Shiane

Oscar Alonso Bustos
Diseño y desarrollo de videojuegos
Área de Posgrado Informática, Multimedia y Telecomunicación

Helio Tejedor Navarro
Javier Luis Cánovas Izquierdo

2019-06-16

Esta obra está sujeta a una licencia de Reconocimiento-Compartir Igual 3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	Shiane
Nombre del autor:	Oscar Alonso Bustos
Nombre del consultor/a:	Helio Tejedor Navarro
Nombre del PRA:	Javier Luis Cánovas Izquierdo
Fecha de entrega (mm/aaaa):	06/2019
Titulación:	Diseño y desarrollo de videojuegos
Área del Trabajo Final:	Área de Posgrado Informática, Multimedia y Telecomunicación
Idioma del trabajo:	Español
Palabras clave	Desarrollo de Videojuego

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

El diseño y desarrollo de un videojuego no es algo que se pueda hacer a la ligera, sino que se necesita de tiempo y recursos para poder alcanzar a realizar aquello que se ha pensado.

Llevar una idea simple de un papel a un videojuego completo requiere de mucho tiempo y de saber cómo y que se debe hacer. Requiere planificación y establecer todos los objetivos que se quieren alcanzar, obviando aquellos que esta fuera del alcance, ya sea por presupuesto o por capacidad de desarrollo.

En este proyecto, se pretende establecer una idea y, en un tiempo establecido, llevarla a cabo. No hay otro limite que el tiempo y mi propia capacidad. Todo lo aprendido a lo largo del master y todo lo que pueda aprender durante el desarrollo se debe poner en práctica aquí para desarrollar el mejor producto que sea capaz de hacer por mi propia cuenta.

La idea de la que se parte es la de desarrollar un videojuego de plataformas 2D, donde el jugador deberá superar todos los obstáculos y enemigos que se le presenten para llegar al final.

Abstract (in English, 250 words or less):

Creating a video game is not an easy task that can be done in a few days. It takes a lot of time and resources to create what was planned.

When someone wants to create a video game, no matter how simple the original idea is, he will have to know how and what to do. He needs to plan and establish the goals that he wants to be implemented in the game, he needs to put on the table what he is able to do and what he is not, the time and resources that he has and so on.

Everything I have learnt over these two years of the master's must be put into practice to develop the best product I can do on my own. The time I was granted and my ability are the only two limitation I have to create the game.

The game I am starting from is a 2D platformer game, where the player will have to jump and get over a lot of obstacles and enemies that will try to stop her from reaching the end.

Índice

Tabla de contenido

1. Introducción 1
1.1 Contexto y justificación del Trabajo 1
1.2 Objetivos del Trabajo 1
1.3 Enfoque y método seguido 1
1.4 Planificación del Trabajo 1
1.5 Breve sumario de productos obtenidos 3
1.6 Breve descripción de los otros capítulos de la memoria 3

2. Estado del arte 5
2.1 Plataformas 2D como género 5
2.2 Principales características 5

3. Definición del juego 6
3.1 Conceptualización 6
3.1.1 Breve descripción del juego 6
3.1.2 Tipo de interacción juego-jugador 6
3.2 Ambientación 7
3.2.1 Ambientación e historia 7
3.2.2 Personajes y elementos 8
3.2.3 Interacción entre los actores del juego 9
3.3 Objetivos planteados al jugador 11
3.4 Concept Art 11

4. Diseño técnico 14
4.1 Entorno 14
4.2 Requerimientos del entorno 16
4.3 Herramientas 16
4.4 Assets 17
4.4.1 Protagonista 17
4.4.2 Arquero enemigo 18
4.4.3 Lancero enemigo 19
4.4.4 Riole, *boss* final 19
4.4.5 Objetos y elementos mágicos 19
4.4.6 Terreno 22
4.4.7 *Background* 24
4.4.8 Elementos decorativos 25
4.4.9 Sonidos 27
4.5 Arquitectura del juego 29
4.5.1 Jugador 30
4.5.2 Enemigos 31
4.5.3 *Boss* 31
4.5.4 Demás elementos 32
4.5.6 Eventos 32
4.6 IA de los enemigos 32
4.6.1 Arqueros y lanceros 32

4.6.2 Riole 33
4.7 Mecánicas 33
4.7.1 Movimiento 33
4.7.2 Salto 33
4.7.3 Salto contra la pared 33
4.7.4 *Dash* 33
4.7.5 Escudo 34
4.7.6 Bola de fuego 34
4.7.7 Enemigos 34
4.7.8 Pinchos 34
4.7.9 Plataformas 34
4.7.10 Diamantes 34
4.7.11 La ausencia de *checkpoints* 34
4.7.12 Combinaciones 34

5. Diseño de niveles 35
5.1 Testeo y balanceo 47
5.1.1 Testo 47
5.1.2 Balanceo 48

6. Manual de usuario 49
6.1 Controles 49

7. Conclusiones 51
7.1 Lecciones aprendidas 51
7.2 Logros alcanzados 51
7.3 Seguimiento y análisis 52
7.4 Mejoras 52

8. Glosario 54

Bibliografía 56

Lista de figuras

Ilustración 1 Diagrama Gantt planificación total 2
Ilustración 2 Diagrama Gantt planificación versión parcial 2
Ilustración 3 Diagrama Gantt planificación versión jugable 3
Ilustración 4 Enemigo arquero disparando 11
Ilustración 5 Enemigo arquero andando 11
Ilustración 6 Tyson, el protagonista, andando 12
Ilustración 7 Escudo 12
Ilustración 8 Tyson con escudo 12
Ilustración 9 Engranajes 12
Ilustración 10 Pinchos 12
Ilustración 11 Terreno niveles externos 13
Ilustración 12 Terreno niveles internos 13
Ilustración 13 Todos los sprites de Tyson 17
Ilustración 14 Arquero completo 18
Ilustración 15 Lancero completo 19
Ilustración 16 Riole 19
Ilustración 17 Bola de fuego 20
Ilustración 18 Escudo 20
Ilustración 19 Diamante de fresco 20
Ilustración 20 Flecha 21
Ilustración 21 Flecha completar *sprites* 21
Ilustración 22 Arco para completar *sprites* 21
Ilustración 23 Lanza para completar *sprites* 21
Ilustración 24 Pinchos 22
Ilustración 25 Engranajes 22
Ilustración 26 Terreno básico exterior 23
Ilustración 27 Terreno básico de un solo *sprite* 23
Ilustración 28 Plataformas flotantes tutorial 23
Ilustración 29 Terreno básico de la Torre de los Dioses 23
Ilustración 30 Plataformas trampa 24
Ilustración 31 Plataformas especiales 24
Ilustración 32 Cielo *background* 25
Ilustración 33 Montañas en primera capa 25
Ilustración 34 Montañas en segunda capa 25
Ilustración 35 Nubes con distinto tamaño y forma 25
Ilustración 36 Varios elementos decorativos exteriores 26
Ilustración 37 Conjunto de *sprites* que forman un árbol 26
Ilustración 38 Elementos decorativos interiores 26
Ilustración 39 *Background* interior 26
Ilustración 40 *Alt background* interior 27
Ilustración 41 GameManager arquitectura 30
Ilustración 42 Arquitectura jugador 30
Ilustración 43 Arquitectura enemigo 31
Ilustración 44 *Walkthrough* fase 1 35
Ilustración 45 *Walkthrough* fase 2 36
Ilustración 46 *Walkthrough* fase 3 36

