tenías que poseer una consola, hoy en día basta con tener un teléfono móvil para poder jugar a alguna de estas joyas del entretenimiento capaces de hacer que el tiempo pase volando.

Actualmente el sector de los videojuegos está en auge [ANT 2009], y si alguien quiere dedicarse al entretenimiento de calidad, es un sector que no puede pasar por alto.

## 1.2 Propuesta

Nuestra propuesta pretende ser un primer paso dentro del mundo de los videojuegos, con la intención de profundizar en lo que yace tras el producto final que conocemos para conocer los recovecos de la creación de un videojuego. Para llevarlo a cabo se creará el proyecto llamado *Just 1 More Time*.

*Just 1 More Time* se trata de un proyecto que englobará lo aprendido en un breve lapso de aprendizaje. Para ello, se creará un videojuego de tipo PvE (Jugador contra Entorno) en el que se pueda observar lo aprendido.

Para la creación de *Just 1 More Time* se realizará un estudio de los distintos *Game Engines* disponibles en el mercado, tras lo que se escogerá uno de ellos para realizar el aprendizaje del mismo, así como las pruebas pertinentes y el producto final.

## 1.3 Motivación personal

Desde que era pequeño me ha encantado todo lo relacionado con la imagen, ya fuesen películas, fotografías, cuadros, todo en general me gustaba, hasta que descubrí por primera vez los videojuegos.

Desde que los vi por primera vez tuve claro que quería dedicarme a ello, quería dedicarme a crear esos mundos llenos de color que veía, quería dar vida a esos personajes con los que niños como yo podrían vivir aventuras e identificarse.

Y así, mientras otros niños de mi edad soñaban con ser astronauta, bombero, futbolista, doctora, profesora, o mil cosas más que se les ocurrieran, yo siempre decía que quería crear videojuegos. Así, el tiempo pasaba y la gente olvidaba esos sueños que de niño tenían, el que quería ser astronauta estudiaba idiomas, el que quería ser bombero estudiaba para profesor, el que ansiaba ser futbolista estudiaba psicología. Todos cambiaron su meta, salvo yo.

Seguí estudiando, siempre siguiendo esa ruta que de niño me había fijado, siempre buscando el camino que más me acercara a esa meta que cada vez más posible parecía de alcanzar. Y de esa forma acabé en Ingeniería Informática, otro puente más que cruzar en mi camino, cada vez más cercano a lo que yo había sabido desde que era pequeño que era mi pasión.

Sin embargo, tras terminar los cuatro años que conforman la carrera sentí que necesitaba adentrarme más en ese mundo al que yo sabía que pertenecía, y al encontrarme frente al proyecto de Fin de Grado pensé que era una gran oportunidad para hacer algo ligado más que nada a mi vocación.

De modo que decidí que mi trabajo consistiría en ahondar en ese mundo que hasta ahora solo había mirado desde fuera, sabía que no sería fácil, pero el mero hecho de tratarse de aquello que en verdad me apasiona hacía que no pudiera más que sonreír ante lo que me esperaba.

# Capítulo 2: Objetivos y Antecedentes

El proyecto consiste en la creación de un prototipo de videojuego, para ello se realizarán diversas tareas de investigación con el fin de obtener las herramientas y el conocimiento necesarios para ello.

En este capítulo se habla acerca de los objetivos que se pretenden alcanzar en el proyecto, así como de las tareas previas a realizar para llevarlos a cabo. Se detallan las tareas de investigación acerca de géneros de videojuegos, *Game Engines* y herramientas utilizadas. También se comentan las elecciones realizadas y sus razones, así como el proceso de aprendizaje realizado y los distintos análisis llevados a cabo.

## 2.1. Tareas de Investigación

Las tareas de investigación se dividirán en dos grupos principales.

Tareas de investigación de géneros de videojuegos, para seleccionar el adecuado en base a nuestra situación actual.

Tareas de diseño, centradas en encontrar un estilo gráfico que utilizar en el juego que se va a realizar, así como en encontrar una fórmula que seguir a la hora de diseñar los niveles.

### 2.1.1. Géneros de videojuegos

¿Qué es un género de videojuego? En palabras de Simone Belli y Cristian López:

"Un género de videojuego designa un conjunto de juegos que poseen una serie de elementos comunes. A lo largo de la historia de los videojuegos aquellos elementos que han compartido varios juegos han servido para clasificar como un género a aquellos que les han seguido, de la misma manera que ha pasado con la música o el cine.

Los videojuegos se pueden clasificar como un género u otro dependiendo de su representación gráfica, el tipo de interacción entre el jugador y la máquina, la ambientación y su sistema de juego, siendo este último el criterio más habitual a tener en cuenta."[SIM CRI 2008]

A continuación procedemos a analizar diferentes tipos de géneros utilizados en la industria de los videojuegos, identificando sus rasgos distintivos en diversas áreas.

#### 2.1.1.1. Beat 'em Up

Género similar al de Lucha, usualmente compuesto de varios niveles con gráficos 2D en los que el jugador debe ir avanzando mientras derrota a las oleadas de enemigos que le salen al paso. Es común que varios jugadores puedan jugar a la vez en un modo cooperativo para hacer así más fácil el progreso en el juego.

#### 2.1.1.2. Lucha

Juegos basados en la recreación de combates entre dos o más contendientes, ya sean controlados mediante jugadores o por la propia inteligencia artificial del juego.

Los personajes suelen contar con una amplia variedad de habilidades y movimientos que realizar durante los combates, ya sean basados en artes marciales, diversos estilos de lucha con armas o incluso magia dependiendo del contexto en el que se realice el juego.

En sus comienzos la mayoría de estos juegos optaban por una estética 2D con vista lateral, pero con el paso de los años el género ha evolucionado incluyendo también elementos 3D a los mismos. Un claro ejemplo serían los juegos de la serie "*Naruto*": al principio se trataba de juegos de lucha en un entorno 2D, pero con el paso de las generaciones y la llegada de nuevas consolas se han transformado en juegos de lucha en un entorno 3D.

#### 2.1.1.3. Acción en primera persona

Más conocidos por sus siglas en inglés "FPS" (First Person Shooter), en estos juegos el jugador goza de una vista en primera persona que representaría el campo de visión de su personaje, realizando las acciones pertinentes como correr, saltar, etc... desde este punto de vista.

Usualmente en este género de juegos el jugador encarna a algún tipo de soldado, héroe o similares que debe realizar distintas misiones dependiendo del guión del juego y suele ir equipado con armas de fuego para acabar con sus enemigos.

Este género suele usar gráficos en 3D para aumentar la inmersión del jugador dentro del juego y causar la sensación de que es él quien se encuentra dentro del juego.

El mayor auge de este género de juegos se encuentra en PC, ya que suelen requerir una gran capacidad de reacción que en consolas no se puede alcanzar con tanta precisión debido al sistema de control usado.

#### 2.1.1.4. Acción en tercera persona

Estos juegos son en su gran mayoría similares a los FPS, salvo que en esta ocasión la cámara se suele situar en un plano detrás del personaje y ocasionalmente con una perspectiva isométrica.

Si bien con este tipo de cámara se pierde precisión en algunos aspectos, también se ganan otros beneficios, como ganar una gran libertad de movimiento.

Ejemplos de este género son los juegos de la saga *"Grand Theft Auto"*.[$3DJ_a$]

#### 2.1.1.5. Infiltración

Similares a los juegos de acción en tercera persona y popularizados a partir de la creación de la saga *"Metal Gear Solid"* [$HOB_a$]. En los juegos de infiltración el objetivo no suele ser el conflicto directo con el enemigo, sino en el uso del sigilo, la estrategia y la inteligencia del usuario para avanzar.

Suelen contar con una vista en tercera persona en la que vemos a nuestro personaje para controlarlo mientras tenemos constancia de nuestro entorno. Se realizan en gráficos 3D.

#### 2.1.1.6. Plataformas

En este género de juegos, uno de los primeros y más sencillos en la historia de los videojuegos, se observa claramente cómo ha ido evolucionando el mundo de los videojuegos en lo que a posibilidades de diseño se refiere.

El planteamiento general de estos juegos siempre es similar. El jugador controla a un personaje que debe ir recorriendo diferentes niveles a la par que sortea los obstáculos, trampas y enemigos que se hallan en los mismos.

Para realizar el objetivo del juego, el personaje controlado por el jugador cuenta con diversas habilidades, siendo las básicas el hecho de correr y saltar y añadiendo otras en función del diseño, ya sea nadar, escalar, agacharse, etc.

También suele ser común el hecho de que en los niveles los jugadores encuentren unos objetos especiales, comúnmente conocidos como *"Power Ups",* que les otorgan habilidades especiales durante un tiempo determinado, ya sea la habilidad de volar, invulnerabilidad, o lo que se requiera en el juego.

Al comienzo estos juegos se desarrollaban en un entorno puramente 2D, como en los clásicos *"Super Mario Bros"* o *"Megaman",* pero con la llegada de las nuevas consolas pasaron de un entorno 2D a un entorno 3D en el que los niveles se extendían en cualquier dirección, siendo pionero de este estilo el conocido *"Super Mario 64"*[$MER_a$]*,* innovando además en el movimiento de la cámara en los videojuegos.

#### 2.1.1.7. Arcade

Género nacido en la época de las recreativas, en plena década de 1980, juegos simples de acción rápida que destacaban por la jugabilidad, no era necesaria historia alguna, eran juegos simples, largos y repetitivos.

Ejemplos reconocibles del género arcade serían *"Space Invaders"* o *"Pac-Man"*.

#### 2.1.1.8. Sport

Se trata de un género cuyo objetivo es simular deportes existentes, ya sea fútbol, tenis, baloncesto, golf o cualquier otro.

En ocasiones se han modificado los deportes al adaptarlos al videojuego, incluyendo añadidos fantasiosos o diversas alteraciones al original, como por ejemplo con el juego *"Mario Power Tennis"*.

#### 2.1.1.9. Carreras

Género basado en las carreras de vehículos de la actualidad, encontrándose dentro de él dos tipos claramente diferenciables: los simuladores de carreras realistas y los juegos de carreras fantásticos. Abarcando así este género desde simuladores extremadamente fieles a la realidad hasta locuras fantasiosas donde la diversión es el objetivo principal.

Este género se basa en la idea normal de las carreras, que es llegar a meta antes que tus oponentes, dependiendo del juego añadiendo objetivos adicionales, atajos o maneras de estorbar a los otros participantes de la carrera, siendo un ejemplo de esto último el reciente *"Mario Kart 8"*[$HOB_b$] en el que no solo se recorrían circuitos normales, sino que también incluían partes submarinas, aéreas o partes del circuito en las que anulaban la gravedad creando así maravillas visuales para el jugador.

#### 2.1.1.10. Agilidad mental

Ya sea con el objetivo directo de entrenar la mente o para pasar el rato, en esté género el objetivo es resolver ejercicios o puzles cada vez de mayor dificultad para entrenar la mente del jugador.

El juego más conocido de este género es el famoso *"Tetris",* mientras que también hay otros juegos que de una manera distinta nos hacen entrenar nuestra mente a la par que nos entretienen con una historia absorbente, como podrían ser los juegos de la saga del *"Profesor Layton"*.

#### 2.1.1.11. Musicales

Como su nombre indica, estos juegos giran por completo en torno a la música, yendo desde "Guitar Hero", donde en lugar de mando usabas una guitarra especial para jugar mientras tocabas diversas melodías, hasta juegos de karaoke como *"Singstar",* donde micrófono en mano debías cantar correctamente la canción para aumentar tu puntuación.

Cabe mencionar también que en esta categoría podrían entrar también otros juegos musicales como por ejemplo *"Theatrhythm Final Fantasy",* juego de ritmo donde debías introducir diferentes comandos al ritmo de la música.

#### 2.1.1.12. Party Games

En este género se sigue la mecánica de los juegos de mesa tradicionales de ir avanzando por un tablero para ganar, pero con la diferencia de que aquí además hay que superar diversos minijuegos o pruebas durante la competición. El mayor ejemplo de este género serían los juegos de la saga *"Mario Party"*.

#### 2.1.1.13. Juegos On-Line

Cada vez más extendido y no como género en sí, sino como un añadido cada vez mayor a los distintos géneros de videojuegos existentes, se encuentra el On-Line. Hoy en día pocos son los juegos que no cuentan con algún componente online, ya sea algún tipo de modo cooperativo o competitivo, o simplemente algún tipo de red social de algún tipo para comparar los logros dentro del juego.

Sin embargo, también existen juegos y géneros que son exclusivamente online, como podrían ser los del género MOBA (Multiplayer Online Battle Arena).

En los MOBA, normalmente dos equipos de jugadores luchan entre sí en una arena en la que deben avanzar para alcanzar la base del enemigo y así ganar la partida. Se trata de un género que ha ido en auge últimamente, gozando algunos juegos de esta categoría del apelativo e-sports, o deporte electrónico, como por ejemplo el afamado *"League of Legends"* (LoL) [$MER_b$] o el *"Defense of the Ancients 2"* (DotA2), que cuentan ya con torneos nacionales e internacionales y cuya asistencia no tiene nada que envidiar a los deportes tradicionales como el fútbol.

### 2.1.2. Elección de Género

Tras analizar los distintos tipos de géneros de videojuegos existentes, y juzgando el tiempo disponible para realizar el proyecto, se ha decidido que el objetivo del proyecto será crear un prototipo de juego del género plataformas.

## 2.2. Tareas de diseño

Dentro de esta categoría encontramos dos aspectos principales:

Diseño de Nivel, donde se detallan los aspectos principales de los niveles del juego, así como su objetivo.

Diseño de gráficos, donde se detalla el estilo que se va a utilizar en la creación de los gráficos del juego.

### 2.2.1. Diseño de Nivel

En *Just 1 More Time* el jugador se enfrentará a un total de once niveles que deberá superar. Para superar los niveles exitosamente, deberá recoger un total de siete coleccionables repartidos por cada nivel y llegar a salvo al transportador para avanzar al nivel siguiente.

Los primeros dos niveles son sencillos y fáciles de completar, apenas hay peligro o desafío en ellos ya que su objetivo es que el jugador se acostumbre a los controles y comprenda el funcionamiento del juego.

Los ocho niveles siguientes serán de un estilo que llamaremos desafíos individuales. En cada uno de estos niveles el jugador tendrá que seguir una mecánica diferente para superarlo.

El último nivel será mayor que los anteriores en tamaño, ya que en este se combinarán varias de las mecánicas introducidas en niveles previos, así como elementos nuevos propios del nivel final.

Tras haber superado el nivel final se habrá completado el juego, sin embargo, si muere por el camino el usuario estará condenado a comenzar de nuevo desde el primer nivel.

### 2.2.2. Diseño de Gráficos

Para nuestro juego de plataformas vamos a optar por unos gráficos 2D, ya que el estilo 2D nos ofrece mayor facilidad tanto a la hora de diseñar los niveles como a la hora de trabajar sobre el motor observando el tiempo del que disponemos.

### 2.2.3. Elección del estilo artístico

Acerca del estilo artístico que se usará en *Just 1 More Time*, se ha optado por usar un estilo de gráficos sencillo, del estilo *Stickman*, para así poder realizar los gráficos manualmente, ya que no se poseen cualidades artísticas lo suficientemente trabajadas y se desea tener cierto control sobre las imágenes utilizadas.

El entorno también estará formado por gráficos sencillos que faciliten la interpretación por parte del jugador.

## 2.3. Objetivos del proyecto (Alcance y Exclusiones)

El alcance del proyecto se recoge en una serie de objetivos que pueden ser divididos en dos secciones principales: Objetivos de aprendizaje y objetivos de diseño.

Los objetivos de aprendizaje se centran en la búsqueda de herramientas y en aprender el uso de las mismas, así como de discernir la más adecuada para cada situación:

- Búsqueda de *Game Engines*
- Elección de *Game Engine*
- Aprendizaje de *Game Engine*
- Pruebas con *Game Engine*
- Investigación de géneros de videojuegos
- Elección de género para el proyecto
- Elección de estilo gráfico

Los objetivos de diseño se centran en el uso de las herramientas obtenidas para comenzar a crear el producto final.

- Diseño del tipo de juego
- Elección del rasgo distintivo del juego

- Diseño de niveles del juego
- Diseño de elementos del escenario
- Diseño de animaciones del juego
- Creación de elementos gráficos necesarios

Se excluirán del proyecto los elementos relacionados con los efectos sonoros de los videojuegos. Elementos como efectos de sonido y similares quedan excluidos del proyecto.

La música de fondo no queda excluida del proyecto.

## 2.4. Herramientas utilizadas

Para la realización del proyecto se prevé el uso de dos herramientas principales: un *Game Engine* y una herramienta de creación de gráficos.

### 2.4.1. Análisis de *Game Engines*

Tras analizar los *Game Engines* disponibles en el mercado, se decidió hacer una búsqueda más detallada de las características de los dos *Game Engines* más conocidos: Unity3D y Unreal Engine 4.

#### 2.4.1.1. Unity3D

- Software líder a nivel mundial en la industria de los juegos [$UNI_a$]

Unity es el software de desarrollo de juegos predominante a nivel mundial. Se crean más juegos con Unity que con cualquier otra tecnología para juegos.

Solo en el primer trimestre de 2016, los jugadores de todo el mundo han descargado juegos creados con Unity a casi 2 mil millones de dispositivos móviles.

El 34% de los 1000 principales juegos para dispositivos móviles gratuitos han sido creados con Unity.

Algunos de los clientes de Unity son Electronic Arts, LEGO, Microsoft, Square Enix y Ubisoft. Tanto grandes estudios como pequeños desarrolladores se están pasando a Unity en un número cada vez mayor.

- Gran número de jugadores

Unity tiene contacto con 770 millones de jugadores a lo largo del globo a través de juegos creados con su motor.

- Mercado en crecimiento

El mercado de los juegos móviles crece cada vez con mayor rapidez. En este mercado Unity sigue dominando y ampliando su cuota.

Unity también se encuentra en primera línea en el creciente mercado de la realidad virtual. De los juegos de lanzamiento de Samsung Gear VR y de Oculus Rift, el 90% y el 53% respectivamente eran juegos creados con Unity.

En términos de ingresos, el mercado de los videojuegos está superando abiertamente a la industria del cine y la televisión.

- Herramientas del editor [$UNI_b$]

Unity pone a disposición del desarrollador una infinidad de herramientas de forma totalmente gratuita. Si bien posee contenidos más avanzados de pago, cualquiera puede descargar la versión personal de Unity para crear sus propios proyectos.

Otros de los aspectos que cabría destacar de Unity es su optimización de contenido, así como su potencial en imagen y sonido y su facilidad para realizar iteraciones rápidas mediante el modo *"Play",* con el que podremos modificar aspectos del juego al mismo tiempo que lo probamos, pudiendo incluso realizar una depuración frame por frame.

- Multiplataforma [$UNI_c$]

Unity tiene en su poder el soporte multiplataforma líder de la industria, y gracias al motor Unity basta con compilar una sola vez el producto para poder desplegarlo de forma sencilla a todas las plataformas de las que dispone.

- Dispositivos móviles

Unity es el motor de juegos favorito del mundo para la creación de juegos para dispositivos móviles, ello debido en gran medida a factores como su sencillez de despliegue a Android, iOS, Windows Phone y Tizen, así como a las optimizaciones que aporta el motor gracias a sus prestaciones y a los servicios de monetización y retención de clase mundial para juegos móviles.