Ilustración 47 *Walkthrough* fase 4 37
Ilustración 48 *Walkthrough* fase 5 38
Ilustración 49 *Walkthrough* fase 6 38
Ilustración 50 *Walkthrough* fase 7 39
Ilustración 51 *Walkthrough* fase 8 40
Ilustración 52 *Walkthrough* fase 9 41
Ilustración 53 *Walkthrough* fase 10 41
Ilustración 54 *Walkthrough* fase 11 42
Ilustración 55 *Walkthrough* fase 12 42
Ilustración 56 *Walkthrougn* fase 13 43
Ilustración 57 *Walkthrough* fase 14 44
Ilustración 58 *Walkthrough* fase 15 45
Ilustración 59 *Walkthrough* fase 16 46
Ilustración 60 *Walkthrough* fase 17 46
Ilustración 61 Controlador de la Xbox 49

Sonido 1 Sonido de las pisadas del protagonista 27
Sonido 2 Sonido del salto del protagonista 27
Sonido 3 Sonido al hacer dash 27
Sonido 4 Sonido al disparar la bola de fuego 27
Sonido 5 Sonido clic menú 28
Sonido 6 Sonido victoria 28
Sonido 7 Sonido muerte jugador 28
Sonido 8 Sonido impacto flecha contra escudo 28
Sonido 9 Sonido disparo flecha 28
Sonido 10 Sonido al atravesar un diamante 29

1. Introducción

1.1 Contexto y justificación del Trabajo

Desarrollar un videojuego es una tarea que requiere un esfuerzo titánico y más cuando no se tiene experiencia en dicho proceso. Hay miles de cosas a tener en cuenta que pueden ser pasadas por alto y luego volver con más fuerza, haciendo que perdamos más tiempo y recursos de los que hubiesen sido necesarios. La finalidad de este trabajo es hacernos pensar en todos estos puntos mientras desarrollamos el videojuego que nosotros queramos.

No hay mejor forma que desarrollar un videojuego de principio a fin para que pongas en práctica todo lo que se ha estudiado durante estos dos años. Hacernos trabajar desde el punto en el cual se nos ocurre la idea del videojuego, hasta que lo hemos desarrollado. Pasando por todos aquellos puntos intermedios, que incluye no solo el diseño de los personajes y los niveles y la programación del mismo videojuego, sino también la planificación de los recursos y del tiempo, del trabajo que se ha de realizar, de establecer que podemos hacer y que no dentro del tiempo que tenemos del que disponemos. Todos estos puntos mencionados y muchos más se van a explicar en este documente que nos invita a conocer el videojuego en profundidad.

1.2 Objetivos del Trabajo

El objetivo de este trabajo está claro: desarrollar un videojuego con todo lo que ello conlleva, es decir, el videojuego final y su correspondiente Documente de Diseño de Videojuego.
Este es el objetivo final que tenemos nosotros, sin embargo, el verdadero objetivo, como ya se ha comentado en el punto anterior, no es solo que el producto final, es decir, el videojuego, este completo y bien, si no entender y realizar todo lo que nos permite llegar a ese punto: la planificación y el diseño previo del videojuego, así como el desarrollo como tal.

1.3 Enfoque y método seguido

Como ya se ha comentado, no hay mejor forma que aprender a desarrollar un videojuego que ponerlo en práctica. A partir de una idea, se debe realizar todo el proceso de desarrollo, sin saltarse ningún punto. Como ya veremos, estableceremos y planificaremos las tareas, se creará un documento de diseño del videojuego y se programará el propio juego, haciendo una pequeña demo que tendrá las principales mecánicas del juego que se ha pensado crear.

1.4 Planificación del Trabajo

En este proyecto hay tres días de entrega marcados, en los cuales hay que cumplir unos mínimos.

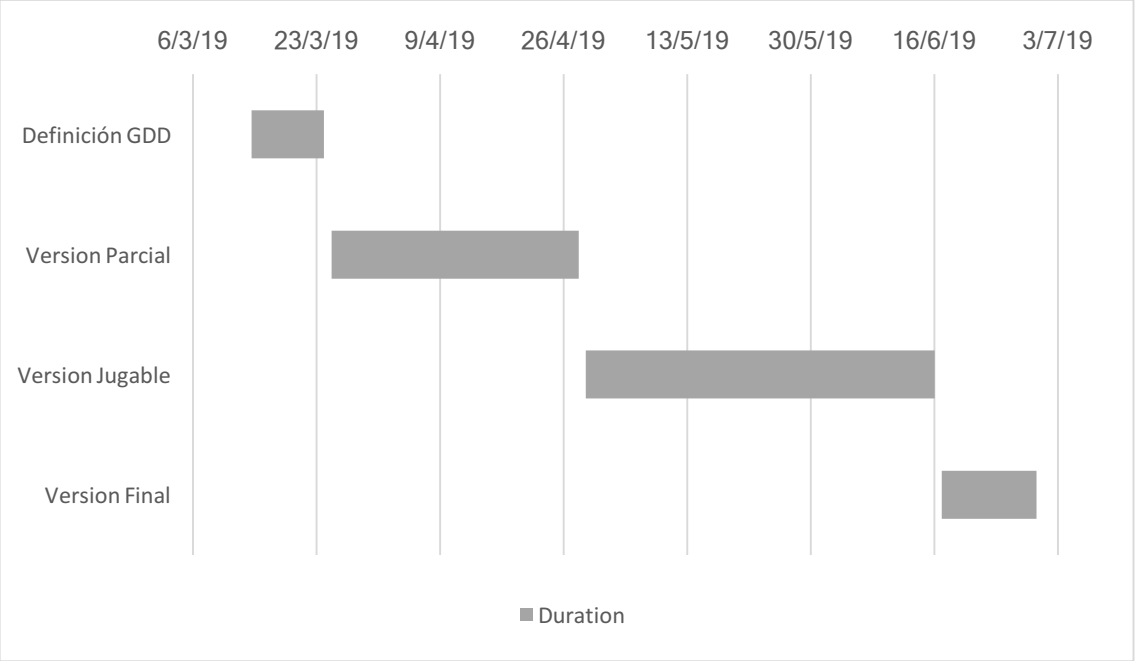

Ilustración 1 Diagrama Gantt planificación total

Vamos a mostrar un poco mejor cada uno de las dos entregas de la versión parcial y de la versión jugable, ya que dentro hay muchas tareas por realizar.

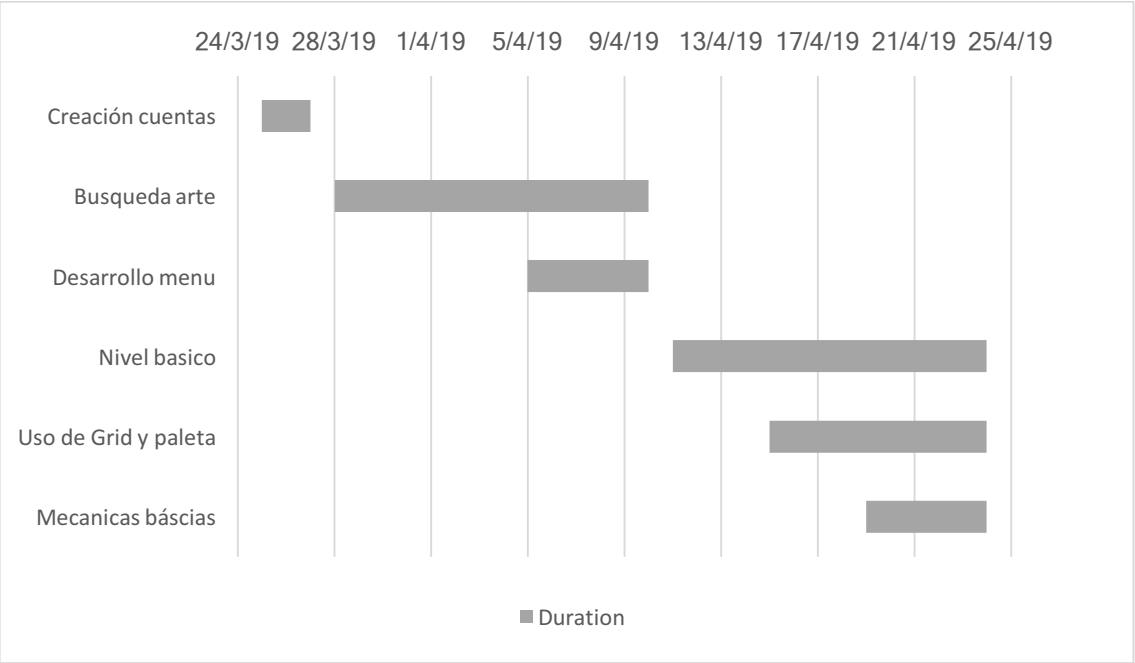

Ilustración 2 Diagrama Gantt planificación versión parcial

En esta primera entrega, se debe entregar una primera versión del juego, donde ya exista el principal flujo del juego. Esto significa que deben existir las pantallas de menú principal, créditos, menú de pausa y una pantalla de nivel básica.
Como se puede ver en el diagrama de Gantt, la primera semana se dedicará a crear el repositorio en “github” y vincular la cuenta de “UnityConnect”, unos requisitos indispensables para la primera entrega. Después, se empezará a buscar los *assets* necesarios para realizar el videojuego. En esa misma semana

se empezará a desarrollar las pantallas de la interfaz gráfica y el flujo básico del juego y posteriormente el nivel básico. Dado que el nivel se realiza con *Tiles* se deberá aprender a usar este sistema en *Unity* así que hemos dejado un poco de tiempo solo para este fin, para poder ir con más tranquilidad.
Una vez acabada esta fase, se empezará con las mecánicas básicas del jugador como morir, saltar y moverse, lo que será, cuando el juego esté terminado, el tutorial.

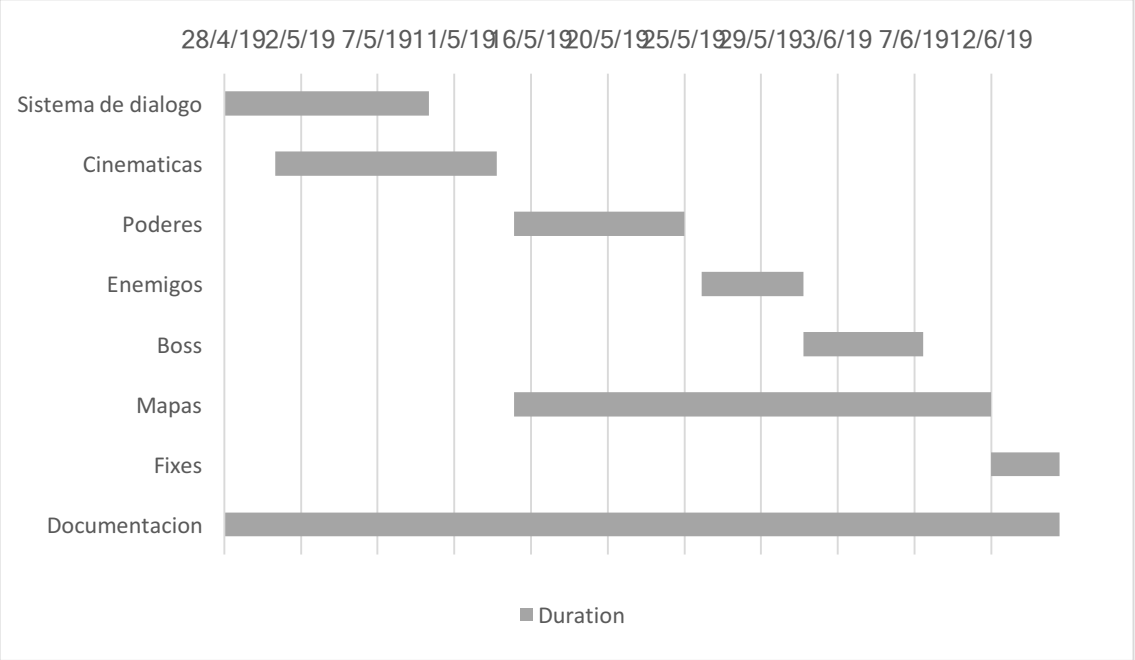

Ilustración 3 Diagrama Gantt planificación versión jugable

En esta versión se debe acabar el producto, creando todo aquello que se ha planificado. La creación de los mapas no es tan costoso como pone en el diagrama, sino que se alarga tanto en el tiempo porque se van a ir cambiando constantemente mientras se prueba los poderes y los enemigos, ya que como funcionen estos puede modificar el diseño de los mapas.
Mientras todo avanza, es recomendable ir haciendo la documentación en paralelo.