- Realidad Virtual y Realidad Aumentada

Unity dispone de un pipeline de renderizado altamente optimizado para ayudar a conseguir frame rates excepcionales, lo cual unido a la capacidad de iteración rápida del editor han hecho que Unity sea la plataforma de desarrollo de realidad virtual más utilizada del mercado.

- Escritorio

Gracias a los shaders físicos de Unity, así como a Realtime Global Illumination de Enlighten, la creación de juegos bellos e inmersivos para escritorio

- Consolas

Unity ofrece grandes facilidades para los desarrolladores independientes, permitiéndoles publicar en plataformas de consola como PS4, Xbox One, WiiU y muchas otras sin costo.

- Web

Dependiendo del uso que se le desee dar, la opción de compilación WebGL altamente optimizada que ofrece Unity puede hacer que el rendimiento tenga velocidades nativas. Ya utilizado en una serie de títulos comerciales exitosos.

- Smart TV

Unity también soporta el despliegue en tvOS, Android TV y Samsung Smart TV. Con solo hacer unos pocos ajustes a las entradas, un juego de móvil puede ser llevado con relativa sencillez a estas plataformas de rápido crecimiento.

- En crecimiento [$UNI_d$]

Unity sigue mejorando cada vez más, y en su web posee un completo Roadmap con las características en las que están trabajando actualmente para futuras versiones, así como el estado de las mismas.

- Aprendizaje [$UNI_e$]

Unity tiene a disposición de todos una enorme base de datos con todo lo necesario para aprender a utilizar el editor, así como tutoriales de distintos niveles de conocimiento para poner en práctica lo aprendido y multitud de aportes de usuarios para facilitar aún más si cabe la labor de aprendizaje.

#### 2.4.1.2. Unreal Engine 4

- Al alcance de todos [$UNR_a$]

Hasta hace relativamente poco, el motor Unreal Engine, uno de los *Game Engines* más potentes del mercado, era de pago. Sin embargo, realizaron un cambio en su política y lo convirtieron en un motor gratuito para todos a cambio de llevarse un 5% de las ganancias obtenidas mediante juegos y aplicaciones creados con el motor.

- Contenido completo

Unreal Engine 4 no tiene diferentes versiones del mismo, de modo que no nos encontramos con características únicas de versiones de pago mejoradas, sino que todo el mundo que descargue Unreal Engine 4 tiene acceso a todo el contenido del mismo. Unreal Engine 4 te lo da todo para que puedas crear cualquier cosa que se te ocurra.

- Mercado móvil

Unreal Engine 4 está diseñado para móviles, ahora y en el futuro. Con Unreal Engine puedes crear desde simples juegos 2D a impresionantes gráficos de último nivel, Unreal Engine 4 te da la capacidad de desarrollar tu juego y desplegarlo a dispositivos iOS y Android.

- Proyectos

El scripting visual de Unreal Engine 4 permite crear prototipos y construir juegos completos, simulaciones y similares sin necesidad de programar. Las herramientas para ello, así como para debugear los proyectos vienen incluidas en Unreal Engine 4.

- Herramientas

Unreal Engine 4 tiene integradas multitud de herramientas para permitirte construir cualquier aspecto de tu proyecto. Características avanzadas incluyen opciones de renderizado adicionales, interfaces de usuario, animaciones, efectos visuales, físicas, etc…

- Código fuente

Todo desarrollador que usa Unreal Engine 4 tiene acceso al motor C++ al completo y al editor de código fuente. Poseer todo el código fuente otorga al usuario la capacidad de personalizar su juego, así como hacer más sencillo el proceso de debugeo y lanzamiento. Cualquiera puede unirse a la comunidad de Unreal Engine y extender las más de tres millones de líneas de código disponibles en GitHub.

- Realidad virtual [$UNR_b$]

La fidelidad visual se une a las grandes capacidades del motor para crear experiencias inmersivas de realidad virtual en Unreal Engine 4. Las capacidades del motor permiten un frame rate de 90 Hz estéreo con altas resoluciones sin necesidad de cambios en el código, mientras que al poseer herramientas escalables se pueden crear todo tipo de escenas, desde simples a complejos entornos cinemáticos. Todo ello con una velocidad de iteración que facilita el proceso creativo.

- Mercado [$UNR_e$]

Unreal Engine 4 posee un mercado propio en el que la gente puede comprar y vender sus creaciones. Así, se puede acelerar el desarrollo haciéndose con ejemplos sencillos de juegos, arte y audio, código C++ y más cosas.

- Apto para todos

Unreal Engine 4 es una herramienta usado tanto por estudiantes como por desarrolladores independientes, así como por grandes estudios profesionales. Unreal Engine

4 es una herramienta que te permite hacer todo tipo de cosas, así que a pesar de que progreses podrás seguir utilizándola

- Potencial gráfico

Unreal Engine 4 es un motor que permite empujar los gráficos a su límite máximo en PC, consolas y realidad virtual mediante el uso de iluminación personalizada, efectos visuales y elementos cinemáticos. Permite crear imágenes asombrosas para visualizaciones de arquitectura, simulaciones, películas digitales y mucho más.

- En crecimiento [$UNR_f$]

Unreal Engine 4 sigue implementando nuevas características y posee un completo Roadmap con las características en las que están trabajando actualmente para futuras versiones, así como el estado de las mismas. Todo esto ordenado por secciones tales como audio, gameplay, plataformas, etc.

- Aprendizaje [$UNR_c$] [$UNR_d$]

Unreal Engine 4 cuenta con una inmensa cantidad de información para el aprendizaje, ya sea en forma de extensa documentación acerca de las diversas herramientas disponibles, así como video tutoriales para facilitar el aprendizaje de ciertas áreas.

#### 2.4.1.3. Elección de *Game Engine*

Tras ver que ambos motores ofrecían una gran cantidad de opciones, se decidió descargar ambos para realizar pruebas de su funcionamiento en el equipo sobre el que se iba a realizar el trabajo como medida final para decidir con cuál de los dos *Game Engines* se realizaría el proyecto.

En primer lugar se descargó y probó Unity, se realizaron con él unas pruebas sencillas y se comprobó que funcionaba correctamente en el equipo.

Una vez finalizadas las pruebas de Unity, se descargó Unreal Engine 4 para realizar sus pruebas pertinentes. Si bien al principio parecía que funcionaba correctamente, se pudo ver que la carga inicial del programa fue muy lenta, hecho que no auguraba buenos resultados.

Efectivamente, tras finalizar la carga inicial y poder comenzar con las pruebas quedó claro que el equipo en el que se iba a trabajar no tenía la potencia necesaria para utilizar Unreal Engine 4 con comodidad. El motor funcionaba con lentitud, se atascaba constantemente, no recibía algunos comandos, etc.

Al comprobar que no se podía trabajar con comodidad en Unreal Engine 4 debido a las características técnicas del equipo en el que se iba a trabajar, se decidió que se usaría el *Game Engine* Unity para el proyecto.

### 2.4.2. Herramientas de creación de gráficos

Para la creación de los gráficos que se vayan a utilizar en el proyecto se usará la herramienta Photoshop.

La selección de esta herramienta se debe en mayor medida a que ya se disponía de la misma y se poseían unos conocimientos mínimos de su funcionamiento, eliminando así la necesidad de tener una fase de aprendizaje de la misma. A pesar de que no se posee una gran habilidad utilizando la herramienta, el estilo artístico escogido facilita el trabajo a la hora de crear los elementos necesarios.

## 2.5. Aprendizaje

En este apartado se abarcarán los pasos realizados para llevar a cabo el aprendizaje de las herramientas necesarias para el proyecto.

### 2.5.1. Aprendizaje de Unity

Para llevar a cabo el aprendizaje básico de Unity se realizaron cuatro tareas de aprendizaje principales, para lo cual se utilizaron los recursos proporcionados por Unity en su web ya que nos parecían adecuados para lo que buscábamos.

En primer lugar se siguieron dos tutoriales básicos:

1) "Roll-a-ball Tutorial", tutorial para aprender los aspectos básicos de Unity.
2) "2D UFO Tutorial", tutorial básico para creación de juegos en 2D.

Tras finalizar estos dos tutoriales, se decidió realizar el tutorial llamado "Tanks Tutorial", en el que se detallaban aspectos más avanzados del manejo de Unity.

Para terminar de completar el aprendizaje básico, se visualizaron vídeos oficiales de Unity acerca de la creación de juegos en 2D para aprender conceptos específicos de los mismos. [$UNI_i$]

#### 2.5.1.1. Roll-a-ball Tutorial

Roll-a-ball es un tutorial para crear un juego simple. Consiste en una esfera que gira por una superficie recogiendo unos coleccionables. Con este tutorial se aprenden algunos de los principios básicos de Unity, como por ejemplo los *game objects*, los componentes, los *prefabs*, las físicas del juego y el *scripting*. [$UNI_f$]

Resumen de los pasos seguidos en el tutorial:

- Aprender a configurar el entorno de trabajo y cómo preparar el comienzo del juego creando los elementos que serán el jugador y el mapa por el que nos moveremos.
- Recibir entradas desde el teclado y tratarlas para convertirlas en acciones dentro del juego para así poder mover al jugador.

- Mover la cámara de forma relativa al jugador, aprendiendo a evitar que la cámara se mueva de forma indebida y logrando que se adapte a nuestras necesidades.
- Terminar de crear el área de juego añadiendo muros contra los que impactará nuestro jugador para evitar así que caiga al vacío al salirse de los límites.
- Crear los objetos coleccionables del juego y animarlos de manera que resulten atractivos para el jugador. También se aprenden a crear *prefabs*, modelos de objetos guardados para poder usarlos en el futuro.
- Modificamos los coleccionables que habíamos creado para así poder recogerlos con nuestro personaje.
- Se aprende a llevar la cuenta de la puntuación y a mostrarla en una interfaz de usuario.
- Aprendemos a montar un juego ya terminado para la plataforma que deseemos.

#### 2.5.1.2. 2D UFO Tutorial

2D UFO es un tutorial para crear un juego simple en 2D, muy similar al anterior salvo que aquí se muestran ciertos aspectos únicos del modo 2D y se realizan acciones diferentes al anterior, como descargar *assets* del *asset store*. [$UNI_g$]

Resumen de los pasos seguidos en el tutorial:

- Aprender a descargar del *asset store* los *assets* necesarios para llevar a cabo el tutorial, así como aprender a configurar el entorno de trabajo para juegos en 2D.
- Añadimos *sprites* del *asset* descargado al entorno de trabajo para utilizarlos en el juego. Preparamos *sprites* del jugador y el campo de juego.
- Asignamos al jugador un *rigidbody* para habilitar el movimiento del mismo y que reaccione a fuerzas físicas. Creamos el script encargado del movimiento del jugador.
- Se aprende el funcionamiento de los *colliders 2D* para detectar colisiones con el entorno.
- Configuramos la cámara para que siga al jugador alrededor del campo de juego mediante un script.
- Creamos los coleccionables a partir de un *prefab* y los hacemos rotar mediante un *script*.
- Hacemos que los objetos coleccionables sean recogidos mediante el jugador modificando el *script* de control del jugador.
- Creamos la interfaz de usuario y modificamos el *script* del jugador para que se cuenten los coleccionables y se avise del final del juego una vez recogidos todos.
- Montamos el juego.

#### 2.5.1.3. Tanks Tutorial

Tutorial avanzado llevado a cabo durante el Unite Boston 2015. En este tutorial se enseña a crear un juego *shooter* en 3D para dos jugadores. Se aprenderán mecánicas de

juego, a crear interfaces de usuario adaptables, arquitectura de juegos y a realizar mix de audio. [$UNI_h$]

Resumen de los pasos seguidos en el tutorial:

- En primer lugar se enseña a descargar los *assets* necesarios para llevar a cabo el proyecto, para a continuación preparar la zona de juego usando los mismos.
- Aprendemos a crear y controlar el tanque que usaremos como jugador. Para ello usamos como base el modelo de tanque que se proporciona, para a continuación ir añadiéndole los elementos necesarios para su funcionamiento, tales como un *rigidbody*, *colliders* y el resto de elementos necesarios. También se crea un *script* para controlar el tanque mediante el teclado, así como sus efectos sonoros y el resto de atributos.
- Se crea la cámara del juego, la cual mantendrá a los dos tanques en pantalla en todo momento.
- Se genera el atributo de vida de los tanques, así como la interfaz de usuario que la representa y los cambios que sufrirá al ser impactada por los proyectiles en función de la zona de impacto.
- Creamos los proyectiles que lanzarán los tanques, así como su radio de explosión para el cálculo de daño.
- Mediante *scripts* creamos el lanzamiento de los proyectiles desde los tanques. Además, creamos una interfaz de usuario que facilita su lanzamiento y los efectos de sonido que le acompañarán.
- Creamos un *Game Manager* que se encarga de controlar el juego. Se encargará de las labores de creación de las rondas del juego y decidirá el ganador del juego. También se aprende acerca de las *coroutines* y cómo se usan en escenarios como este.
- Aprendemos a manejar los efectos de audio del juego, creando un mix de audio en el que los efectos sonoros reducen el sonido de la música de fondo.

## 2.6. Análisis de Riesgos

Para este proyecto existen tres riesgos principales que destacan sobre todos los demás: el factor del tiempo, el peligro de la pérdida de datos y la creación de la memoria del proyecto.

### 2.6.1. Tiempo

Debido a circunstancias personales el proyecto de fin de carrera tuvo que retrasar su comienzo hasta mayo, dejando disponible únicamente la convocatoria de septiembre para la defensa del mismo.

### 2.6.2. Pérdida de Datos

El equipo en el que se va a realizar el proyecto es doméstico, y en la zona en la que está instalado ocasionalmente se pierde la red eléctrica debido a diversos factores, apagándose el equipo y perdiendo todo el proceso no guardado.

La red de internet de la zona tampoco es muy estable y ha habido problemas con la misma, pudiendo causar estos mismos pérdidas de información al transferir archivos.

### 2.6.3. Memoria

Anteriormente nunca se ha realizado un documento de tal magnitud, de modo que es difícil estimar lo que puede ocurrir con el mismo. El riesgo principal pasa por no poder estimar el tiempo que llevará crear la misma.

## 2.7. Análisis de Factibilidad

El riesgo del tiempo no es tan grave como aparenta ser, si bien únicamente se poseen cuatro meses para realizar el proyecto, estos meses son meses de verano. Durante esta época no se tienen las restricciones horarias que se tienen durante el resto del año debido a que las actividades extraescolares que se suelen realizar no tienen lugar del mes de julio en adelante, teniendo así a disposición mucho más tiempo del que a simple vista aparentaba poseerse.

El problema de la pérdida de datos es previsible. Si se llevan a cabo a menudo guardados de copias del proyecto en la nube, en caso de sufrir una pérdida el impacto de la misma se vería mitigado. Si bien también existen problemas de pérdidas de datos debido a problemas con la red de internet, estos son menos graves que los anteriores, ya que siempre se pueden guardar los archivos en un sistema de almacenamiento portátil para subirlos a la nube desde otro terminal, así como para descargarlos en caso de que fuese necesario.

El riesgo de la memoria es el mayor de los riesgos, para evitarlo se observarán memorias de otros proyectos anteriores disponibles para su estudio, así como consultas al director del proyecto para asegurarse de que se va por buen camino con la misma.

En resumen, si bien el proyecto tiene sus riesgos, el alcance del mismo hace que estos riesgos se reduzcan en cierta medida, haciendo así que el proyecto sea factible.

# Capítulo 3: Gestión del Proyecto

En este capítulo se tratarán las primeras fases del proyecto tras finalizar el aprendizaje. Se mostrarán las tres etapas que se realizaron antes del producto final, el objetivo de las cuales era ir formando diferentes aspectos del que sería el producto terminado.

Las tres fases fueron el prototipo inicial, la versión *Alpha* y la versión Beta.

Dentro de cada fase hay tres secciones principales:

Diseño de nivel, donde se detallan los aspectos del nivel creado.

Gráficos, donde se habla acerca de los gráficos utilizados en la creación del nivel.

Comentarios técnicos, donde se detallan aspectos técnicos del nivel y sus elementos, así como los *scripts* que regulan el funcionamiento de los elementos del juego.

## 3.1. Prototipo inicial

Prototipo inicial creado como prueba para los conocimientos adquiridos. Se pretende comprobar si se es capaz de crear un personaje que se mueva por un nivel.

### 3.1.1. Diseño de Nivel

Se pretende crear un nivel sencillo, que consiste en un par de plataformas suspendidas en el aire sobre las que el jugador se mantendrá para así poder probar los controles de movimiento básicos del jugador. (Ver figura 3.1)

**1: 3.1 Nivel prototipo pruebas**

## 1.4 Gráficos

Se pretenden crear gráficos simples para este prototipo. Tanto el nivel como el jugador estarán formados por figuras geométricas sencillas. Las plataformas que forman el nivel

serán figuras rectangulares blancas, mientras que el jugador será un triángulo rojo. (Ver figura 3.2)

2: 3.2 Nivel prototipo pruebas + jugador

### 3.1.2. Comentarios Técnicos

Tanto suelo como jugador tendrán *colliders* para interactuar entre sí. Los *colliders* son unos elementos del motor Unity que se utilizan para trabajar la interacción de distintos elementos entre sí. En nuestro caso la función de los *colliders* es hacer que el jugador pueda mantenerse sobre las plataformas del nivel.

El jugador contará con un *script* para el control del mismo (Ver script 3.1). Mediante este *script* básico recibimos el movimiento horizontal del personaje, y mediante el uso de la barra espaciadora podemos saltar cada vez que lo presionamos aunque nos encontremos en el aire.

La cámara se mantiene fija en la posición inicial, el movimiento de la misma se implementará en futuras versiones.

*Script 3.1: Prototipo script jugador*

```
public class PlayerControllerScript : MonoBehaviour {

    public float maxSpeed = 7f;

    public float jumpForce = 350f;


    void Start () {    }

    void FixedUpdate ()
{


    float move = Input.GetAxis("Horizontal");
```

```
    GetComponent<Rigidbody2D>().velocity = new Vector2(move *
maxSpeed, GetComponent<Rigidbody2D>().velocity.y);
}

    void Update()
   {
            if (Input.GetKeyDown(KeyCode.Space))
            {
                GetComponent<Rigidbody2D>().AddForce(new Vector2(0,
jumpForce));
            }

    }
```

## 3.2. Versión *Alpha*

La versión *Alpha* extiende el prototipo inicial. Esta vez hemos usado imágenes para el personaje, y le hemos creado al mismo tiempo un control mejorado. Se pretende probar que se pueden realizar animaciones para el personaje y que el control del mismo es mejor que el de la versión anterior.