1.5 Breve sumario de productos obtenidos

El objetivo de este proyecto es obtener una pequeña versión jugable de un videojuego. Se obtendrá un fichero exe que pueda ser ejecutado en cualquier ordenador que cumpla los requisitos mínimos. Debe ser una versión que muestre las mecánicas que se quiera implementar en la versión final, aunque ya que es una demo, no tienen por qué estar todas las mecánicas que se quieren implementar realmente.

1.6 Breve descripción de los otros capítulos de la memoria

La estructura que se va a seguir para enseñar el trabajo realizado va a ser de la siguiente manera.

Primero se explicará un poco el género del videojuego, así como la situación actual de este tipo de videojuegos, para poner en contexto al lector y dar a entender el tipo de videojuego que se quiere desarrollar.
Posteriormente se dará una pequeña introducción al mundo del videojuego, el protagonista, los enemigos, la historia… También se plantearán los objetivos que tiene el jugador a lo largo de su aventura.
En este punto, empezamos a ver más afondo el videojuego en sí. Se mostrarán los *assets* que se han usado, las herramientas, el entorno de desarrollo, etc. Además, se presentará la estructura del videojuego, como se han creado los componentes y la relación entre ellos.
Una vez acabado con la parte técnica, se pasará a un ámbito más de diseño, donde se mostrará un nivel, en el cual se hará un *walkthrough* completo, indicando que es lo esperado por parte del jugador para avanzar.
Para finalizar se mostrarán los controles básicos y las conclusiones que se han sacado al desarrollar este proyecto.

2. Estado del arte

2.1 Plataformas 2D como género

Como ya se ha comentado anteriormente, el género de este videojuego es el de plataformas 2D. Durante muchos años, al inicio de la historia de los videojuegos, este género era el más común y popular sin duda alguna. Con la llegada de las nuevas tecnologías y consolas, llegaron nuevos géneros, que provocaron que el de plataformas 2D decayese en popularidad.
Sin embargo, en la última década se ha vuelto a ver un auge en la cantidad de videojuegos pertenecientes a este género principalmente debido al bajo coste de desarrollo que tienen este tipo de videojuegos frente a los 3D, que necesitan una inversión en el aspecto gráfico cada vez mayor para captar al público.
Por otro lado, la aparición de motores de videojuegos que facilitan a los desarrolladores *indie* a realizar sus juegos como, por ejemplo, el recién salido *Jump King*.
Otros juegos populares son *Celeste* y *Hollow Knight*, desarrollados para consolas. En el ámbito *smartphones*, podemos encontrar juegos como *Sonic The hedgehog 2 Classic*, “Mario Run” y *Meganoid*, juegos que muy famosos y populares anteriormente que se han traído de vuelta.

2.2 Principales características

La principal y la más común característica de este género es la necesidad de saltar para alcanzar lugares mientras se esquivan diversos obstáculos. Fallar uno de estos saltos suele suponer la muerte o la caída a niveles inferiores que ya has superado. Ambos “castigos” por haber fallado, supone volver a repetir fragmentos del juego que ya has superado.
Adicionalmente hay otras mecánicas que se han ido añadiendo a este género y se han repetido en diversos juegos, aunque no es común para todos ellos. Algunos ejemplos son el *dash,* enemigos, plataformas trampas, trampolines o propulsores, etc.
Mientras que anteriormente la plataforma objetivo eran las consolas de sobremesa, ahora estos videojuegos son más populares en las consolas portátiles, sobre todo con la llegada de la *Nintendo Switch*, el *Smartphone* o el ordenador, este último debido que es más fácil para los desarrolladores *indies* crear un videojuego en esta plataforma.

3. Definición del juego

3.1 Conceptualización

3.1.1 Breve descripción del juego

Shiane es un juego de plataformas 2D en el cual solo los reflejos y la habilidad del jugador permitirán a nuestro héroe avanzar a lo largo de las diferentes pantallas. Numerosos enemigos y obstáculos se interpondrán entre nuestro héroe y su objetivo: llegar a cima de la Torre de los Dioses para liberar a los 6 dioses de *Shiane*, que se encuentran sellados. Para ello deberá matar, esquivar los peligros que se encuentren, saltar y realizar piruetas imposibles. Para ello contará con ayuda de poderes de los dioses que le ayudarán y le facilitarán esta difícil tarea.

3.1.2 Tipo de interacción juego-jugador

El juego se desarrollará a lo largo de diversos niveles en 2D, con una perspectiva lateral ortogonal. Esto significa que en las pantallas no habrá una profundidad, sino que el jugador, los enemigos y las plataformas se encontrarán siempre en el mismo plano. Sin embargo, sí que existirá un *background*, haciendo uso del efecto *parallax scrolling*, en aquellas pantallas que sucedan al aire libre. Con diversas capas (2 o 3, dependiendo del nivel que estemos jugando) y asignando a cada una diferentes velocidades de *scroll*, dotará al juego de la sensación de que el jugador avanza a través del nivel.
La cámara seguirá constantemente al jugador, dejando a este siempre en el centro. De esta forma el jugador podrá ver siempre el problema que tiene que superar inminentemente. Aunque la cámara se comporta igual tanto en el eje horizontal como el vertical, este último tiene una restricción extra: nunca bajará de un umbral establecido.

Nuestro personaje tendrá diversas habilidades, las cuales irá adquiriendo a medida que supera niveles y mata enemigos. Además de andar y saltar, nuestro personaje puede utilizar también puede realizar un *dash*[1] de corta distancia, que le permitirá llegar más lejos o esquivar los ataques enemigos. Este *dash* solo puede realizarse dos veces por cada impulso o salto que realice. Es decir, debe apoyarse en el suelo o en un enemigo para poder volver a usarlo. Adicionalmente, realizando el *dash* sobre un enemigo, lo matará. El jugador tendrá otro ataque más: un potente proyectil de fuego que tiene un gran alcance y un pequeño *hitbox*[2].
Otra habilidad que permitirá sobrevivir a nuestro héroe es la de reflejar los proyectiles. Durante medio segundo, el jugador bloqueará todos los proyectiles que tenga alrededor, bloqueando el daño que se le fuese a causar, aunque no podrá volver a usar la habilidad en un tiempo.

---

[1] Movimiento rápido, casi como si se tele transportase del punto inicial al final.
[2] En un juego, espacio de colisión que tiene un objeto o personaje. A mayor tamaño de *hitbox*, mayor facilidad de ser golpeado

Por otra parte, habrá diversas mecánicas para matar a nuestro héroe y hacer la vida imposible al jugador. Plataformas que se mueven o que se caen, pinchos, enemigos que dispararán flechas o que darán espadazos cuando el héroe se acerque… La muerte es algo constante en este juego, y el volver a intentarlo repetidas veces la única forma de superarlo. A diferencia de otros juegos, no existen *checkpoints* en los cuales el jugador reaparece al morir, sino que cada vez que muera, tendrá que empezar la pantalla de nuevo y volver a pasarse todo. Los enemigos no volverán a reaparecer una vez los hayas derrotado, ni los *bosses*[3] finales recuperarán su vida. El objetivo de esta mecánica es hacer que pasarse el juego no sea complicado o imposible, pero si el hacerlo en el menor tiempo y con el menor número de muertes posible.

Como ya hemos comentado, existirán diversos *bosses* que el jugador deberá derrotar. Estos tendrán características propias, como ataques nuevos (cargas, salva de flechas…) y formas de defenderse (por ejemplo, un escudo similar al del jugador).

3.2 Ambientación

3.2.1 Ambientación e historia

Shiane es un mundo de fantasía, ambientado en una época parecida a la medieval de nuestro mundo.
La única religión de esta tierra está compuesta por 6 dioses, cada uno de los cuales representa un elemento: agua, fuego, viento, tierra, luz y oscuridad. Cuenta la leyenda que hace miles de años el dios de la oscuridad inició una guerra que, en menos de un día, diezmó a la población y redujo a cenizas los castillos y ciudades de los reinos. Para evitar que los humanos desapareciesen de la faz de la tierra, el dios de la luz se enfrentó a él en una batalla sin fin, que dura hasta el día y de hoy y que durará hasta el fin de los tiempos. Mientras tantos, los otros 4 dioses fueron encomendados a mantener el orden en el mundo, velando por el bienestar de todos los seres vivos.
Los humanos empezaron a rezar a los dioses y a comunicarse con ellos. Aprendieron que, con las oraciones y las ofrendas adecuadas, estos respondían haciéndoles favores y atendiendo sus necesidades. Es lo que ahora se conoce como la magia. Gracias a esto, los hombres aprendieron a vivir sin tener que presenciar hambrunas ni epidemias. Así, durante muchos años, el mundo vivió en una paz, que solo se veía rota por pequeñas guerras y escaramuzas de nobles que deseaban las tierras de los vecinos, aunque no necesitasen los recursos que estos proporcionaban, únicamente por ansias de poder.
Sin embargo, hace 75 años el dios de la oscuridad contestó por primera vez a aquellos que le rendían culto. Este, entregó parte de su poder a diversos altos cargos de esta iglesia. El poder de este dios es mucho mayor que el de los demás dioses, excluyendo al dios de la luz. 3 años más tarde, los devotos de la oscuridad sellaron a los dioses del agua, fuego, viento, tierra y luz. Cuando estos

---

[3] Enemigos más fuertes que los demás, normalmente situados al final de la pantalla o en la suya propia

se dieron cuenta ya era demasiado tarde. El dios de la oscuridad había ganado, pero no hay oscuridad sin luz. Cuando los otros 5 dioses fueron sellados, él también desapareció con ellos. Con los 6 dioses, desapareció también la magia y la prosperidad de Shiane.
72 años más tarde, las hambrunas y las guerras por las tierras que proporcionan más recursos estaban a la orden del día. La iglesia de la oscuridad mantuvo parte del poder que se le otorgó, destruyó la religión que rinde culto a los 6 dioses y acaparó el poder. Sin embargo, el poder del sello que retiene a los dioses y el poder que tenían los devotos de la oscuridad se ha debilitado y es vulnerable a ser roto y los seis dioses son capaces de comunicarse con los algunos humanos de nuevo.
Es aquí donde empieza la lucha de nuestro héroe para romper el sello y liberar a los dioses. Deberá luchar e infiltrarse en la iglesia de la oscuridad para debilitar el poder de estos. Los 6 dioses se le presentarán, proporcionándole conocimiento y poderes nuevos que le permitirá llevar a cabo su misión.

3.2.2 Personajes y elementos

- **Tyson:** Él es nuestro protagonista. Es un nombre de 23 años, de estatura media y más bien flaco, como la mayoría de habitantes actuales de Shiane. Debido a la hambruna que acecha el país, se ha pasado la vida robando oro y comida, oculto en las sombras. Es rápido y listo, siempre actuando casi instintivamente, pensando exclusivamente en su propia supervivencia y en la de los pocos seres queridos que le quedan. Viste siempre con ropa ajustada, para que no impida su movimiento, y oscura, para que le oculte en la oscuridad. Tiene un gran sentido para comunicarse con los dioses y, por lo tanto, una afinidad muy elevada a la magia, pero los dioses y la magia hace tiempo de desapareció para nunca volver. O eso piensa todo el mundo. El dios de la luz se le presentó un día para pedirle que les liberase. A partir de entonces los demás dioses se le presentaron también, incluyendo el dios de la oscuridad. Aunque al principio era reacio, al final acepta su destino y salva al mundo de la situación a la que está sometida actualmente.

- **Hikari:** Es el dios de la luz. Es estricto y directo, con gran un gran honor que le impide mentir. Se encariñó con la humanidad y debido a eso inicio la guerra sin fin contra el dios de la oscuridad. Inmerso totalmente en la guerra, no ha podido comunicarse con los humanos en mucho tiempo, y muchos de estos han dejado de creer y de rezarle. Esto ha provocado que su poder mengue, permitiendo que se vea superado por el dios de la oscuridad. Es el primer dios que se le presenta a Tyson, y su guía en el camino. Convence a Tyson para que los libere y, posteriormente poder matar al dios de la oscuridad: este ha sido desde hace mucho tiempo el único propósito por el cual ha vivido.

- **Kurayami:** Es el dios de la oscuridad. Es mentiroso e impulsivo. Según la religión principal, solo ansia poder y siente envidia de los otros 5 dioses, representa los pecados de los humanos. Se presenta varias veces a Tyson, intentándolo convencer de que no lo mate, que no siga los consejos de Hikari.

- **Hi:** Es la diosa del fuego. Es impulsiva y peligrosa, fácil de provocar su ira. El poder que otorga a los humanos es exclusivamente destructivo. Aunque muchos humanos le temen, a diferencia de Kurayami no busca poder ni intenta engañar a los humanos, ni tampoco los ha dañado nunca, no por voluntad propia. Es el dios más famoso, usado en la mayoría de guerras mediante las oraciones. Otorga a Tyson el ataque fogonazo.

- **Chi:** Es la diosa de la tierra. Es obstinada y tranquila. Siempre piensa las cosas antes de actuar, tal vez demasiado. Los poderes que otorgan son muy versátiles y ha permito a los humanos construir edificios que impensables para la época que habitan. De la misma forma, sus oraciones sirven para hacer que los cultivos crezcan y den muchos frutos. Tyson opina que habla demasiado lento y le aburre. Otorga a Tyson el poder de desviar proyectiles.