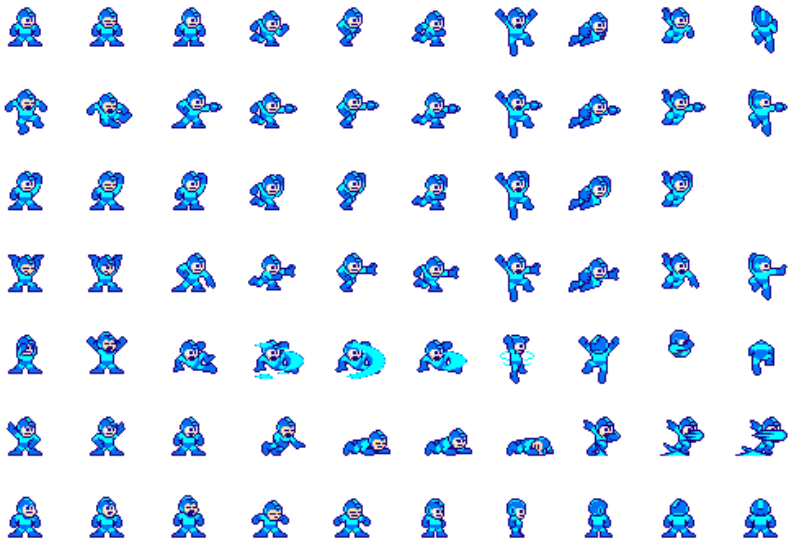
### 3.2.1. Diseño de Nivel

El nivel de la versión *Alpha* es el mismo del prototipo inicial, ya que en la versión *Alpha* los cambios realizados se han centrado en el personaje controlado por el jugador. El nivel sigue constando de dos plataformas horizontales suspendidas en el aire sobre las que el jugador se mantiene en pie.

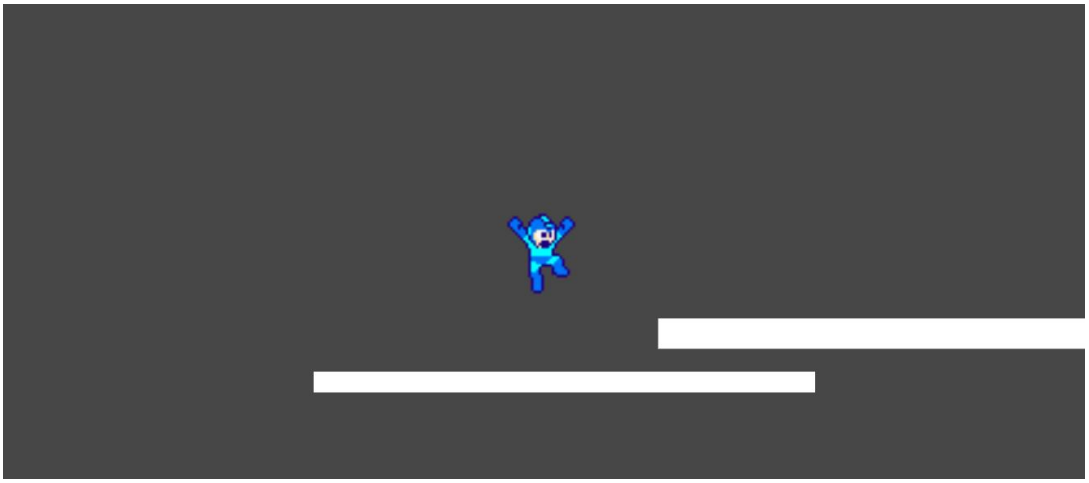
### 3.2.2. Gráficos

En esta versión los gráficos del jugador se han creado mediante una imagen de un personaje hallada en internet. Mediante esta imagen se han creado las animaciones que conforman el personaje. (Ver figura 3.3)

No todos los elementos de la imagen se han utilizado, solo se han cogido los necesarios para crear las animaciones de reposo, correr y saltar. ( Ver figura 3.4, captura de animación de salto)



3: 3.3 *Sprites* jugador versión *Alpha*

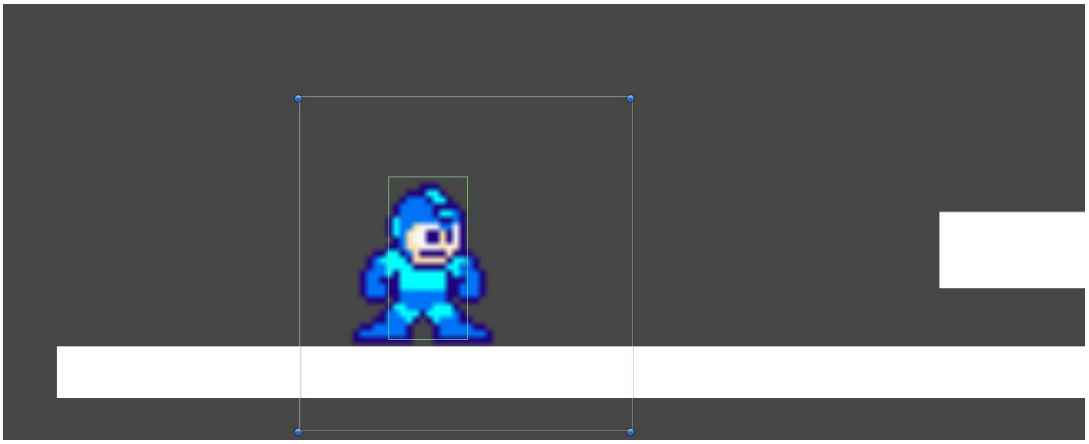


4: 3.4 Animación de salto versión *Alpha*

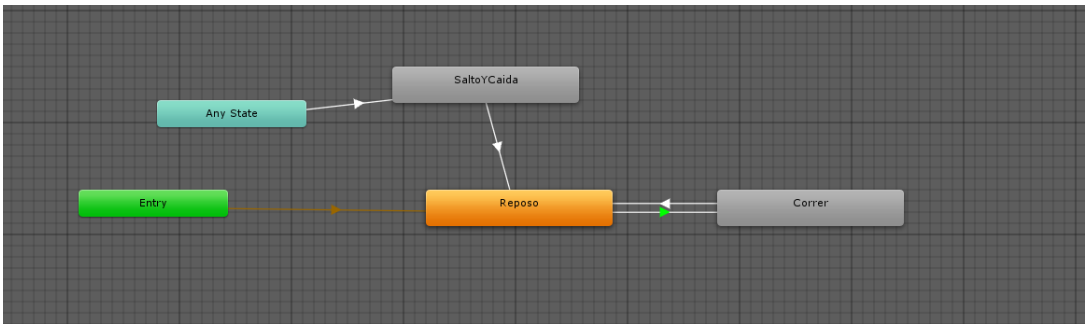
### 3.2.3. Comentarios Técnicos

La forma del personaje controlado por ha cambiado en esta nueva versión, ahora en lugar de tratarse de un triángulo tiene una forma humanoide, de modo que se ha adaptado la forma del *collider* para que se adapte mejor a la nueva figura. (Ver figura 3.5)

Ahora que el personaje cuenta con animaciones, también necesita un controlador para estas. Este controlador se encarga de regular las animaciones del jugador, haciendo que pase de una a otra en función de lo que a nosotros nos convenga. El mapa de transiciones del animador muestra las animaciones disponibles y las transiciones entre ellas. (Ver figura 3.6)



5: *3.5 Collider* del jugador




6: 3.6 Controlador de animaciones de jugador (*Alpha*)

El *script* de control del jugador también ha sido modificado para que transmita al animador los datos pertinentes para que cambie las animaciones según el momento. (Ver script 3.2)

Ahora comprueba si el jugador se encuentra en tierra o en el aire para decidir si puede saltar o no. Si se encuentra en tierra, el jugador puede saltar y se le aplicarán las fuerzas pertinentes para que realice el salto y al mismo tiempo se avisará al animador para que reproduzca la animación de salto hasta que el jugador toque tierra de nuevo.

Por otra parte el *script* maneja la velocidad a la que se mueve el jugador, y ahora le pasa al animador la información acerca de ello para que cambie entre las animaciones de reposo y correr.

*Script 3.2: Alpha script jugador*

```
public class PlayerControllerScript : MonoBehaviour {

    public float maxSpeed = 7f;
    bool facingRight = true;

    Animator anim;

    bool grounded = false;
```

```
    public Transform groundCheck;

    private bool canControl = true;
    float groundRadius = 0.002f;
    public LayerMask whatIsGround;
    public float jumpForce = 350f;



    void Start () {

        anim = GetComponent<Animator>();
    }

      void FixedUpdate () {
            grounded = Physics2D.OverlapCircle(groundCheck.position,
groundRadius, whatIsGround);
            anim.SetBool("Ground", grounded);

            anim.SetFloat("vSpeed",
GetComponent<Rigidbody2D>().velocity.y);



            float move = Input.GetAxis("Horizontal");

            anim.SetFloat("Velocidad", Mathf.Abs(move));
            GetComponent<Rigidbody2D>().velocity = new Vector2(move
* maxSpeed, GetComponent<Rigidbody2D>().velocity.y);

            if (move > 0 && !facingRight)
            { Flip(); }
            else if (move < 0 && facingRight)
            { Flip(); }


      }

    void Update()
    {

            if (grounded && Input.GetKeyDown(KeyCode.Space))
            {
                anim.SetBool("Ground", false);
                GetComponent<Rigidbody2D>().AddForce(new Vector2(0,
jumpForce));

            }

        }

void Flip()
    {
        facingRight = !facingRight;
        Vector3 theScale = transform.localScale;
        theScale.x *= -1;
        transform.localScale = theScale;
    }
}
```

## 3.3. Versión Beta

En esta versión se planea crear un nivel con diversos elementos para probar así el funcionamiento en la versión final. Se pretende probar que se pueden crear niveles funcionales con las herramientas disponibles.

### 3.3.1. Diseño de Nivel

En esta ocasión se ha creado un nivel a diferencia de las anteriores ocasiones en las que únicamente había un par de plataformas. En este nivel se deberán recoger siete coleccionables y a continuación alcanzar la salida para finalizar. En caso de morir en algún momento se comenzará de nuevo el nivel.

Al tratarse de un nivel tan grande, si se mostraba todo el nivel en una sola imagen la calidad de esta era insuficiente para mostrar los detalles de cada zona, de modo que se dividió la imagen por zonas. (Ver imagen 3.7 )

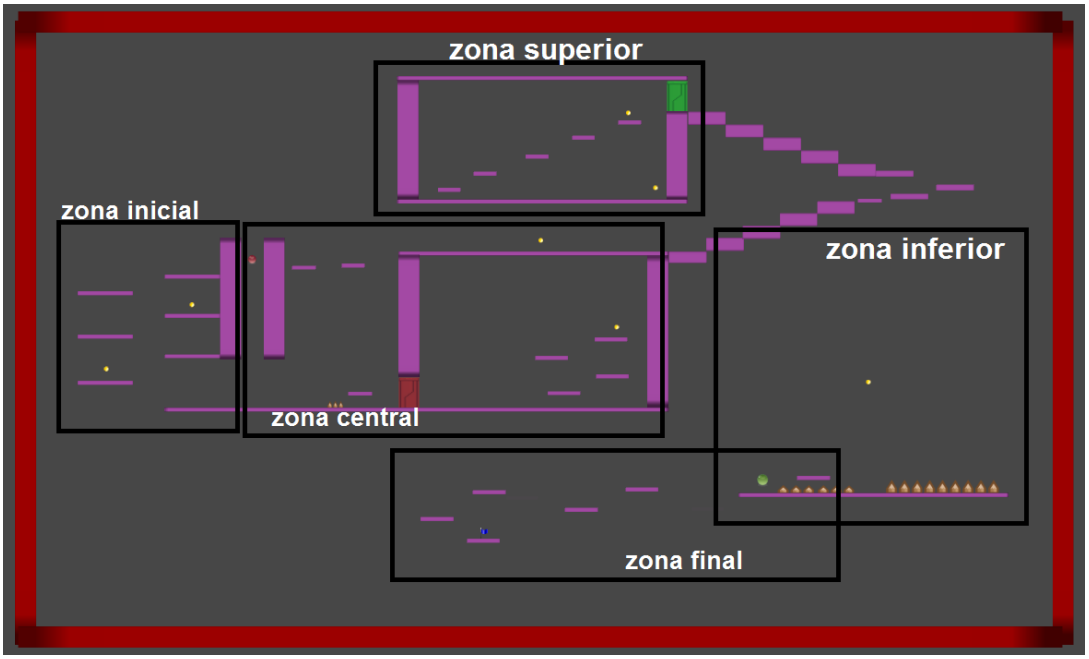
El personaje comenzará en una plataforma, desde la cual deberá descender por otras cinco hasta llegar a la parte inferior mientras recoge los coleccionables del camino. (Ver imagen 3.8)

Una vez en la parte inferior, el jugador deberá subir por unas plataformas móviles desde las cuales deberá saltar a través de una pared falsa para conseguir la llave de la puerta roja y conseguir abrirla para recoger los coleccionables del interior. La sala cerrada por la puerta roja se encuentra al lado de la plataforma móvil por la que se subió. (Ver imagen 3.9)

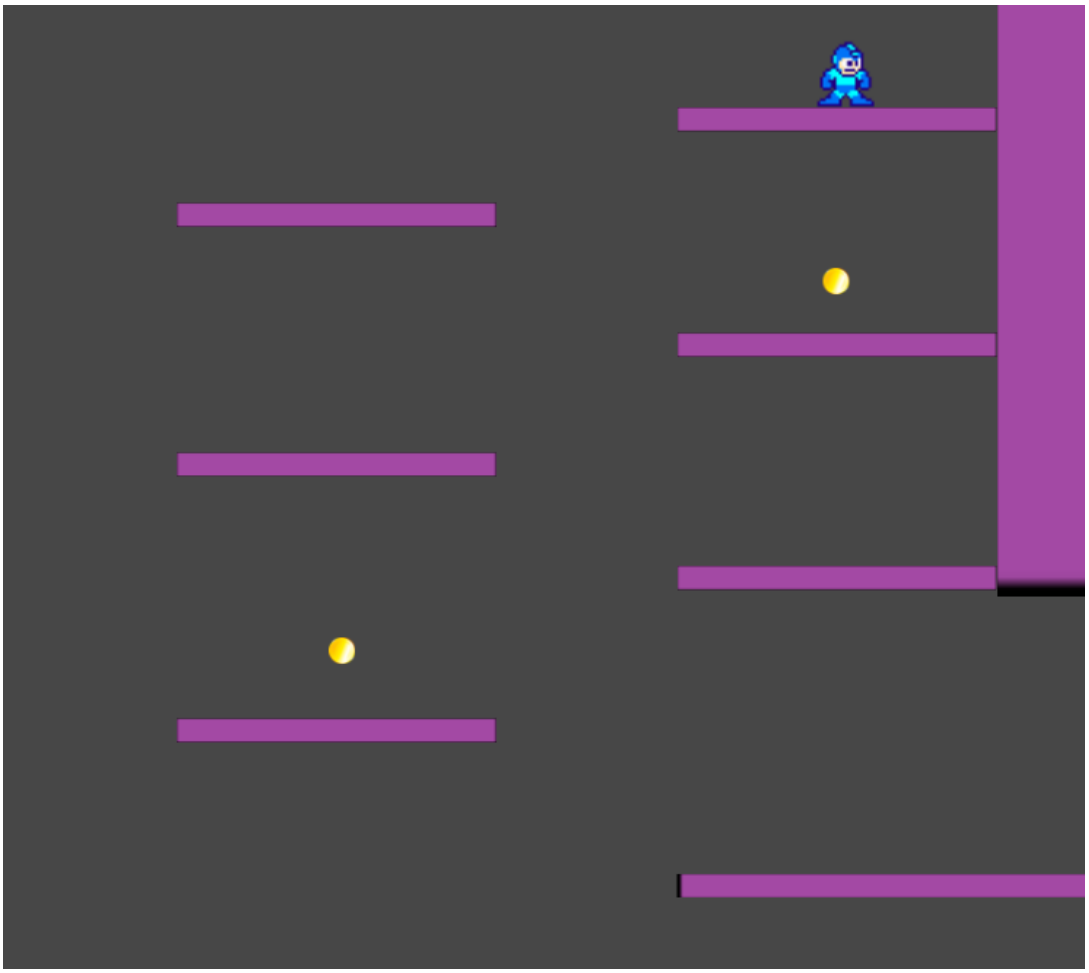
Tras pasar la sala de la puerta roja, el jugador deberá volver por la plataforma móvil por la que subió anteriormente y saltar en dirección contraria para subirse al tejado de la sala de la puerta roja para seguir avanzando hasta la siguiente plataforma móvil, por la que deberá descender para coger la llave de la puerta verde y volver a subir. (Ver imagen 3.10)

Una vez arriba, deberá seguir subiendo hasta alcanzar la sala de la puerta verde, dentro de la cual encontrará los últimos coleccionables. (Ver imagen 3.11)

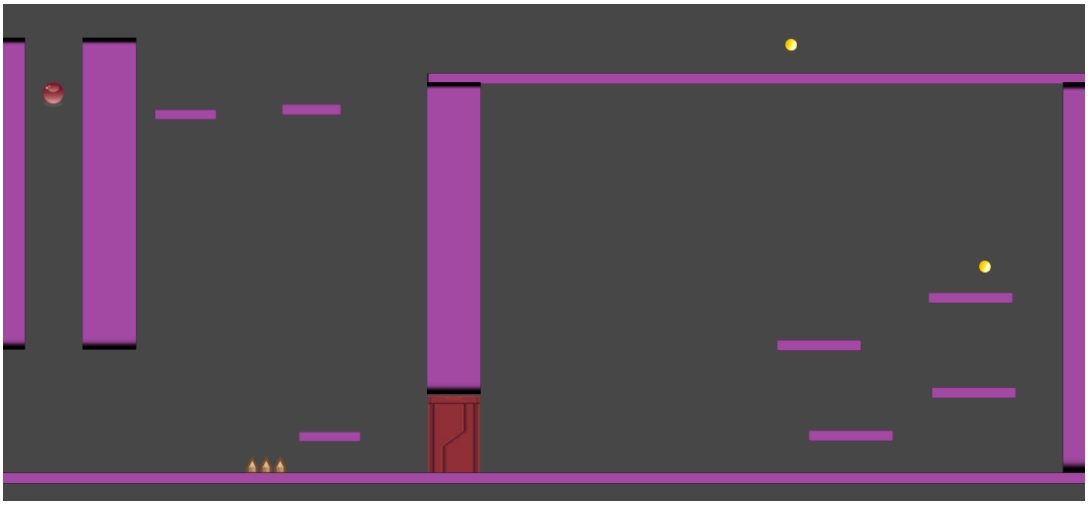
Una vez recogidos todos los coleccionables deberá volver a la zona donde encontró la llave verde para encontrar un camino compuesto por plataformas visibles e invisibles para alcanzar la meta. (Ver imagen 3.12)



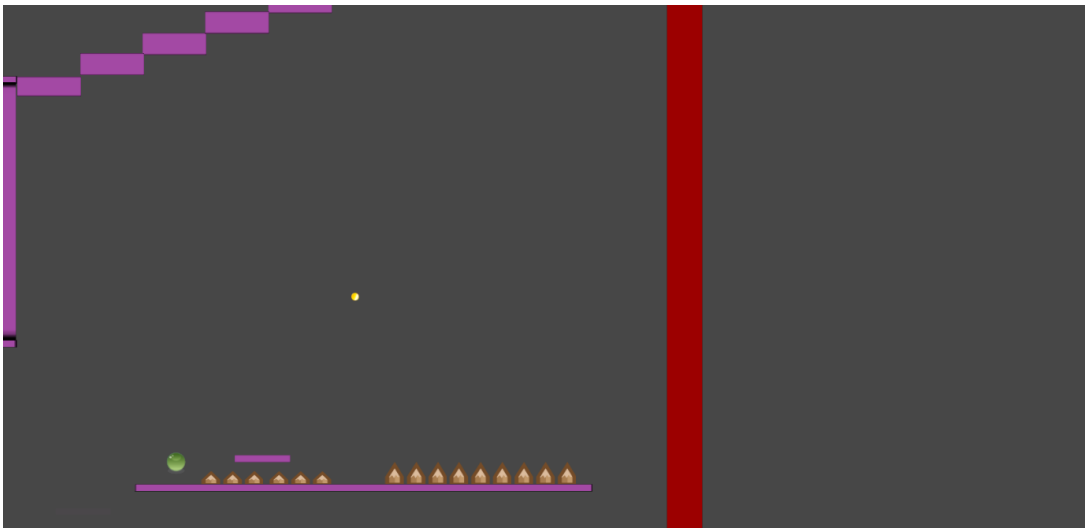

7: 3.7 Zonas de nivel Beta



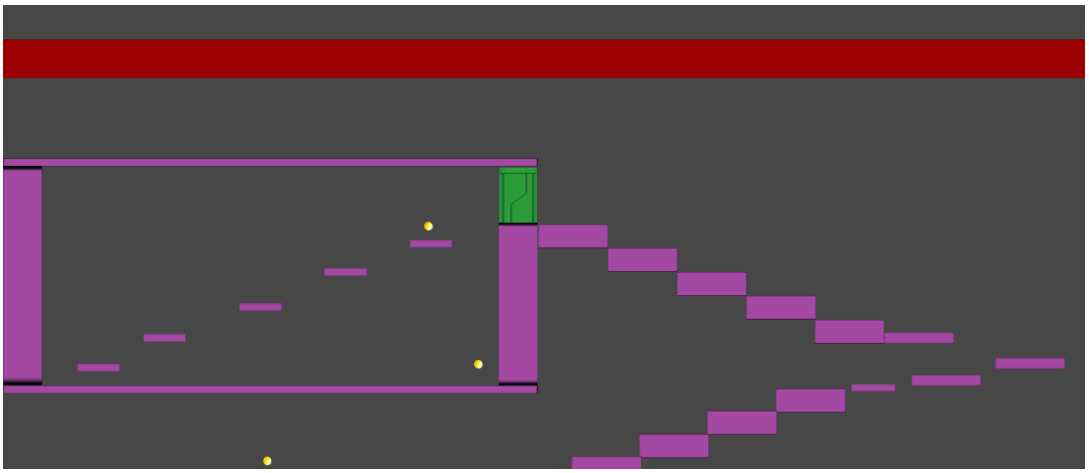
8: 3.8 Inicio nivel Beta



9: 3.9 Zona central nivel Beta



10: 3.10 Zona inferior nivel Beta



11: 3.11 Zona superior nivel Beta

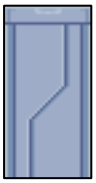
12: 3.12 Zona final nivel Beta

### 3.3.2. Gráficos

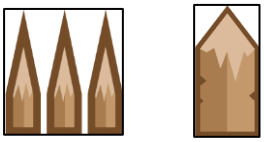
Para esta versión se han añadido a los gráficos ya existentes algunos nuevos modelos obtenidos de una página de contenido libre para su uso.

Se han añadido gráficos para:

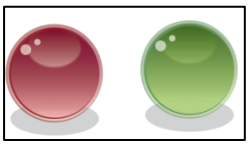
- Puertas cerradas (Ver imagen 3.13)
- Obstáculos del escenario (Ver imagen 3.14)
- Llaves (Ver imagen 3.15)
- Meta (Animación) (Ver imagen 3.16)
- Coleccionables (Animación) (Ver imagen 3.17)



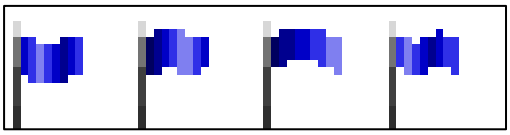
13: 3.13 Puerta versión Beta



14: 3.14 Obstáculos versión Beta



15: 3.15 Llaves versión Beta



16: 3.16 Meta versión Beta

17: 3.17 Coleccionables versión Beta

### 3.3.3. Comentarios Técnicos

Al *script* del jugador se le han añadido apartados para que interaccione con los elementos del escenario y así pueda morir, recoger llaves y coleccionables, etc. Así como elementos para controlar la interfaz de usuario. (Ver *script* 3.3)

El sistema de apertura de puertas mediante llaves también es controlado mediante un *script*, el cual se encarga de desactivar la puerta cuando el jugador recoge la llave. (Ver *script* 3.4)

El movimiento de las plataformas también es controlado mediante un *script*, el cual va moviendo la plataforma entre la posición de origen y la de destino continuamente. (Ver *script* 3.5)

Finalmente, la cámara tiene asociado un *script* para seguir al jugador en todo momento a medida que este se mueve por el escenario, a la vez que mantiene la relación que había entre ambos elementos al comenzar. (Ver *script* 3.6)

*Script 3.3: Beta script jugador*

```
public class PlayerControllerScript : MonoBehaviour {

    public float maxSpeed = 7f;
    bool facingRight = true;

    Animator anim;

    bool grounded = false;
    public Transform groundCheck;


    public Text countText;
    private bool canControl = true;
    private int count;
    public Text winText;
    public Text deathText;
    float groundRadius = 0.002f;
    public LayerMask whatIsGround;
```

```
    public float jumpForce = 350f;
    private WaitForSeconds m_EndWait;


    void Start () {

        anim = GetComponent<Animator>();
        count = 0;
        SetCountText();
        winText.text = "";
        deathText.text = "";
        m_EndWait = new WaitForSeconds(3f);
   }


      void FixedUpdate () {
        if (canControl == true)
        {
            grounded = Physics2D.OverlapCircle(groundCheck.position,
groundRadius, whatIsGround);
            anim.SetBool("Ground", grounded);

            anim.SetFloat("vSpeed",
GetComponent<Rigidbody2D>().velocity.y);


            float move = Input.GetAxis("Horizontal");

            anim.SetFloat("Velocidad", Mathf.Abs(move));
            GetComponent<Rigidbody2D>().velocity = new Vector2(move
* maxSpeed, GetComponent<Rigidbody2D>().velocity.y);

            if (move > 0 && !facingRight)
            { Flip(); }
            else if (move < 0 && facingRight)
            { Flip(); }

        }
      }

    void Update()
    {
        if (canControl == true)
        {

            if (grounded && Input.GetKeyDown(KeyCode.Space))
            {
                anim.SetBool("Ground", false);
                GetComponent<Rigidbody2D>().AddForce(new Vector2(0,
jumpForce));

            }

        }
    }

    IEnumerator OnTriggerEnter2D(Collider2D other)
    {

            if (other.gameObject.CompareTag("PickUp"))
```

```
            {
                other.gameObject.SetActive(false);
                count++;
                SetCountText();
            }

        if (other.gameObject.CompareTag("Hazard"))
            {
                anim.SetBool("muerto", true);
                deathText.text = "You Died";
                canControl = false;
                GetComponent<Rigidbody2D>().isKinematic = true;
            yield return m_EndWait;
            Application.LoadLevel(Application.loadedLevel);

            }

        if (other.gameObject.CompareTag("Finish"))
            {
            if (count == 7)
            {
                winText.text = "You WIN";
                canControl = false;
                GetComponent<Rigidbody2D>().isKinematic = true;
                yield return m_EndWait;
                Application.LoadLevel(Application.loadedLevel);
            }

        }

    }


    void SetCountText()
    {
        countText.text = "Count: " + count.ToString();

    }

    void Flip()
    {
        facingRight = !facingRight;
        Vector3 theScale = transform.localScale;
        theScale.x *= -1;
        transform.localScale = theScale;
    }
  }
```

*Script 3.4: beta script puertas*

```
public class OpenDoor : MonoBehaviour {

    public GameObject Key;

      void Start () {
```

```
      }

      void Update () {

        if(Key.active == false)
        {

this.GetComponent<Collider2D>().gameObject.SetActive(false);
        }

      }
   }
```

*Script 3.5: beta script plataformas móviles*

```
public class MovingPlatformY : MonoBehaviour {

    private int wait = 0;
    public float increase;
    public int speed = 1;
    private Vector3 origin;
    private Vector3 offset;
    private Vector3 destination;
    private bool ida = true;
    private bool subida = true;

    void Start () {
        origin = transform.position;
        destination = new Vector3(origin.x, origin.y + increase,
origin.z);
        offset = (destination - origin)/speed;
        if(destination.y < origin.y)
        {
            subida = false;
        }
      }

      void Update () {
        wait += 1;
        if (subida)
        {
            if ((wait == 1) && ida)
            {
                transform.position = transform.position + offset;
                wait = 0;
                if (transform.position.y >= destination.y)
                {
                    ida = false;
                }

            }

            if ((wait == 1) && !ida)
            {
                transform.position = transform.position - offset;
                wait = 0;
```

```
                if (transform.position.y <= origin.y)
                {
                    ida = true;
                }

            }
        }
        else
        {
            if ((wait == 1) && ida)
            {
                transform.position = transform.position + offset;
                wait = 0;
                if (transform.position.y <= destination.y)
                {
                    ida = false;
                }

            }

            if ((wait == 1) && !ida)
            {
                transform.position = transform.position - offset;
                wait = 0;
                if (transform.position.y >= origin.y)
                {
                    ida = true;
                }

            }

        }
    }
}
```

*Script 3.6: beta script cámara*

```
public class ControladorCamara : MonoBehaviour
{
    public GameObject player;

    private Vector3 offset;

    void Start()
    {
        offset = transform.position - player.transform.position;
    }

        void LateUpdate()
    {
        transform.position = player.transform.position + offset;
    }
  }
```

# Capítulo 4: Diseño

En este capítulo se habla acerca del proceso de diseño y creación del producto final. Se detallan los diferentes niveles que forman el juego, indicando en cada uno de ellos dos apartados principales:

Definición, donde se detalla el nivel en sí y las mecánicas que contiene, así como la explicación de las mismas en caso de ser nuevas y se muestra una imagen del nivel.

Información adicional, donde se comentarán aspectos adicionales del nivel, como peculiaridades de los mismos.

Tras mostrar los niveles, se procederá a listar los gráficos usados en la creación del proyecto, así como su uso.

Las animaciones del personaje se mostrarán tras los gráficos, detallando en cada una de ellas los fotogramas que la conforman, así como sus condiciones de activación.

Finalmente se detallará la interfaz de usuario que tiene el proyecto, así como los *scripts* que se encargan de controlar la cámara, el personaje controlado y el entorno de juego.

Se puede encontrar un vídeo mostrando todos los niveles y como superarlos en el siguiente enlace:

https://drive.google.com/open?id=0B5qsRzooDKxcRERILUxaWU5KMm8

## 4.1. Concepto General

Un único intento, esa es la idea principal del juego. Normalmente en los juegos de plataformas, y de cualquier género en general, el jugador cuenta con puntos de continuación, algún tipo de salud, etc. En este juego todas esas ventajas adicionales de las que se suele gozar desaparecen. Nuestro jugador tiene un único punto de salud y una única vida para superar todos los niveles.

## 4.2. Diseño de Nivel

En este apartado se recogen los niveles que conforman el juego, así como la función de estos, su mecánica propia y detalles adicionales.

### 4.2.1. Nivel 1

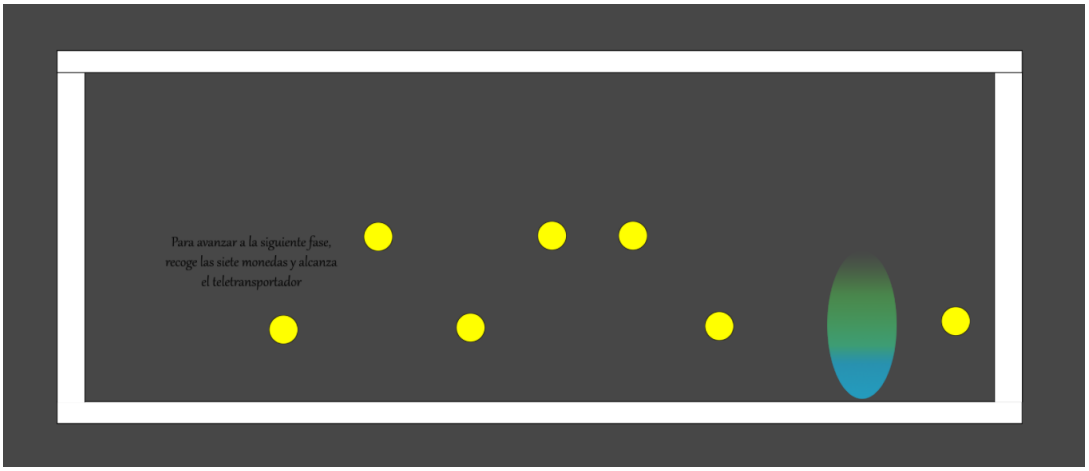
#### 4.2.1.1. Definición

En el primer nivel el jugador prueba los controles y aprende la mecánica de los coleccionables y los transportadores para pasar al nivel siguiente.

En todos los niveles se van a encontrar siete monedas con una animación propia. Cuando el jugador pasa por el sitio en el que se encuentran estas monedas, desaparecen y su contador de coleccionables aumenta en una unidad. Para poder avanzar al siguiente nivel el jugador debe recoger las siete. Esto son los coleccionables.

Los transportadores son las puertas que llevan de un nivel a otro al jugador. El jugador puede pasar por los transportadores ya que no tienen *colliders* que bloqueen el movimiento del jugador. Si el jugador tiene en su poder los siete coleccionables del nivel, al pasar por la zona del transportador se verá trasladado al comienzo del siguiente nivel con su contador de coleccionables devuelto a cero.

El nivel esta formado por una habitación rectangular en cuyo interior se hallan los siete coleccionables y el transportador. (Ver figura 4.1)




**18: 4.1 Nivel 1**

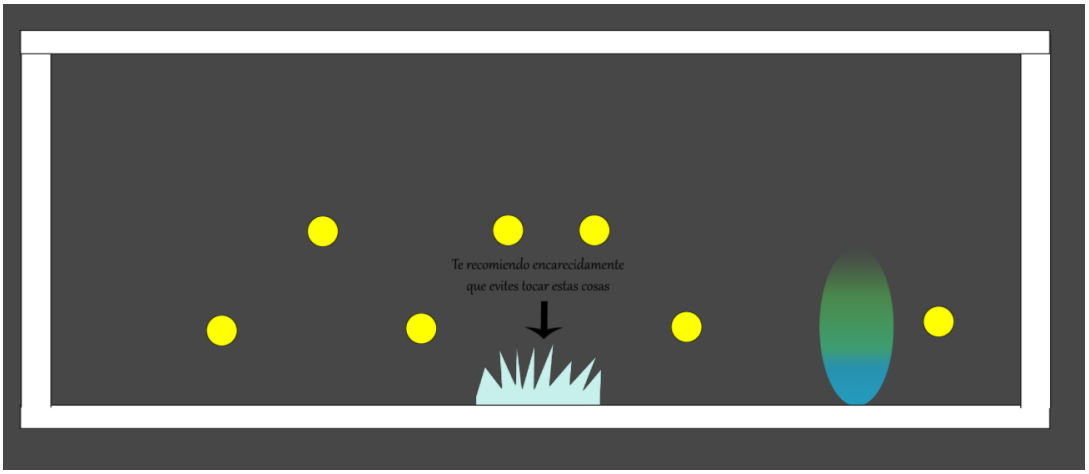
#### 4.2.1.2. Información adicional

En este primer nivel no existe ningún tipo de *hazard*, de modo que es imposible para el jugador morir en este primer nivel.

### 4.2.2. Nivel 2

#### 4.2.2.1. Definición

El segundo nivel es idéntico al primero con la excepción del *hazard* colocado en el centro. Si el jugador toca este *hazard* o cualquier otro localizado en otros niveles más adelante, se verá obligado a comenzar de nuevo desde el primer nivel. (Ver figura 4.2)




19: 4.2 Nivel 2

#### 4.2.2.2. Información adicional

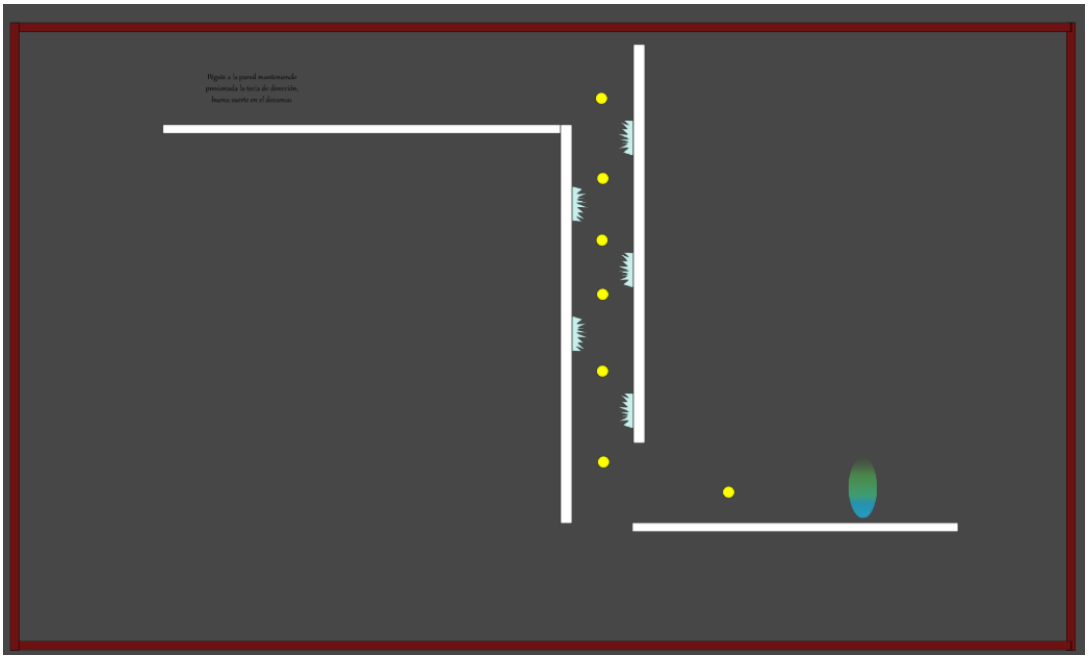
En este nivel se introducen por primera vez los *hazards* y la posibilidad de morir por parte del jugador. Si bien es un peligro sencillo de evitar, la intención es que el jugador muera por decisión propia para que descubra que deberá comenzar de nuevo si muere en niveles más avanzados.

### 4.2.3. Nivel 3

#### 4.2.3.1. Definición

El tercer nivel es el comienzo de los niveles con mecánica propia. En este nivel el jugador debe hacer uso de la habilidad para mantenerse agarrado a la pared para así poder controlar el descenso evitando los peligros del escenario a la vez que recoge los coleccionables en la bajada. (Ver figura 4.3)

Si mientras se encuentra en el aire el jugador impacta contra un muro, puede mantenerse aferrado a él si mantiene presionada la tecla de movimiento hacia el muro. El jugador se mantendrá quieto contra la pared mientras mantenga la tecla presionada, para volver a caer debe presionar en dirección contraria o soltar la tecla que estaba presionando. Se puede maniobrar durante el descenso moviendose a izquierda y derecha.