- **Mizu:** Es la diosa del agua. Es bastante ingenua y fácil de manipular. Tiene poca confianza en sí misma y siempre pregunta a los otros dioses antes de actuar. Le gusta ayudar y ver crecer a los humanos. Sus oraciones son curativas y de manipulación de la vida. Otorga a Tyson el poder de refrescar los *dashes* y las demás habilidades.

- **Kaze:** Es la diosa del viento. Actúa como si nada de su alrededor le importase. Muchas de las oraciones son ignoradas, haciendo caso solo cuando el fin de dichas oraciones le puede otorgar diversión o, en su defecto, por curiosidad. Otorga a Tyson, más por diversión y curiosidad, el poder del *dashes*.

- **Riole:** Es el líder del culto del dios de la oscuridad. Ha vivido durante más de 200 años gracias al poder de Kurayami. Durante los primeros 100 años vivió oculto, acumulando poder y adeptos, escondiéndose de los demás dioses mientras esperaba a que Hikari se debilitase. Después de vivir tantos años y ver como su poder se está extinguiendo, teme a la muerte ahora más que nunca. Hará lo posible por huir de Tyson y dejar que sus secuaces lo maten, sin arriesgar su propia vida.

3.2.3 Interacción entre los actores del juego

A lo largo del desarrollo del juego, los dioses hablaran con Tyson, ya sea para contarle trozos de historia, darle consejos o para explicarles los poderes que estos son capaces de otorgarle y que aprenda a controlarlo a usarlos.
Estas interacciones se realizan mediante un cuadro de texto semitranslúcido que aparece en la parte inferior de la pantalla, cubriendo casi el total de la misma. El texto aparecerá poco a poco, y el jugador podrá clicar sobre un botón para ignorarlo y pasar al siguiente. Estas interacciones son más bien monólogos, donde solo intervienen los dioses u otros personajes, pero nunca nuestro héroe.

La primera interacción ocurre con Hikari en el tutorial y se desarrolla de la siguiente forma:

- Hola Tyson. Mi nombre es Hikari. Supongo que has oído hablar de mí, soy el dios de Luz. Oh, no te preocupes. Estoy acostumbrado a que nadie me crea y se piensen que estoy loco. Aunque hace muchísimos años que no me encontraba en esta situación. Hacía mucho tiempo que no hablaba con nadie... Miles y miles de años, desde que empecé la guerra con Kurayami... Aunque supongo que ya conoces la historia. El dios de la luz y de la oscuridad se enfrentan en una batalla que durará hasta el fin de los tiempos.... Perdona, estoy desvariando un poco. No he venido a contarte la historia de nuestro mundo, sino a pedirte ayuda. Sabes bien que los 6 dieses de esta tierra han estado sellados durante los últimos 72 años. Y por fin el poder que Kurayami dejó a sus súbditos se está debilitando debido a que el mismo dios de la oscuridad está perdiendo poder. Aunque a nosotros también nos quedan pocos años de vida, sin vuestras oraciones, todos los dioses están destinados a ser olvidados y a morir. Es por eso que necesito que pongas fin al culto de Kurayami y nos liberes cuanto antes. Nosotros nos ocuparemos de matar de una vez por todas a Kurayami. Oh, estas demasiado ocupado, ¿verdad? Pero, ¿que hay más importante que la supervivencia del mundo entero? ¿Es que acaso no me crees? No te preocupes te demostraré que realmente soy un dios. Antes de que acabe el día habrás cambiado de opinión. Bueno, de momento te recomiendo huir de aquí. Muévete con las teclas "a", "s", "d" y "w", salta con la tecla "espacio".

Vemos que esta conversación sirve para poner un poco de contesto al jugador, pero también para indicarle como moverse por el mundo. Posteriormente, en el mismo tutorial se usan estas conversaciones para enseñarle a realizar el *dash,* en este caso de mano de Kaze.

Aunque no están en esta demo que se presenta, muchas otras conversaciones de los demás dioses y, sobre todo de los *bosses* a los que se enfrentan, aparecerán en el juego. Sin ir más lejos, Kurayami también se le presentará, arrepentido por lo que ha hecho, diciéndole que él es el culpable de la miseria que hay en el mundo. Pero le suplica constantemente que no haga lo que Hikari le dice, si después de liberar a los dioses Kurayami es asesinado por los otros 6 dioses, el dios de la luz también perecerá, no puede existir uno sin el otro:

- El último de los dioses que sellamos fue Hikari. En el mismo momento que lo envié a exilio, mi poder se desvaneció. Todo lo que había conseguido, todos los poderes que había repartido en la tierra, todo lo que llevaba años construyendo se esfumó. La catedral donde hicimos el conjuro no resistió y se desmoronó, todos los que estaban allí presentes, mantenidos por más de 100 años vivos gracias a mí, murieron. Habían sobrevivido a enfermedades, “asesinatos” y experimentos sangrientos, pero murieron…No hasta que yo desapareí. Riole ya no es inmortal tampoco, está envejeciendo rápidamente, está perdiendo mi poder, está agonizando, noto que aún está vivo, aunque dudo que dure más de un par de años… Si me matas, Tyson, pasará lo mismo con Hikari, lo matarás, lo matarás a él, a los demás dioses y a la humanidad…

3.3 Objetivos planteados al jugador

Los objetivos planteados al jugador se han explicado de forma indirecta al explicar la historia y contextualizar el juego. Debe salvar a los dioses que están sellados. Para ello deberá avanzar por diversos niveles con diferentes enemigos y objetos que supondrán un obstáculo.
Aunque la idea principal es bastante sencilla, habrá diversos contadores y retos que se le presentan secundariamente y totalmente optativos. Por ejemplo, el número de muertes, el tiempo que tarda o la cantidad de enemigos que asesina pueden otorgarle "medallas". De esta forma, una vez pasado el juego siempre se puede re-jugar y conseguir todo aquello que no se ha podido hacer antes.
Existen muchos otros retos ocultos que el jugador deberá descubrir a medida que juega.

3.4 Concept Art

Aunque se presentarán todos los recursos utilizados en otra sección[4], quiero mostrar aquí alguno de los elementos básicos para dar una idea de cómo son los *sprites* utilizados para desarrollar el videojuego.

La primera tira de *sprites* que podemos ver es el enemigo que más aparecerá en el juego, el arquero. Es un enemigo que ataca a distancia, disparándonos flechas. Un solo impacto de una de estas flechas es mortal y hará que nuestro jugador muera, a no ser que tenga el escudo activado. En cuanto tenga visión directa sobre nosotros, disparará sin pensárselo ni un segundo.

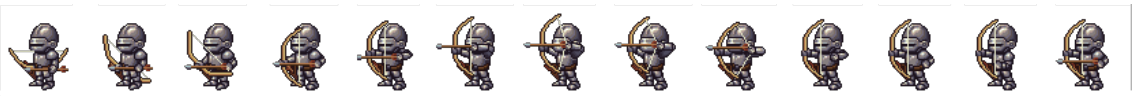

Ilustración 4 Enemigo arquero disparando

Aquí podemos verlo cargando y disparando la flecha[5].

Ilustración 5 Enemigo arquero andando

En esta segunda tira, podemos verlo en la animación de caminar, aunque casi ninguno de los arqueros que nos encontremos caminará, simplemente esperarán a que el jugador salga para dispararle. Otro tipo de enemigo, esta vez con un

---

[4] También se dirá de donde se han obtenido dichos elementos
[5] Podemos observar que la flecha no sale. Esto se debe a que es un objeto individual en *Unity* y por eso no sale en la tira aquí mostrada.

ataque cuerpo a cuerpo, tiene las mismas animaciones y casi los mismos *sprites* que el aquí mostrado. Simplemente se cambia la flecha y el arco por una lanza, pero el diseño es exactamente el mismo.

El jugador se ha sacado del mismo recurso, y podemos ver que es el mismo estilo y que usa las mismas animaciones, cambiando el personaje en sí, pero reaprovechando la idea.

Ilustración 6 Tyson, el protagonista, andando

Como ya hemos dicho, Tyson puede ponerse un escudo para protegerse de los ataques de los enemigos. En las dos siguientes imágenes podemos ver como es el escudo y también como se aplica sobre nuestro protagonista. El escudo se utiliza también para alguno de los *bosses.*

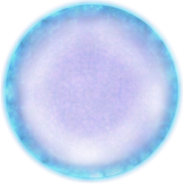

Ilustración 7 Escudo

Ilustración 8 Tyson con escudo

Dejamos de lado el concept art de los caracteres para mostrar como son los escenarios donde sucede la historia, es decir, los niveles.

Como ya hemos dicho antes, el mundo está lleno de elementos que, en caso de que nuestro protagonista lo toque, morirá. Estos son dos ejemplos de este tipo de objetos:

Ilustración 9 Engranajes

Ilustración 10 Pinchos

Por último, un par de imágenes del terreno y de las plataformas, así como de los elementos decorativos que hay en las diversas pantallas.

Ilustración 11 Terreno niveles externos

Ilustración 12 Terreno niveles internos

4. Diseño técnico

4.1 Entorno

A la hora de seleccionar que entorno de desarrollo se va a usar para llevar a cabo este videojuego, se plantearon diversas opciones y finalmente se decidió por una de ellas. En esta sección se explicará porque razón se decidió usar *Unity*, y cuáles eran las principales alternativas que se tuvieron en cuenta.

Actualmente hay decenas de motores de videojuegos capaces de desarrollar grandes videojuegos, con una gran comunidad detrás, tutoriales y recursos que permitan a muchos desarrolladores, ya sea grupos pequeños o incluso una única persona, crear su videojuego. De entre todos, los 4 motores que se tuvieron en cuenta en esta ocasión son: *Unity, Unreal Engine, Game Maker Studio 2* y *Cocos2d.*

El primero que se descartó fue *Unreal Engine*, a pesar de que este motor de videojuegos es extremadamente potente y se han realizado videojuegos que han vendido miles de copias en todo el mundo. Algunos de los juegos que hicieron que este motor[6] ganase publico fueron *Gears of Warsm Dishonored, Daylight* o *Mass Effect* .Muchos juegos que actualmente se encuentran entre los juegos más jugados están hechos con este motor: *PlayerUnknown's Battlegrounds, Draksiders III* o *Life is Strange* [1]. Si este motor de videojuegos es tan potente, ¿Por qué no se ha usado para desarrollar el juego en este trabajo final de carrera?
La potencia que este motor de videojuegos trae es un arma de doble filo. No es fácil de usar y su curva de aprendizaje es mucho más elevada que la de los demás motores de videojuegos que se han estudiado aquí. Por otra parte, el videojuego que se está desarrollando es un juego 2D, mientras que este motor está más enfocado a desarrollar videojuegos 3D, ya que la principal característica que diferencia a este motor de los demás es su capacidad para crear unos gráficos ultra realistas. Obviamente, esto no quiere decir que no pueda usarse este motor para este tipo de videojuegos, ya que también se han realizado algunos videojuegos 2D con este motor que han triunfado, como por ejemplo el aclamado *Octopath Traveler*.

GameMaker Studio 2 sí que está enfocado en realizar videojuegos en 2D pero tampoco ganó esta "batalla". El famoso juego de *Undertale* se ha desarrollado con este motor y tal vez gracias a esto es por lo que está en esta pequeña lista. Este motor de videojuegos es realmente fácil de usar, ya que permite realizar sencillos videojuegos sin la necesidad de programar y dispone de diversas herramientas que te permiten desarrollar todo el flujo sin necesidad de programas externos. Gracias a esto, mucha gente cuyo principal rol no es el de desarrollador, sino que son más bien artistas o diseñadores, han podido crear su videojuego.

---

[6] No están hechos con la versión actual del motor, sino con la anterior.

Sin embargo, esto tiene un coste bastante elevado, las posibilidades del juego se limitan bastante. A la hora de realizar acciones no tan "comunes" se tiene que usar un lenguaje propio (GML), el cual no está orientado a objetos. Sumando que no tengo experiencia en este motor de videojuego, no solo debería aprender a usar el motor de videojuego si no también el lenguaje único que usa.
Para finalizar, *GameMaker Studio 2* no es gratuito, aunque que dispone de una versión de prueba, tiene un precio de casi 100$.

Cocos2d también está focalizado a videojuegos 2D y no tiene ninguno de los problemas comentados anteriormente. Es gratis y *Open Source*. Usa C++ o JavaScript a la hora de desarrollar los scripts, dos lenguajes probados y en los cuales sí que tengo experiencia. También proporciona un entorno de desarrollo completo, donde puedes realizar casi todo sin necesidad de herramientas externas. Por si fuese poco, este motor está muy optimizado para realizar videojuegos basados en *Tiles*[7], que casualmente, es una de las características de nuestro videojuego.
La razón principal de haber rechazado este motor de videojuegos como el encargado de ayudarme a realizar el videojuego, es la falta de experiencia que tengo con él. Además, la comunidad, la documentación y los tutoriales disponibles no son lo suficiente elevados como poder aprender a usarlo en tan poco tiempo.