20: 4.3 Nivel 3

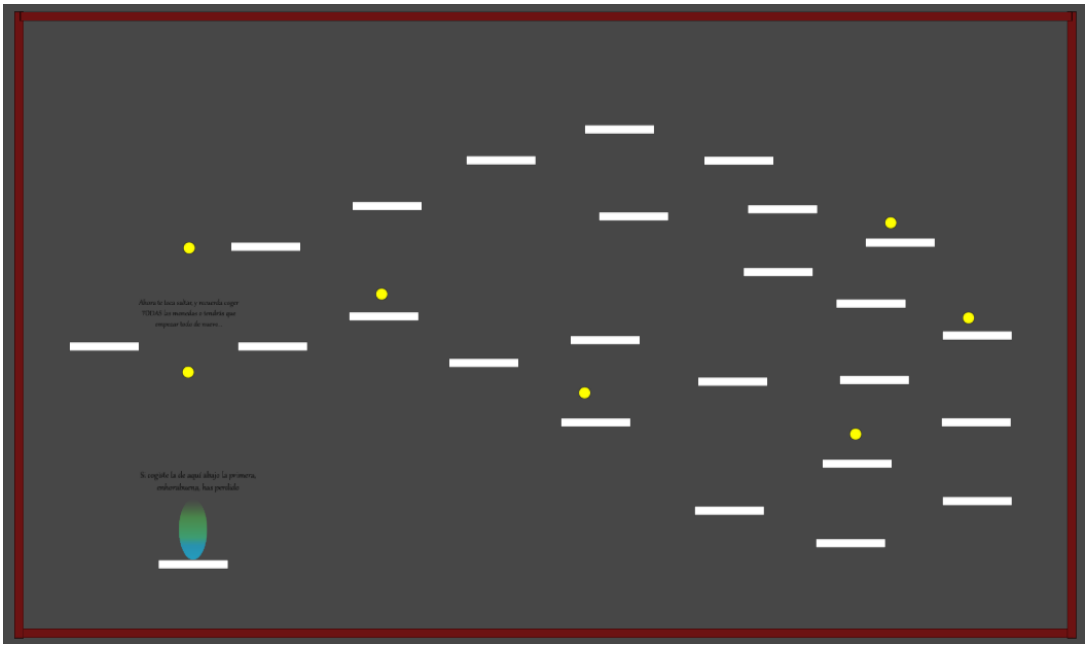
#### 4.2.3.2. Información adicional

En este nivel, aparte de los peligros del mismo también existe la posibilidad de que el jugador caiga al vacío. Para que el jugador no caiga eternamente la mayoría de niveles se encuentran rodeados por un perímetro que es un *hazard* en sí mismo para así provocar la muerte del jugador cuando caiga al vacío e impacte contra ellos.

### 4.2.4. Nivel 4

#### 4.2.4.1. Definición

El cuarto nivel está formado por plataformas suspendidas en el vacío con los coleccionables distribuidos sobre ellas. En caso de caer al vacío el jugador morirá. (Ver figura 4.4)



21: 4.4 Nivel 4

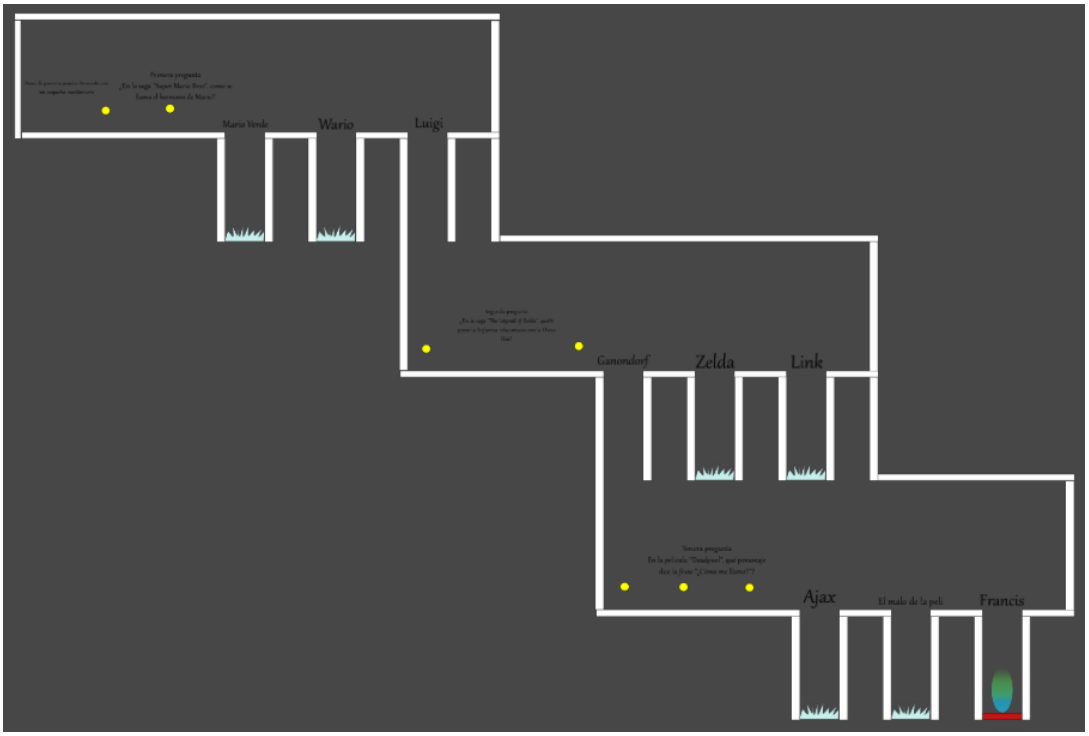
#### 4.2.4.2. Información adicional

Este nivel tiene una peculiaridad nada más comenzar, y es que el jugador puede ver un coleccionable en una posición inferior a la que se encuentra nada más comenzar el nivel. En caso de que el jugador escoja ir a por ese coleccionable nada más comenzar perderá inmediatamente, ya que se trata del último que debe recoger debido a que al cogerlo el jugador acaba en una posición desde la que no puede volver atrás, estando obligado a morir voluntariamente para poder volver a comenzar.

### 4.2.5. Nivel 5

#### 4.2.5.1. Definición

El quinto nivel se trata de un nivel cuestionario. Hay tres preguntas a responder, cada una con tres respuestas posibles que señalan una caída u otra. La respuesta correcta lleva al camino para continuar mientras que la respuesta incorrecta lleva a la muerte del jugador. Las preguntas tienen que ver con el mundo del cine y los videojuegos. (Ver figura 4.5)




22: 4.5 Nivel 5

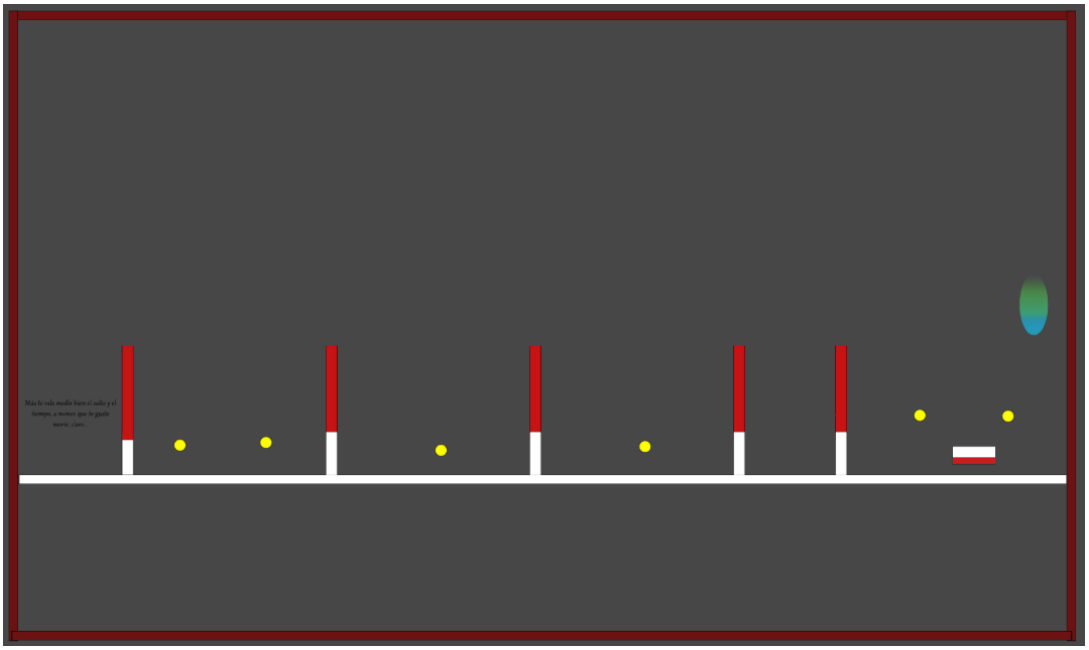
#### 4.2.5.2. Información adicional

La última de las tres preguntas está ideada para despistar al jugador, ya que las tres posibles respuestas son válidas. Sin embargo, si el jugador ha visto la película de la que habla la pregunta, sabrá discernir la respuesta correcta.

### 4.2.6. Nivel 6

#### 4.2.6.1. Definición

El sexto nivel trata acerca de buena coordinación y saber calcular bien los lapsos de tiempo. El jugador deberá sortear mediante saltos unas paredes móviles que dejan abierto el paso cada cierto tiempo. Las paredes móviles a su vez son un *hazard* que provocará la muerte del jugador si este calcula mal el tiempo del que dispone para saltar entre ellas. (Ver figura 4.6)



23: 4.6 Nivel 6

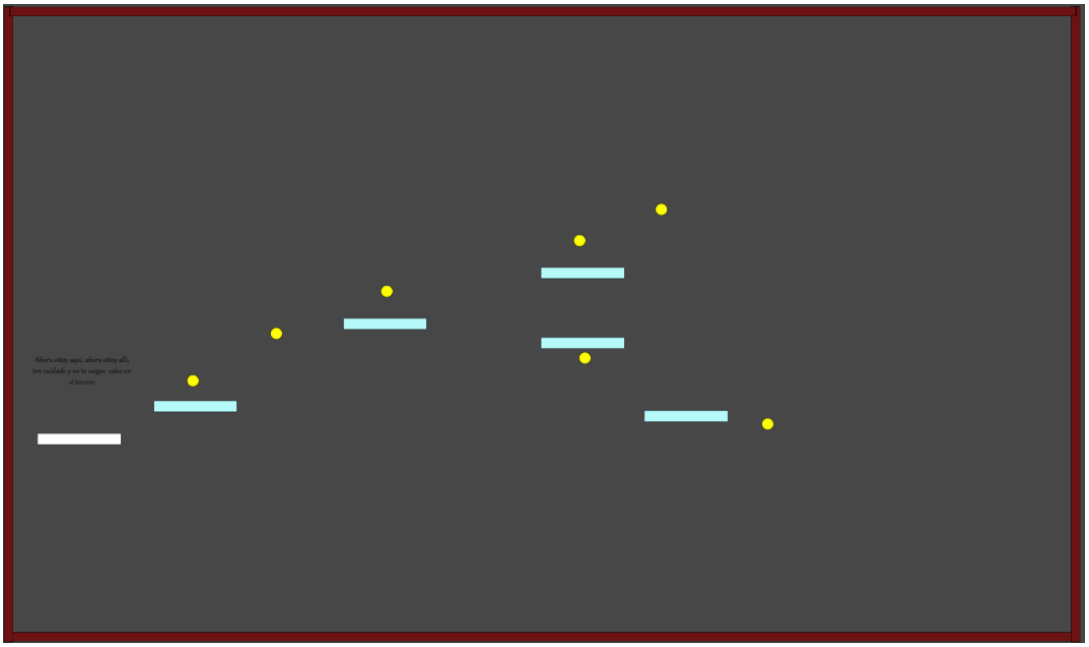
#### 4.2.6.2. Información adicional

La abertura que se crea para dejar paso al jugador está calculada para ser lo más pequeña posible. El jugador puede pasar a través de ella, pero el menor error de cálculo será mortal. Si bien en la figura 4.6 solo aparecen 6 coleccionables, el séptimo se halla tras una de las barreras móviles, haciendo que solo sea visible cuando esta se abre.

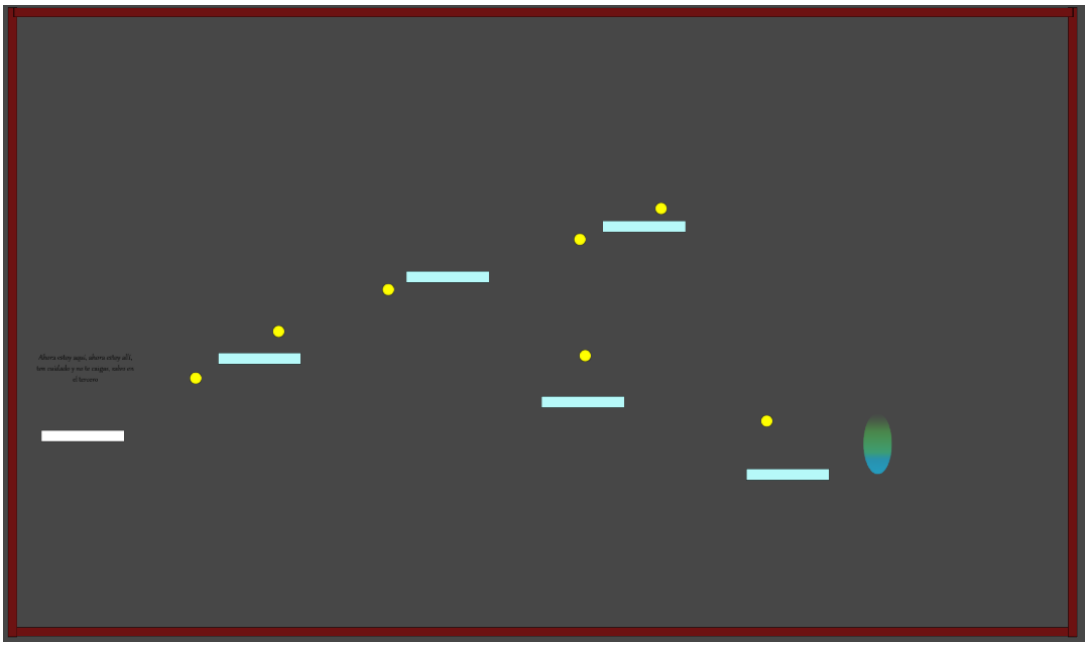
### 4.2.7. Nivel 7

#### 4.2.7.1. Definición

En el séptimo nivel el jugador debe saltar sobre plataformas que se activan y desactivan siguiendo un patrón para recoger los coleccionables. El transportador al nivel siguiente también se activa y desactiva al igual que las plataformas. (Ver figuras 4.7 y 4.8)



24: 4.7 Nivel 7 (Posición 1)



25: 4.8 Nivel 7 (Posición 2)

#### 4.2.7.2. Información adicional

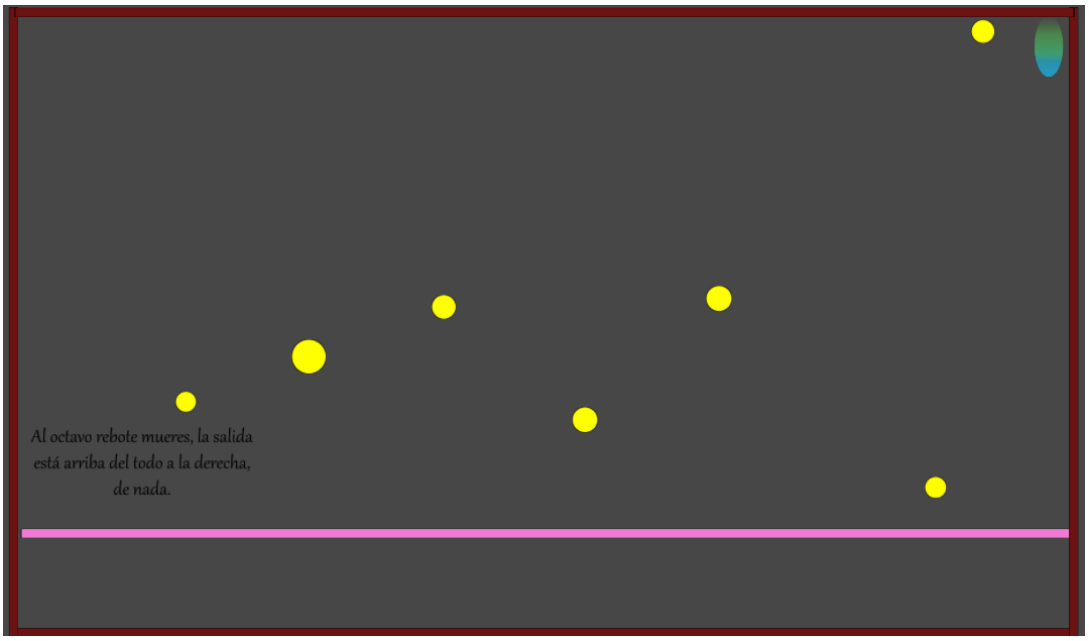
Para superar el nivel el jugador debe atravesar las plataformas sin caer al vacío, las cuales se activan y desactivan siguiendo un patrón por parejas que hace que cuando una se encuentra activa la otra se encuentre desactivada. El tiempo que hay entre activación y desactivación, si bien no es muy amplio, da el tiempo suficiente para saltar de una a otra,

aunque el jugador deberá llevar el ritmo. Para finalizar, en un punto del camino el jugador debe dejarse caer desde una plataforma para poder seguir avanzando.

### 4.2.8. Nivel 8

#### 4.2.8.1. Definición

El octavo nivel es una sala en la que el suelo fuerza al jugador a saltar constantemente, rebotando cada vez más alto, mientras que los coleccionables están dispersos por la sala y deben recogerse mientras se rebota constantemente. Los muros y el techo del nivel son mortales para el jugador para evitar que pueda agarrarse a las paredes y controlar así la fuerza del rebote. (Ver figura 4.9)




26: 4.9 Nivel 8

#### 4.2.8.2. Información adicional

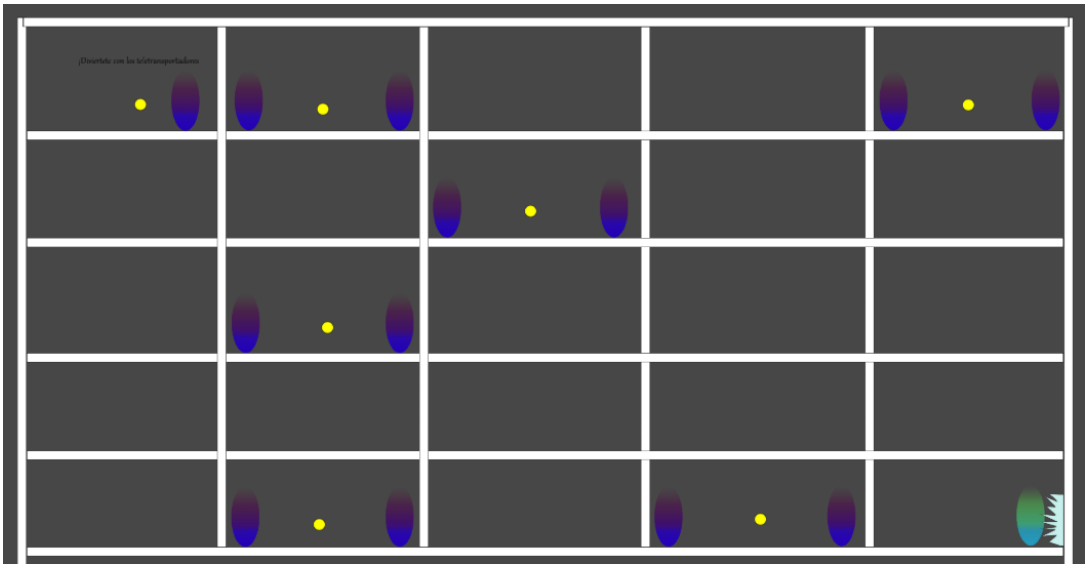
En este nivel hay varios aspectos a tener en cuenta:

- El jugador cuenta con un total de 8 rebotes antes de morir, ya que el octavo hace que impacte contra el techo.
- Los coleccionables se han hecho de mayor tamaño para facilitar la recogida de los mismos.
- Uno de los coleccionables solo se puede recoger con el último de los saltos no mortales, es decir, con el séptimo.
- La salida del nivel solo puede ser alcanzada con el octavo salto, el que provoca la muerte del jugador.

### 4.2.9. Nivel 9

#### 4.2.9.1. Definición

El noveno nivel se trata de un nivel sin *hazards* por el camino. Es un laberinto de transportadores en el que el jugador debe dar con el camino correcto para avanzar. El único peligro que puede tener aquí el jugador es alcanzar la meta sin haber recogido todos los coleccionables, ya que una vez se alcanza la meta no se puede volver atrás. (Ver figura 4.10)



27: 4.10 Nivel 9

#### 4.2.9.2. Información adicional

Si bien en este nivel no hay *hazards* que puedan causar la muerte del jugador en el camino al final del nivel, sí que hay un posible peligro: que pase de largo uno de los coleccionables y llegue al final, sin posibilidad de volver.

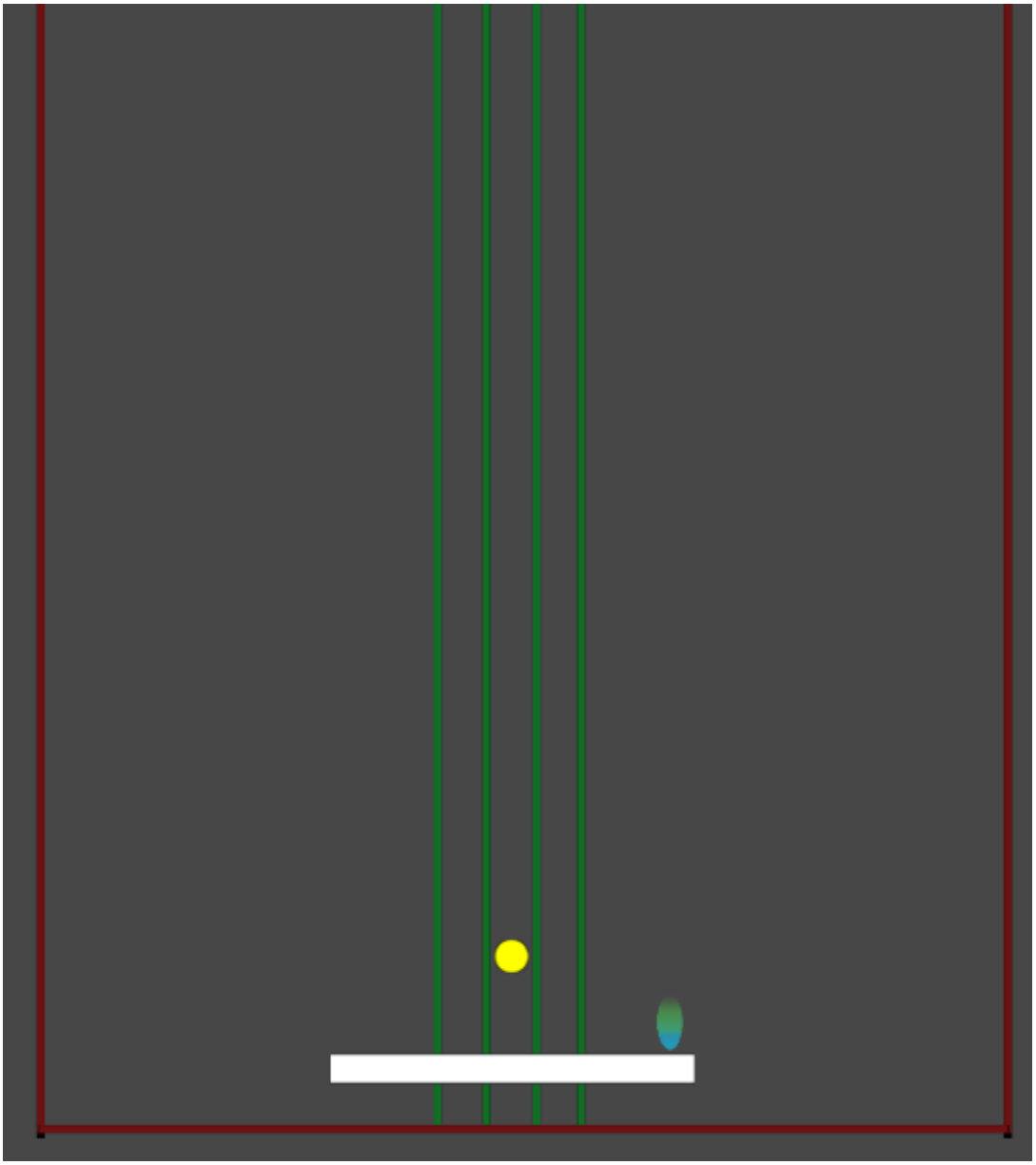
Para evitar este problema, se ha colocado un *hazard* en la habitación final para que el jugador lo pueda usar para morir en caso de que se dé la situación descrita anteriormente.

### 4.2.10. Nivel 10

#### 4.2.10.1. Definición

En el penúltimo nivel el jugador se encontrará en una caída libre durante la cual deberá reaccionar rápidamente a las señales para poder recoger todos los coleccionables antes de llegar abajo, donde se halla el portal al nivel siguiente.

Debido al tamaño del nivel, solo se muestra la parte final del mismo, donde se puede ver el final del nivel junto con un coleccionable y las líneas guía para el jugador. (Ver figura 4.11)



28: 4.11 Nivel 10

#### 4.2.10.2. Información adicional

En este nivel se han puesto guías para hacer más fácil al jugador colocarse correctamente durante la caída. También se han agrandado los coleccionables para que sea más fácil recogerlos durante la bajada.

### 4.2.11. Nivel 11

#### 4.2.11.1. Definición

Nivel final del juego, en esta ocasión en lugar de encontrase el jugador con una única mecánica que afrontar, debe recorrer un nivel con distintos puzles que superar para conseguir los coleccionables.

Algunas de las mecánicas que aparecen en este nivel se han visto anteriormente en otros niveles, como las plataformas que desaparecen o el suelo rebotante. Sin embargo, en este nivel se han añadido algunas mecánicas nuevas, como los mecanismos de puertas con llave o paredes falsas.

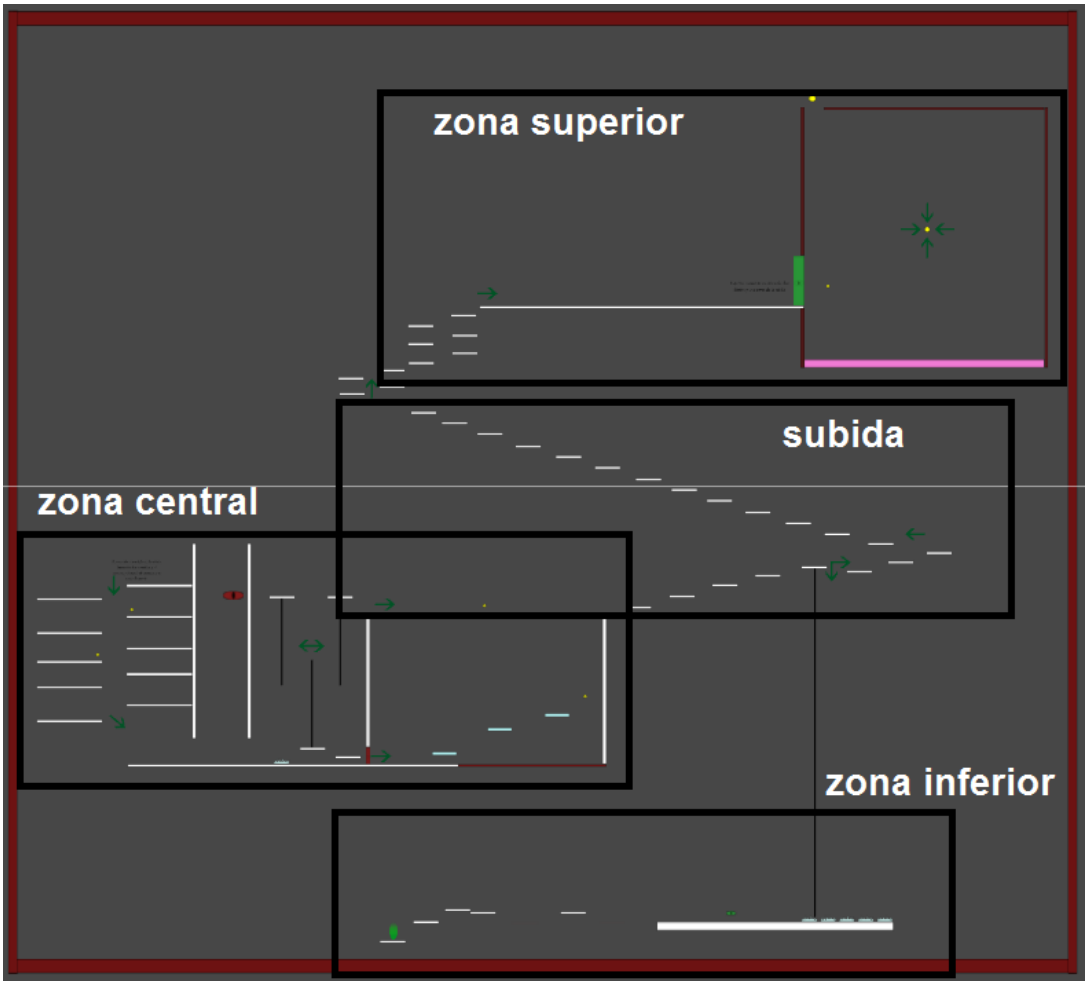
Debido a que el nivel es demasiado grande como para poder abarcarlo en una sola imagen, se ha dividido en varias según indica la figura 4.12.

La zona superior la forma la sala cerrada por la puerta verde. Dentro de esta sala hay tres coleccionables, pero el suelo provocara rebotes constantes y nada más entrar a la sala la entrada se cerrará, dejando como única salida la abertura de la parte superior de la sala. (Ver figura 4.13)

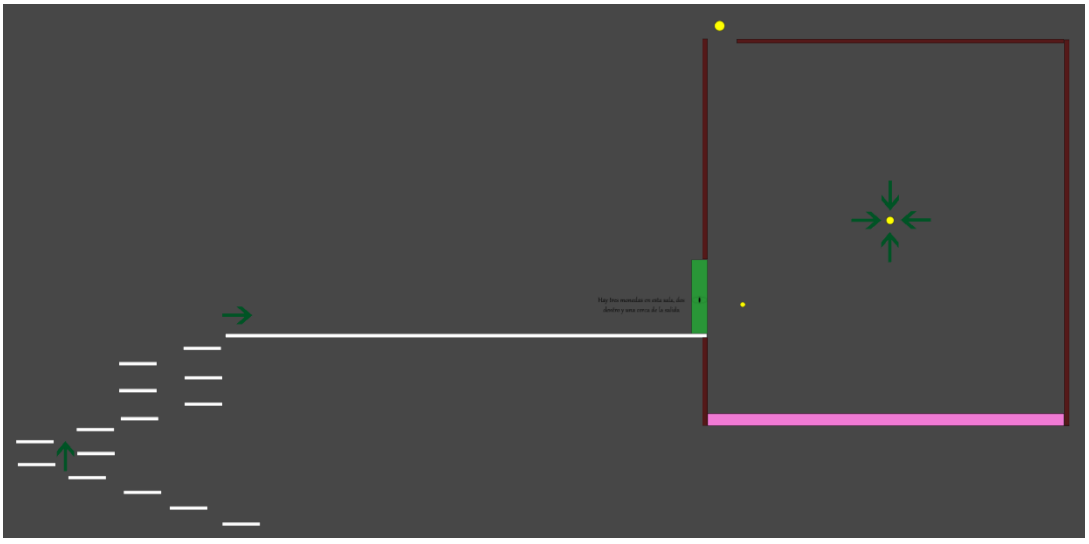
La subida es una serie de plataformas por las que se accede a la zona superior. (Ver figura 4.14)

La zona central es donde el jugador comienza el nivel. Debe recoger los coleccionables en la bajada inicial por las plataformas para luego dirigirse a desbloquear la puerta roja. Dentro de la zona cerrada por la puerta roja hay plataformas que se activan y desactivan que llevan a otro coleccionable más. (Ver figura 4.15)

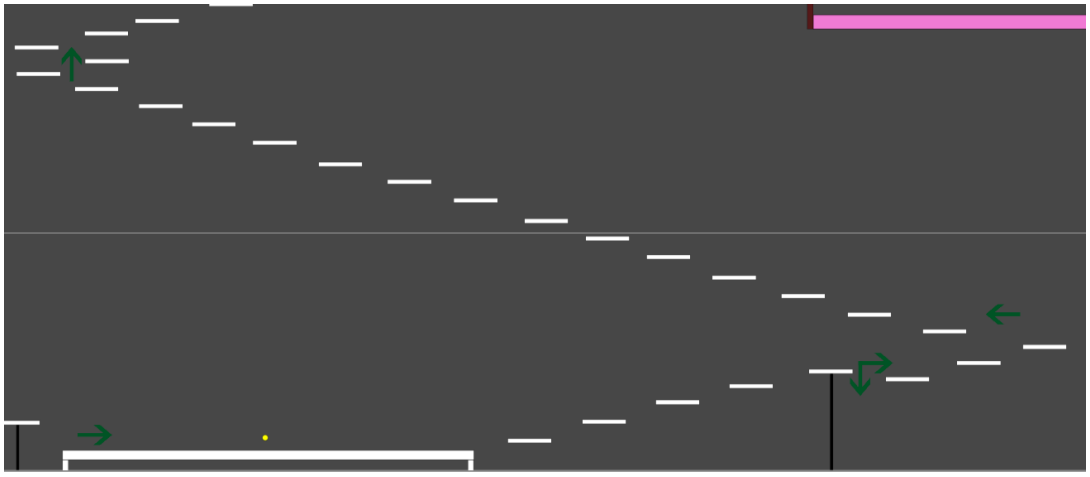
La zona inferior es donde se encuentra la llave para la puerta verde de la zona superior, así como el camino hacia la meta final formado por una serie de plataformas, algunas visibles y otras cercanas a la invisibilidad. (Ver figura 4.16)



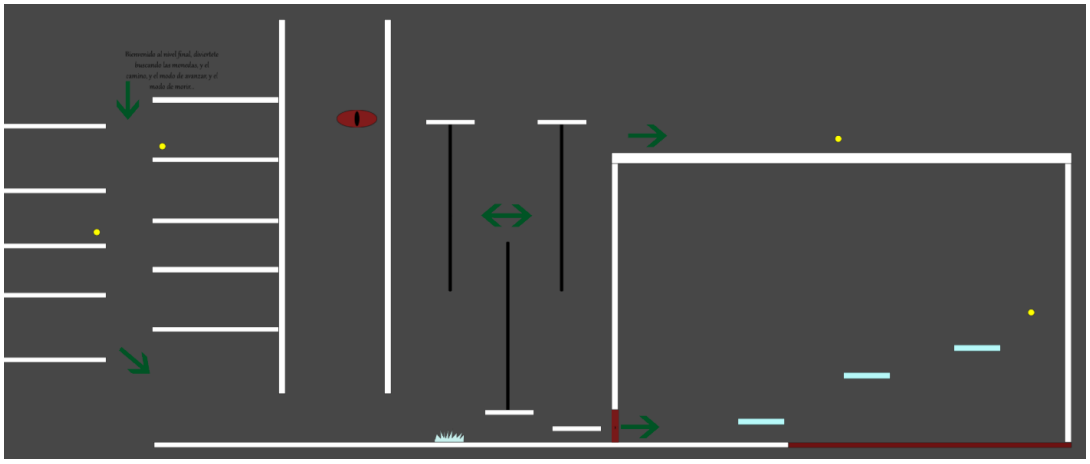

29: 4.12 División nivel 11



30: 4.13 Nivel 11 (Zona superior)



31: 4.14 Nivel 11 (Subida)



32: 4.15 Nivel 11 (Zona Central)

33: 4.16 Nivel 11 (Zona inferior)

#### 4.2.11.2. Información adicional

Este es el nivel final del juego, y en él aparecen ciertos elementos que no se habían visto anteriormente:

- Mecanismos de llave y puertas cerradas:
  - Tienen el mismo color.
  - Al coger la llave la puerta se desactiva, dejando el paso libre.
- Plataformas móviles
  - Se mueven a lo largo de una guía que indica su recorrido.
- Bloqueo automático
  - Al pasar por su zona de activación provocan el cierre de una zona.
- Plataformas invisibles
  - Casi transparentes, un pequeño brillo rojo delata su posición.
- Muros falsos
  - Muro idéntico a los normales, salvo que se puede atravesar debido a que su *collider* ha sido modificado para ello.

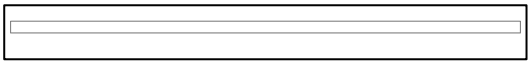
## 4.3. Gráficos estáticos

En esta sección se mostrarán los elementos gráficos usados a la hora de crear los niveles, así como los elementos internos de los mismos.

### 4.3.1. Gráficos del Nivel

Gráficos usados a la hora de crear los elementos básicos de los niveles, plataformas, márgenes, etc.

- Suelos y paredes
  - Se crearon modelos rectangulares para su uso en la creación de elementos como el suelo de los niveles, las paredes, los límites del nivel, plataformas, etc. (Ver figura 4.17)
- Portal
  - Para la creación del portal al siguiente nivel se optó por una elipse semitransparente en tonos verde y azul. (Ver figura 4.18)
  -



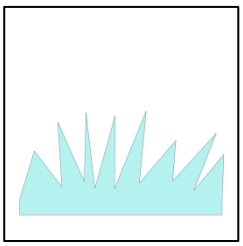
**34: 4.17 Modelo suelos, plataformas y paredes**

**35: 4.18 Modelo portal**

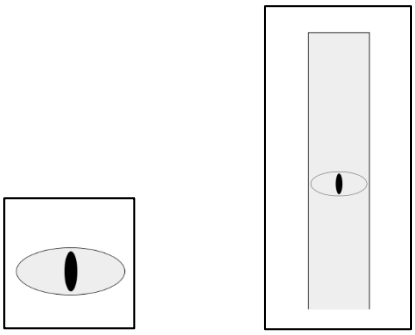
### 4.3.2. Gráficos de Elementos del Entorno

Gráficos usados para crear los distintos elementos internos de los niveles.

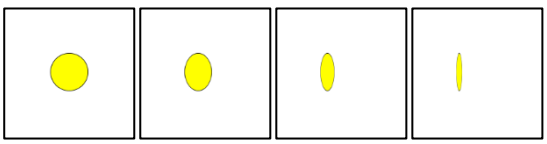
- Pinchos
  - *Hazard*, pinchos con base plana y color azulado para colocar en distintas superficies para crear un elemento a evitar por parte del jugador. (Ver figura 4.19)
- Puertas y llaves
  - Elementos que van en parejas, puerta con símbolo de la llave en su interior y llave. Creados en color blanco para cambiar el color dentro del motor gráfico en función del número necesario. (Ver figura 4.20)
- Monedas
  - Coleccionable a recoger para avanzar en los niveles. Se trata de cuatro imágenes que conforman una animación de moneda girando suspendida en el aire. (Ver figura 4.21)



**36: 4.19 Modelo *hazard***



**37: 4.20 Modelos puerta y llave**



**38: 4.21 Fotogramas animación moneda**

## 4.4. Animaciones Personaje

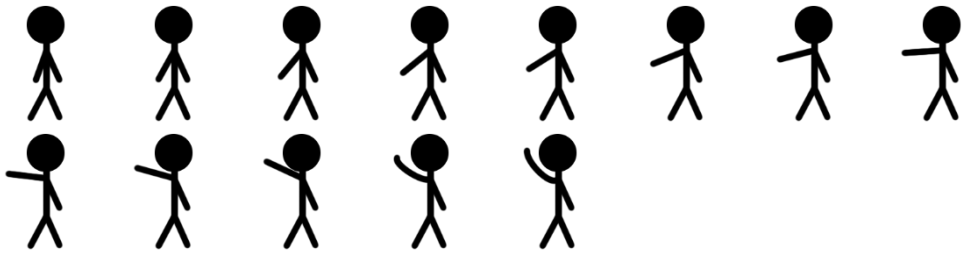
En este apartado se muestran las distintas animaciones que realiza el personaje controlado por el jugador, así como sus fotogramas y la condición para que se lleven a cabo.

### 4.4.1. Reposo

En esta animación nuestro personaje saluda con el brazo mientras mantiene la posición estática. Animación formada por 13 fotogramas distintos. Velocidad de animación de 24 fotogramas por segundo. (Ver figura 4.22)

Durante la animación algunos fotogramas se repiten para hacer la bajada del brazo sin necesidad de crear más fotogramas.

#### 4.4.1.1. Fotogramas



39: 4.22 Fotogramas animación reposo

#### 4.4.1.2. Condición de Activación

Animación por defecto del jugador, se activa si el personaje se encuentra sobre una superficie sólida y el jugador no está introduciendo ningún comando de movimiento, como correr o saltar.

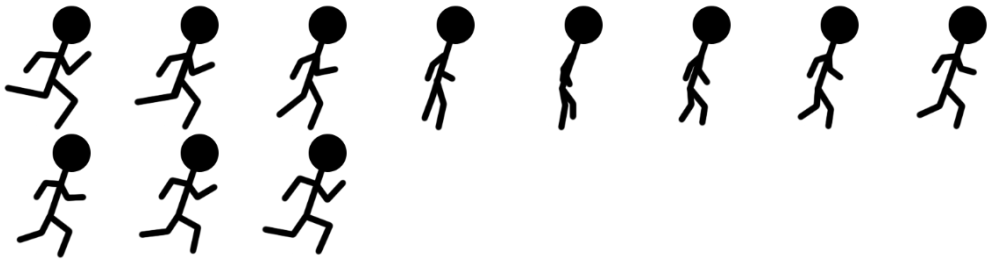
Se llega a esta animación desde la animación de correr si la velocidad horizontal baja a cero.

Se llega a esta animación desde la animación de salto si se aterriza mientras no se está introduciendo ningún comando de movimiento horizontal.

### 4.4.2. Correr

En esta animación nuestro personaje mueve brazos y piernas dando una impresión de movimiento. Animación formada por 11 fotogramas distintos. Velocidad de animación de 24 fotogramas por segundo. (Ver figura 4.23)

#### 4.4.2.1. Fotogramas



40: 4.23 Fotogramas animación correr

#### 4.4.2.2. Condición de Activación

Esta animación se activa mientras el personaje tenga una velocidad horizontal mayor que cero y se desactiva para volver al estado de reposo si esta baja a cero.

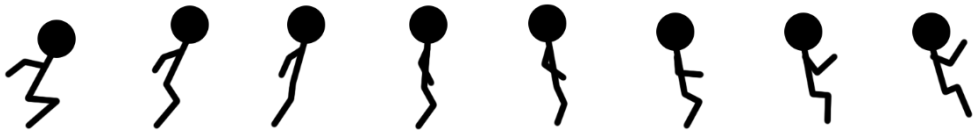
Se llega a esta animación desde la animación de reposo si se introduce un comando de desplazamiento horizontal y la velocidad horizontal se vuelve superior a cero.

Se llega a esta animación desde la animación de salto si al aterrizar se tiene una velocidad horizontal superior a cero.

### 4.4.3. Saltar

En esta animación nuestro personaje mueve piernas y brazos en el aire mientras salta. Animación formada por 8 fotogramas distintos. A diferencia de las otras animaciones, que reproducen una secuencia de fotogramas a cierta velocidad, esta animación decide que fotograma usar en base a la velocidad vertical de nuestro personaje, de forma que la animación sirve tanto para saltar como para caer. (Ver figura 4.24)

#### 4.4.3.1. Fotogramas



41: 4.24 Fotogramas animación salto

#### 4.4.3.2. Condición de Activación

Esta animación se activa cuando el personaje controlado por el jugador no se encuentra sobre terreno, ya sea porque ha saltado o porque ha caído de una plataforma mientras se movía. Se puede llegar a esta animación desde cualquier otra.

Para comprobar si el jugador se encuentra sobre terreno se ha creado una sección bajo sus pies que le indica sobre que terreno se encuentra y si este es sólido o por el contrario se encuentra en el aire.

## 4.5. Interfaz de Usuario (UI)

Para la interfaz de usuario se ha optado por una interfaz sencilla en la que se indiquen los datos estrictamente necesarios.

- En la parte superior izquierda de la pantalla se encuentra el número de coleccionables recogidos en el nivel actual.
  - Color texto: Blanco
  - Tamaño letra: 30
- En la parte central de la pantalla aparecerán dos textos distintos en función de lo ocurrido.
  - Texto de victoria
    - Informa al jugador de que ha superado el juego, aparece al conseguir alcanzar la meta del nivel final.
    - Color texto: Azul claro
    - Tamaño letra: 100
  - Texto de muerte
    - Aparece cuando el personaje muere de alguna forma.
    - Color texto: Rojo
    - Tamaño letra: 100

Todos los textos de la interfaz de usuario usan la fuente llamada *"Perfect Dark"*.

## *4.6. Scripts*

En este apartado se detallarán los scripts usados durante el juego para el funcionamiento del mismo.

### 4.6.1. *Scripts* de Cámara

El *script* encargado del control de la cámara se llama *ControladorCamara.cs*. Se encarga de calcular al comienzo de la partida la posición de la cámara con respecto al jugador para mantener esa relación entre ambos a la vez que la cámara sigue al jugador. (Ver *script* 4.1)

*Script 4.1: Script cámara final*

```
public class ControladorCamara : MonoBehaviour
{

    public GameObject player;
    private Vector3 offset;

    void Start()
    {
        // Calcula la posición de la cámara con respecto a la
posición del jugador al comienzo del juego.
        offset = transform.position - player.transform.position;
    }

    void LateUpdate()
    {
        // Calcula la nueva posición de la cámara en función a la
posición actual del jugador y a la diferencia calculada al comienzo.
        transform.position = player.transform.position + offset;
    }
  }
```

### 4.6.2. *Scripts* de Personaje

El *script* encargado del control del personaje se llama *PlayerControllerScript.cs*. Es el script encargado de diversas funciones relacionadas con el personaje. Acciones específicas de cada sección detalladas en los comentarios de las mismas. (Ver *script* 4.2)

*Script 4.2: Script jugador final*

```
public class PlayerControllerScript : MonoBehaviour {

    //Inicialización de variables
    public float maxSpeed = 7f;
    bool facingRight = true;
    Animator anim;
    bool grounded = false;
    public Transform groundCheck;
    public Text countText;
    private bool canControl = true;
    private int count;
    public Text winText;
    public Text deathText;
    float groundRadius = 0.002f;
    public LayerMask whatIsGround;
    public float jumpForce = 350f;
    private WaitForSeconds m_EndWait;

    void Start ()
    {
        //comienzo del programa, asignando valores necesarios
        anim = GetComponent<Animator>();
        count = 0;
        SetCountText();
```

```
        winText.text = "";
        deathText.text = "";
        m_EndWait = new WaitForSeconds(3f);
    }

     void FixedUpdate ()
    {
        if (canControl == true)
        {
            //comprobación de si el personaje se halla sobre el
suelo o en el aire
            grounded = Physics2D.OverlapCircle(groundCheck.position,
groundRadius, whatIsGround);
            //paso de valores al animador
            anim.SetBool("Ground", grounded);
            anim.SetFloat("vSpeed",
GetComponent<Rigidbody2D>().velocity.y);
            //cogemos valores de dirección y damos velocidad al
personaje
            float move = Input.GetAxis("Horizontal");
            anim.SetFloat("Velocidad", Mathf.Abs(move));
            GetComponent<Rigidbody2D>().velocity = new Vector2(move
* maxSpeed, GetComponent<Rigidbody2D>().velocity.y);

            //hacemos que las animaciones se correspondan con la
dirección en la que nuestro personaje se mueve
            if (move > 0 && !facingRight)
            {
                Flip();
            }
            else if (move < 0 && facingRight)
            {
                Flip();
            }
        }
     }

    void Update()
    {
        if (canControl == true)
        {
            //El personaje salta si se presiona la barra espaciadora
y se encuentra en el suelo
            if (grounded && Input.GetKeyDown(KeyCode.Space))
            {
                anim.SetBool("Ground", false);
                GetComponent<Rigidbody2D>().AddForce(new Vector2(0,
jumpForce));
            }

        }
    }

    //Comprobaciones cuando el personaje entra en contacto con
hitbox tipo "Trigger"
    IEnumerator OnTriggerEnter2D(Collider2D other)
    {
        //Si el objeto es un coleccionable
        if (other.gameObject.CompareTag("PickUp"))
        {
```

```
            // desactivamos el coleccionable recogido y aumentamos
la cuenta de coleccionables actual en uno
            other.gameObject.SetActive(false);
            count++;
            SetCountText();
        }

        //Si el objeto es de tipo interruptor (llaves, activadores
trampa, etc)
        if (other.gameObject.CompareTag("Interruptor"))
        {
            //desactivamos el interruptor
            other.gameObject.SetActive(false);
        }

        //Si el objeto es un elemento dañino
        if (other.gameObject.CompareTag("Hazard"))
        {
            //Nuestro personaje muere, hacemos que la UI lo muestre
y arrebatamos el control al jugador
            anim.SetBool("muerto", true);
            deathText.text = "You Died";
            canControl = false;
            GetComponent<Rigidbody2D>().isKinematic = true;
            //tras una espera de 3 segundos el juego vuelve a
comenzar desde el principio
            yield return m_EndWait;
            Application.LoadLevel(Application.loadedLevel);
        }

        //Si el objeto es la meta final
        if (other.gameObject.CompareTag("Finish"))
        {
            if (count == 7)
            {
                //Hacemos que la UI muestre que se ha vencido y se
arrebata el control al jugador
                winText.text = "You WIN";
                canControl = false;
                GetComponent<Rigidbody2D>().isKinematic = true;
                //tras una espera de 3 segundos el juego vuelve a
comenzar
                yield return m_EndWait;
                Application.LoadLevel(Application.loadedLevel);
            }
        }

        //Si el objeto es alguno de los portales al siguiente nivel
te lleva a el si cumples las condiciones.
            if (other.gameObject.CompareTag("TP1"))
            Teleport(22, -24);

        if (other.gameObject.CompareTag("TP2"))
            Teleport(58, -4);

        if (other.gameObject.CompareTag("TP3"))
            Teleport(122, -16);

        if (other.gameObject.CompareTag("TP4"))
            Teleport(192, -7);
```

```
        if (other.gameObject.CompareTag("TP5"))
            Teleport(279.6f, -23.8f);

        if (other.gameObject.CompareTag("TP6"))
            Teleport(359f, -22f);

        if (other.gameObject.CompareTag("TP7"))
            Teleport(432f, -22.22f);

        if (other.gameObject.CompareTag("TP8"))
            Teleport(500f, -6f);

        if (other.gameObject.CompareTag("TP9"))
            Teleport(592.1f, -5.2f);

        if (other.gameObject.CompareTag("TP10"))
            Teleport(671.2f, -14.87f);

        //Si el objeto es uno de los teleportadores del nivel con
teletransportes te llevan cada uno a su destino particular dentro
del nivel sin comprobar tus coleccionables
        if (other.gameObject.CompareTag("TeleportadorFalso"))
            Warp(500f, -6f);

        if (other.gameObject.CompareTag("Teleportador1"))
            Warp(522f, -12f);

        if (other.gameObject.CompareTag("Teleportador2"))
            Warp(512f, -17.6f);

        if (other.gameObject.CompareTag("Teleportador3"))
            Warp(533f, -27f);

        if (other.gameObject.CompareTag("Teleportador4"))
            Warp(512f, -27f);

        if (other.gameObject.CompareTag("Teleportador5"))
            Warp(544f, -6f);

        if (other.gameObject.CompareTag("Teleportador6"))
            Warp(512f, -6f);

        if (other.gameObject.CompareTag("Teleportador7"))
            Warp(541f, -27f);
    }

    //Función que se encarga de mandarte al siguiente nivel si
tienes todos los coleccionables del actual. También pone la cuenta
de coleccionables a cero para el siguiente nivel.
    void Teleport(float x, float y)
    {
        if (count == 7)
        {
            count = 0;
            SetCountText();
            GetComponent<Rigidbody2D>().velocity = Vector3.zero;
            GetComponent<Rigidbody2D>().angularVelocity = 0;
            this.GetComponent<Transform>().position = new Vector3(x,
y, 0.0f);
```

```
        }
    }

    //teletransporte instantaneo a la posición especificada
    void Warp(float x, float y)
    {
        this.GetComponent<Transform>().position = new Vector3(x, y,
0.0f);
    }

    //Actualiza el texto de la UI encargado del número de
coleccionables
    void SetCountText()
    {
        countText.text = "Count: " + count.ToString();

    }

    //Función para girar las animaciones de nuestro personaje, de
esta forma no hay que hacer animaciones individuales para izquierda
y derecha.
    void Flip()
    {
        facingRight = !facingRight;
        Vector3 theScale = transform.localScale;
        theScale.x *= -1;
        transform.localScale = theScale;
    }
  }
```

### 4.6.3. *Scripts* de Entorno

*Scripts* adicionales de elementos del entorno.

- Puertas
  - Las puertas cuentan con un *script* simple para su apertura llamado *openDoor.cs*. Las puertas tienen asignada una llave de tipo "interruptor", cuando esta llave se desactiva al recogerla el jugador la puerta se desactiva junto con ella. (Ver *script* 4.3)
- Puerta trampa
  - En un punto del nivel final hay un interruptor oculto que activa una trampa. Cuando el jugador activa este interruptor la zona por la que entró queda sellada para evitar su huida. Para esto se usa el *script* llamado *closeDoor.cs*. (Ver *script* 4.4)
- Plataforma móvil
  - Las plataformas móviles de la fase final cuentan con un *script* propio llamado *MovingPlatformY.cs*. Este *script* se encarga de mover la plataforma entre la posición original y la de destino constantemente. (Ver *script* 4.5)
- Plataformas intermitentes
  - Las plataformas intermitentes alternan entre dos posiciones con un intervalo indicado que se controla mediante el *script Parpadeo.cs*. Este

*script* se encarga de cambiar la plataforma entre las dos posiciones de las que dispone para causar el efecto de plataforma que desaparece y aparece. (Ver *script* 4.6)

*Script 4.3: Script abrir puertas final*

```
public class OpenDoor : MonoBehaviour {

    public GameObject Key;

      // Use this for initialization
      void Start ()
      {}

      // Update is called once per frame
      void Update () {

        if(Key.active == false)
        {
this.GetComponent<Collider2D>().gameObject.SetActive(false);
        }
      }
}
```

*Script 4.4: Script mecanismo trampa final*

```
public class closeDoor : MonoBehaviour
{
    public GameObject Interruptor;

    void Start () {}

      void Update ()
    {
        if (Interruptor.active == false)
        {
            Warp(777.45f, 34.36f);
        }
    }

    void Warp(float x, float y)
    {
        this.GetComponent<Transform>().position = new Vector3(x, y,
0.0f);
    }
  }
```

*Script 4.5: Script plataforma móvil final*

```
public class MovingPlatformY : MonoBehaviour {

    private int wait = 0;
    public float increase;
    public int speed = 1;
    private Vector3 origin;
    private Vector3 offset;
    private Vector3 destination;
    private bool ida = true;
    private bool subida = true;

    //Calculamos posiciones de origen y destino y ajustamos la
velocidad del trayecto
    void Start ()
    {
        origin = transform.position;
        destination = new Vector3(origin.x, origin.y + increase,
origin.z);
        offset = (destination - origin)/speed;
        if(destination.y < origin.y)
        {
            subida = false;
        }
      }

      //Vamos actualizando la posición de la plataforma a medida que
va subiendo o bajando
      void Update () {
        wait += 1;
        if (subida)
        {
            if ((wait == 1) && ida)
            {
                transform.position = transform.position + offset;
                wait = 0;
                if (transform.position.y >= destination.y)
                {
                    ida = false;
                }
            }

            if ((wait == 1) && !ida)
            {
                transform.position = transform.position - offset;
                wait = 0;
                if (transform.position.y <= origin.y)
                {
                    ida = true;
                }
            }
        }
        else
        {
            if ((wait == 1) && ida)
            {
                transform.position = transform.position + offset;
                wait = 0;
                if (transform.position.y <= destination.y)
```

```
                {
                    ida = false;
                }
            }

            if ((wait == 1) && !ida)
            {
                transform.position = transform.position - offset;
                wait = 0;
                if (transform.position.y >= origin.y)
                {
                    ida = true;
                }
            }
        }
    }
}
```

*Script 4.6: Script plataforma intermitente final*

```
public class Parpadeo : MonoBehaviour {

    private int wait = 0;
    private bool inicio = true;
    public float aumentoX;
    public float aumentoY;
    public int espera = 50;
    private Vector3 destino;
    private Vector3 origin;

    //Calculamos punto de origen y destino
    void Start ()
    {
        origin = transform.position;
        destino = new Vector3(origin.x + aumentoX, origin.y +
aumentoY, origin.z);
    }

      //cada vez que se alcanza el tiempo indicado se cambia la
plataforma entre la posición actual y la otra
      void Update () {
        wait = wait + 1;
        if (inicio && wait == espera)
        {
            transform.position = destino;
            wait = 0;
            inicio = false;
        }

        if (!inicio && wait == espera)
        {
            transform.position = origin;
            wait = 0;
            inicio = true;
        }
    }
}
```

# Capítulo 5: Resultados

Tras haber realizado el proyecto tal y como se indicó al comienzo del documento, se han alcanzado los resultados siguientes:

- Se ha aprendido acerca del mercado de *Game Engines*.
- Se han aprendido aspectos sencillos y avanzados del manejo del *Game Engine* Unity.
- Se ha profundizado en el conocimiento de los géneros de videojuegos.
- Se ha aprendido el uso de *scripts* para manejar comportamientos de los elementos dentro de un juego.
- Se ha aprendido a crear animaciones 2D.
- Se ha aprendido a añadir audio de fondo a un juego.
- Se han diseñado niveles siguiendo distintos patrones.
- Se han evitado los riesgos del proyecto de manera satisfactoria.
- Se han cumplido las fechas establecidas a pesar de eventos imprevistos.

Al final se han podido obtener como resultados los elementos jugables siguientes:

- Demos de aprendizaje finalizadas con ejecutable funcional:
  - Roll-a-ball Tutorial
  - 2D UFO Tutorial
  - Tanks Tutorial
- Productos propios finalizados con ejecutable funcional:
  - Versión Beta del juego
  - Versión final del juego

# Capítulo 6: Conclusiones y posibilidades de mejora

Una vez finalizado el proyecto y constatado que se han cumplido los objetivos especificados, las opciones de mejora son amplias.

Durante el aprendizaje del motor Unity se pudo vislumbrar la potencia del mismo a la hora de crear contenido, y apenas se ha arañado el verdadero potencial de este motor durante este proyecto. Unity dispone de herramientas avanzadas en las que no se ha profundizado, así como de elementos de código para usar en *scripts* y ampliar el funcionamiento de los mismos.

En primer lugar se podría comenzar mejorando el apartado de las colisiones y las animaciones. En nuestro proyecto el personaje controlable contaba con animaciones para diversas situaciones. Sin embargo, a pesar de que el aspecto del modelo cambiaba, las *hitbox* para la colisión con unidades se mantenían fijas en el modelo. Se podría mejorar el modelo actual modificando las animaciones o alterando de algún modo las *hitbox* durante la animación para que se correspondieran con el nuevo modelo gráfico.

El apartado gráfico es otro punto a mejorar. En nuestro proyecto se ha usado un estilo gráfico simple y sencillo debido a que no se tenía la destreza necesaria en el arte del diseño. Con tiempo y práctica los elementos gráficos del juego podrían mejorarse, y gracias a las características de Unity sería bastante sencillo sustituir los modelos antiguos por los mejorados.

Otro punto a mejorar sería el diseño de niveles. En este proyecto se han creado niveles pequeños y simples que con más tiempo y planificación podrían mejorarse. También cabría mencionar que la mecánica inicial del juego está basada en superar niveles sencillos con una sola vida, y que si los niveles aumentaran en dificultad y longitud habría que alterar el mecanismo añadiendo puntos de control que no hicieran frustrante la experiencia de juego.

# Capítulo 7: Referencias

Todas las referencias a páginas web han sido consultadas entre los meses de mayo y agosto de 2016.


[$3DJ_a$] 3djuegos.com. .Análisis *Grand Theft Auto 5*.

http://www.3djuegos.com/juegos/analisis/6992/0/grand-theft-auto-v/

[ANT 2009] Antonio Checa Godoy (2009). *Hacia una industria española del videojuego.* Comunicación, Nº 7, Vol.1, año 2009, PP. 177-188. ISSN 1989-600X.

http://www.revistacomunicacion.org/pdf/n7/articulos/a12_Hacia_una_industria_espanola_del_videojuego.pdf

[$HOB_a$] hobbyconsolas.com. Análisis *Metal Gear Solid V: The Phantom Pain*

*http://www.hobbyconsolas.com/reviews/analisis-metal-gear-solid-v-phantom-pain-124440*

[$HOB_b$] hobbyconsolas.com. Análisis *Mario Kart 8*

http://www.hobbyconsolas.com/reviews/analisis-mario-kart-8-71370

[$MER_a$] meristation.com. Análisis *Super Mario 64*

*http://www.meristation.com/nintendo-64/super-mario-64/analisis-juego/1507898*

[$MER_b$] meristation.com. Análisis de *League of Legends*

http://www.meristation.com/pc/league-of-legends/analisis-juego/1526971

[SIM CRI 2008] Belli, Simone y López, Cristian (2008). *Breve historia de los videojuegos*. *Athenea Digital*, 14, 159179.

[$UNI_a$] Sitio web oficial de Unity3D.

https://unity3d.com/es

[$UNI_b$] Sitio web oficial de Unity3D. Información del editor.

https://unity3d.com/es/unity/editor

[$UNI_c$] Sitio web oficial de Unity3D. Información de plataformas.

https://unity3d.com/es/unity/multiplatform

[$UNI_d$] Sitio web oficial de Unity3D. Roadmap.

https://unity3d.com/es/unity/roadmap

[$UNI_e$] Sitio web oficial de Unity3D.

https://unity3d.com/es/learn

[$UNI_f$] Sitio web oficial de Unity3D. Tutorial Roll-a-ball.

https://unity3d.com/es/learn/tutorials/projects/roll-ball-tutorial

[$UNI_g$] Sitio web oficial de Unity3D. Tutorial 2D UFO.
https://unity3d.com/es/learn/tutorials/projects/2d-ufo-tutorial
[$UNI_h$] Sitio web oficial de Unity3D. Tutorial Tanks.
https://unity3d.com/es/learn/tutorials/projects/tanks-tutorial
[$UNI_i$] Sitio web oficial de Unity3D. Videos relacionados con creación en 2D.
https://unity3d.com/es/learn/tutorials/topics/2d-game-creation
[$UNR_a$] Sitio web oficial de Unreal Engine 4.
https://www.unrealengine.com/what-is-unreal-engine-4
[$UNR_b$] Sitio web oficial de Unreal Engine 4. Realidad virtual.
https://www.unrealengine.com/vr
[$UNR_c$] Sitio web oficial de Unreal Engine 4. Documentación.
https://docs.unrealengine.com/latest/INT/
[$UNR_d$] Sitio web oficial de Unreal Engine 4. Video tutoriales.
https://docs.unrealengine.com/latest/INT/Videos/
[$UNR_e$] Sitio web oficial de Unreal Engine 4. Marketplace.
https://www.unrealengine.com/marketplace
[$UNR_f$] Roadmap de Unreal Engine 4.
https://trello.com/b/gHooNW9I/ue4-roadmap

# Capítulo 8: Anexos

## 8.1. Manual de Instrucciones

Basándose en la dificultad de juegos de hoy en día como *"Dark Souls",* e inspirado por los gráficos 2D de los juegos de plataformas antiguos tal como el afamado *"Super Mario Bros", "Just 1 More Time..."* nos embarca en una aventura de plataformeo en la que el mínimo error nos puede costar muy caro.

En *"Just 1 More Time...",* de ahora en adelante *J1MT* para abreviar, nuestro protagonista debe adentrarse en diversos niveles, llamados "Encrucijadas", con el objetivo de conseguir las siete llaves que abren el camino a la siguiente. Sin embargo, no todo es un camino de rosas, ya que en las Encrucijadas aguardan cientos de peligros dispuestos a acabar con nosotros, provocando en caso de lograrlo que todo nuestro progreso se disolviera, forzándonos a comenzar de nuevo una vez más nuestro viaje en el Laberinto de Encrucijadas.

Quien sabe, tal vez solo un nivel más allá de la última Encrucijada a la que llegaste se hallaba ese final que tanto ansiabas, de modo que adelante, adéntrate en el Laberinto de Encrucijadas, solo, una vez más...

### *Elementos del escenario*

Qué vacío estaría el mundo sin cosas que lo llenaran, menos mal que aquí estoy yo para llenar de vidilla este cotarro, que cuando llegué esto estaba más vacío que un colegio en domingo.

Bueno, dejemos las presentaciones de cada cosa para más adelante, que este no es el lugar adecuado para hablar de muerte y destr... digoooo, de amor y amistad, sí, eso, amor y amistad para todos.

### *Enemigos*

¿Qué sería de un héroe sin villanos?

En *J1MT* nuestro héroe comienza sus andanzas solo, sin más compañía que la de la ilusión por comenzar una nueva aventura. Pobre infeliz, no sabe lo que le espera, ya que si bien al comienzo las Encrucijadas pueden parecer inofensivas y una divertida aventura, poco tardará en darse cuenta de que las profundidades ocultan cosas que es mejor que no salgan a la luz.

Pero quien sabe, tal vez nuestro protagonista agradezca algo de compañía de vez en cuando.

Enemigo 1 – La ignorancia

¿Qué es más peligroso y mortal que el hecho de no saber a qué te enfrentas?

Más peligroso que cualquier cosa que yo sea capaz de invocar desde las profundidades más recónditas del averno, es la ignorancia. La ignorancia es lo que más veces acabará contigo, ya que al no saber lo que espera tras la esquina, o si ese camino te llevará a la muerte o a tu destino, o si no sabes con que tecla se salta porque fuiste tan burro como para empezar a jugar sin leerte el manual y ahora no puedes esquivar esa bola de fuego que va directa a tu cara, todo ello vendrá de mano del mayor de tus enemigos: La Ignorancia.

## *Obstáculos*

¿Quién jugaría a algo que fuese simplemente presionar un botón una y otra vez para ganar?

En *J1MT* no solo los peligros o los enemigos deberán preocuparte, sino también el camino que has de recorrer. Puzles, mecanismos, palancas, transportadores, puertas cerradas con llave a cal y canto, suelos helados que te llevarán al vacío si no mides tus pasos con cautela.

En fin, al menos te queda un consuelo, si mueres por culpa de algo de esta categoría no es que el juego sea cruel contigo, es que TÚ necesitas mejorar.

Obstáculo 1 – Plataformas móviles

Vaya, parece que te acabas de topar de cabeza con un abismo sin fondo (o casi), o con un camino sin salida, más no temas, ya que el camino sigue (o al menos a cachos).

Estas preciosidades vienen en todas las formas y colores que te puedas imaginar, y encima son útiles, ya que te llevarán volando a través de esos sitios por los que creías que no podías pasar, mira que son majetes. Al menos hasta que tienes saltar entre varios de ellos que se mueven como si les hubiera dado un ataque de epilepsia sobre un foso lleno de lava, si te mueres ahí, es tu culpa.

Obstáculo 2 – Puertas

Oooooh, mira qué bonita llave hay ahí delante, es la última que te falta para superar el nivel en el que te encuentras, y te ha costado lo tuyo llegar hasta aquí. Sería una verdadera lástima que "alguien" hubiera puesto justo en el medio una puerta indestructible, en fin sigue adelante a ver si encuentras algo que te ayude con esto.

PD: ¿A que el color de la puerta es bonito? Te recomiendo que lo recuerdes bien.

Obstáculo 3 – Llaves (Pero no las que buscas, estas tienen colores)

¿Ves como no hacía falta que dijeras todas esas cosas acerca de mí y de mi familia solo por una puertecita de nada?

¿Recuerdas esa puerta que antes te bloqueaba el camino? Espero que recuerdes que tenía un color muy bonito, al menos. Bueno, pues he aquí la solución al misterio: una llave abre una puerta, sencillo, ¿verdad?

Vale, tal vez no sea tan sencillo, así que trataré de explicarlo de otra forma a ver si así logras entenderlo. Cuando coges una llave de color, la puerta correspondiente se abre. Espero no tener que volverlo a repetir.

Obstáculo 4 – Caminos ocultos

¡Eh, mira, la llave que necesitaba para abrir la puerta esa de color tan bonito!

Qué pena que esté encerrada dentro de ese muro, y esta vez no hay puerta que se pueda abrir, solo un frío, grueso y duro muro imposible de atravesar.

Menos mal que los caminos ocultos están a la orden del día y cierta parte del muro en realidad no es muro, sino un camino oculto que lo parece.

De ti depende encontrarlo, yo te doy una pista: prueba siempre en primer lugar con aquellos muros que en caso de ser muros reales te llevarían sin lugar a duda a una muerte horrible, confía en mí, somos amigos ¿no?

Obstáculo 5 – Plataformas invisibles (o casi)

"Vaya, parece que el camino sigue por ahí delante, pero no puedo llegar a la plataforma de un salto o me caeré a esa lava tan caliente de ahí abajo y me moriré, creo que me quedaré aquí a esperar, seguro que viene una plataforma móvil y me ayuda a avanzar."

Espero que en serio no hayas pensado esto, en caso de que lo hayas hecho me voy a divertir mucho viéndote esperar sin ningún resultado, y todo por no fijarte bien en la pantalla, así que límpiate las gafas, mira otra vez y me avisas.

¿Ya te las has limpiado? Bien, entonces supongo que ahora sí que podrás intuir una figura vagamente en medio de ese abismo, te recomiendo que intentes saltar encima, confía en mí, ¿cuándo te he mentido?

Obstáculo 6 – Suelo rebotante

Uy, mira, suelo rosa. Seguro que eso es todo lo que piensa tu cerebro del tamaño de una nuez, héroe de pacotilla, ya que no eres capaz de vislumbrar a través de mi gloriosa creación: el suelo rebotante.

Puedes pensar que no es nada más que suelo pero de un color diferente, y no podrías estar más equivocado. Este suelo multiplica la fuerza con la que saltas sobre él cada vez más, haciendo que alcances alturas vertiginosas en menos que canta un gallo, saltando a toda velocidad hacia arriba, normalmente directo a tu muerte, pero no seré yo quien te impida disfrutar.

Obstáculo 7 – Plataformas intermitentes

Ahora hay suelo. Ahora no. Ahora hay suelo. Ahora no. Ahora hay suelo. Ahora no. Simple. Eficaz. Para toda la familia. Si es que queda familia tras caer al abismo al desaparecer el suelo bajo tus pies por no haber sido capaz de calcular correctamente el tiempo del que disponías.

## *Peligros (Trampas)*

Bueno, me parece que ya has visto mi precioso ejército de enemigos, mis endiablados obstáculos y mis astutos engaños y te pensarás: ¿Ya no me puede quedar nada más, no?

Pues te equivocas, ya que ahora llegamos a mi parte preferida, las trampas.

Que sería de una mazmorra sin trampas, los enemigos están bien y te hacen compañía, los obstáculos te entretienen y te hacen pensar, pero las trampas tienen ese algo encantador y maravilloso que le da un toque mágico a toda mazmorra que se precie.

Suelos que se abren bajo tus pies, pinchos que esperan ansiosos tu caída para recibirte con su amoroso abrazo, lava que te derrite, cepos que te atrapan, piedras que te aplastan, rayos que te fulminan, interruptores que las activan...

En resumen, una auténtica obra de arte dedicada exclusivamente para tu disfrute, y el mío también, ya que estamos.

Peligro (Trampa) 1 – Pinchos

Da igual que seas un robot superpoderoso, un valiente fontanero dispuesto a salvar a tu princesa de las garras de una tortuga gigante con tu flamante traje verde, o un erizo que no sabe que los erizos pueden nadar, pínchate el dedo gordo del pie con una de estas preciosidades y ya verás lo deprisa que tu viaje llega a su fin.

Los pinchos son adorables, ya sea en forma de estaca de madera, púas de acero o simples píxeles apilados en forma triangular, su hábitat natural es estar estáticos en algún lugar esperando pacientemente a que te acerques a saludarles y a darles un abrazo afectuoso (te invito a hacerlo).

He oído por ahí que algunos pinchos están preparando fiestas sorpresa en las que se lanzan a abrazarte en cuanto te acercas, pero a mi no me han querido invitar, no se por qué.

Peligro (Trampa) 2 – Fin del Mundo (en realidad solo de la Encrucijada)

Porque todo tiene un final, no te pienses que una vez que entras en una de mis encrucijadas puedes salir indemne de ella, de eso nada.

Solo hay una salida de mis Encrucijadas, y es el portal que te llevará a la siguiente, puedes intentar salir por algún otro sitio, pero no creo que te guste lo que encontrarás.

Veo que lo has querido comprobar, bien, ¿qué se siente al chocarte de cara contra el Fin del Mundo? El nombre se me ocurrió a mí solito, es una adorable pared de magia pura que destruye tu ser hasta lo más hondo y te devuelve al comienzo de la primera Encrucijada, haciéndote empezar de nuevo desde el principio (bueno vale, eso lo hace todo lo que te mata, pero queda guay).

## *Jugador*

Bien, aquí vamos, me toca hablar de tí, y mira que dije que le dieran el trabajo a otro, que por mi yo te dejaba atado de pies y manos al comienzo de cada nivel mientras una marabunta de hormigas caníbales se dirigían a devorarte vivo, pero no me dejaron. Y por si fuera poco encima vinieron los del sindicato de héroes de juegos de plataformas y me dijeron que debía, si si si, lo has oído bien, que YO debía darte habilidades para poder avanzar por el juego y poder superar los obstáculos hasta llegar al final de cada nivel. Bueno, como tras las últimas reformas y el último lote de trampas que encargué me he quedado básicamente sin un duro, voy a tener que acceder por mucho que no quiera, no puedo volver a meterme en líos con el sindicato después de lo de la última vez, ¿Quién iba a pensar que un nivel subacuático usando ácido en vez de agua era una mala idea?

Habilidades del Jugador (o sea TÚ)

Como ya he dicho antes, no me gusta ayudarte, así que no esperes gran cosa de mi parte, solo cumpliré el mínimo para que el sindicato no tenga nada que usar en mi contra cuando la palmes.

Habilidad 1 – Andar (parece que es importante que puedas)

¿En serio tengo que explicarte esto? Bueno, seré breve.

Presionas hacia la derecha en las flechas direccionales del teclado o la tecla D y te mueves hacia la derecha, fácil, hasta un bebé sabría hacerlo, pero parece que tú necesitas que te lo explique.

Presionas hacia la izquierda en las flechas direccionales del teclado o la tecla A y te mueves hacia la izquierda, fácil, hasta un bebé sabría hacerlo, pero parece que tú necesitas que te lo explique.

Acabo de tener una sensación de Deja Vú, ¿tú también?

Bueno, yo quería haberte puesto los controles al revés, pero no me dejaron, eso sí me dijeron que podía crear una trampa que lo hiciera, ya me lo pensaré.

Habilidad 2 – Saltar (porque caminar no era suficiente)

En fin, en el sindicato me dijeron que también tenías que ser capaz de saltar para que un escaloncito de nada no arruinase tu experiencia de juego, así que en un gesto de magnanimidad te concedo la habilidad de saltar presionando la barra espaciadora.

De nada.

Habilidad 3

Me siento de buen humor hoy, así que hasta te daré una habilidad más. Si mantienes presionada la dirección en la que miras cuando te chocas contra una pared en medio del aire, te quedarás agarrado a ella, así será más divertido verte intentar sobrevivir a una caída mientras no puedes hacer nada para evitarlo.

## *Encrucijadas*

Porque todo buen genio malvado creador de mazmorras no es capaz de hacerlo todo bien, he de admitir que viene bien contar con una pequeña diablilla artista que me echa una

mano en la creación de las Encrucijadas, así que en esta sección no puedo evitar describir con todo lujo de detalles las maravillosas Encrucijadas que me ha ayudado a crear.

PD: Héroe, aléjate de aquí, mirar esta sección es trampa ya que en ella se van a detallar los detalles de la creación de Encrucijadas, cosa que tu insignificante mente no soportaría y te provocaría un derrame cerebral por los siglos de los siglos, y no, no es que no quiera que veas aquello a lo que te vas a enfrentar.

PD2: ¿De verdad te has creído que aquí habría algo de utilidad para ti?