Así pues, ya como se había adelanta al principio de este capítulo, el motor gráfico usado en este trabajo es *Unity*. Hemos comentado por que no se han escogido los otros motores gráficos, pero *Unity* no se ha escogido por descarte. Lejos de eso, *Unity* ha demostrado ser un motor de videojuegos de lo más potente, tanto para desarrolladores individuales como para grandes grupos formados no solo de programadores.
Unity se puede descargar y usar gratis, siempre y cuando los beneficios obtenidos con los productos creados con el motor no sean superiores a 100.000 dólares anuales. Muchos de los recursos que proporciona la misma web oficial, ya sean creados por los desarrolladores del motor o por otros desarrolladores de videojuegos que ofrecen sus recursos a quien quiera cogerlo, son gratis y se pueden usar sin restricción alguna.
Por otra parte, la comunidad que hay detrás de este motor es enorme. Tiene un foro propio donde hay miles de usuarios y preguntas y, cualquier duda que tengas, probablemente ya haya sido resuelta antes. Si no es en este foro, es *stackoverflow* también hay muchas preguntas que tratan sobre problemas o dudas relacionadas con este motor. Además, la documentación de las funciones y clases propias de *Unity* es muy completa y con ejemplo. También tienen varios tutoriales que te permiten aprender de mano de los mismos creadores del motor. Sino son suficiente, en *YouTube* también hay miles de desarrolladores que han colgado sus conocimientos. Esto, para mí, es una de las principales razones por las cuales este motor es tan bueno, la comunidad que se ha creado alrededor.

---

[7] Los juegos basados en *Tiles* son aquellos en los cuales el área de juego consiste en pequeñas imágenes, normalmente cuadradas, donde cada una de ellas se llama *Tile*. El conjunto de ellos, conforma el mapa. Normalmente este juego son 2D o 2.5D.

Por último, otra razón es la experiencia que tengo. Tanto el lenguaje como el motor gráfico es algo conocido para mí. Ya he trabajado con ellos anteriormente en varias ocasiones, incluso antes de empezar este master. Esto es algo bastante importante, ya que el tiempo del que dispongo es bastante limitado debido a que trabajo a jornada completa y, el no tener que "perder tiempo" aprendiendo cosas básicas es bastante importante. Aún a pesar de haber usado *Unity,* he querido usar varios componentes nueves que he tenido que aprender a usarlos igualmente.

4.2 Requerimientos del entorno

Según la página oficial de *Unity*[8] los requerimientos mínimos son los siguientes:

- Windows 7 o mayor, versión de 64bit
- Ó MacOs 10.12
- GPU compatible con DX10
- CPU compatible con instrucciones de SSE2

También se puede usar Ubuntu, aunque no está oficialmente soportado. Para el desarrollo del TFM se ha utilizado tanto un Windows 10[9] y un MacOS 10.12.6.[10]

4.3 Herramientas

Además del programa *Unity* se han utilizado dos programas para la creación y manipulación de los *sprites* que se han utilizado en el desarrollo de este videojuego*:*

- *GIMP2*[11]*:.* Es totalmente gratuito y fácil de usar. Sirve para la manipulación de imágenes, como por ejemplo recortes, escalados, composición de imágenes… Permite trabajar en distintas capas, lo que facilita la reutilización de recurso y trabajan en ellos independientemente.

- *Inkscape*[12]*:* Es totalmente gratuito, y tal vez un poco más difícil de usar que el anterior, aunque también es verdad que su objetivo es completamente distinto. Este programa se usa para tratar de forma vectorial las imágenes, lo que permite el re escalado sin perder definición. Se usa sobre todo para la creación de logos, iconos, etc…

Ambos programas se pueden usar tanto en Windows como en MacOs y no requiere unos requisitos demasiado altos, aunque depende más del uso que se le vaya a dar.

---

[8] https://unity3d.com/es/unity/system-requirements
[9] Version de 64bits. Procesador AMD Ryzen 5 1500X. 8.00GB de RAM. Radeon RX 570 series.
[10] Procesador Intel Core i7 2,2GHz. 16GB de RAM. Intel Iris Pro 1536 MB.
[11] Se puede descargar gratuitamente de la página oficial https://www.gimp.org/
[12] Se puede descargar gratuitamente de la página oficial https://inkscape.org/

4.4 Assets

En este apartado se van a listar y explicar cómo se han realizado o, mejor dicho, de donde se han obtenido todos los *assets* que se han utilizado. Dado que hay muchos y ponerlos todos podría ocupar una docena de hojas se va a intentar disminuir al mínimo el espacio ocupado, intentando poner todos los ejemplos. Algunos se de los *assets* se comentarán, pero no se pondrá la imagen. Este caso, normalmente, será porque se ha mostrado uno muy similar y no merece la pena volver a mostrarlo.
Se expondrán los *assets* por temas, tratando primero los personajes, los objetos que hay, el terreno y, finalmente, las decoraciones de los escenarios.

4.4.1 Protagonista

El primer *sprite* que mostramos es, obviamente, nuestro protagonista, Tyson. Es el personaje principal y el que controlara el jugador, así que es lógico que sea el primer que veamos.

Ilustración 13 Todos los sprites de Tyson

Podemos ver tres tiras de *sprites,* aunque en realidad son 4. Más adelante explicaremos como se usa cada una de ellas.

Al igual que todos los demás personajes que aparecen, estas imágenes se han obtenido de la página http://gaurav.munjal.us/Universal-LPC-Spritesheet-Character-Generator. Tiene muchos aspectos a modificar, dando lugar a la creación de diversos protagonistas. Aunque útil, le faltan bastantes ropajes, ya que, según mi punto de vista, tiene muy pocos conjuntos que elegir.

La primera de todas se usa a la hora de saltar. Para ser más específico, se usa cuando el jugador está en el aire. Mientras que la velocidad sobre el eje Y sea positiva, lo cual significa que está subiendo, se activará la animación que está compuesta por las 4 primeras imágenes. Si el jugador subiese constantemente la imagen número 5 sería que sustituiría a “la imagen por defecto” del personaje. Cuando la velocidad sobre el eje Y sea negativo, se utilizará la imagen número 6, que da la sensación de que está cayendo. Si el jugador no salta, si no que se

cae de una plataforma, las últimas imágenes se sucederán, siendo la última de todas la 6, que será visible hasta que toque el suelo.
Esta primera tira también se usa para la animación de *idle* del personaje. Concretamente, las dos primeras imágenes. Mientras el jugador este quieto, estos dos *sprites* se sucederán constantemente.
La siguiente tira compone la animación de andar de nuestro personaje. Solo se aplicará cuando se mueva horizontalmente y no esté en el aire.
Por último, la tira final corresponde a la muerte del jugador. Hay que tener en cuenta que esto solo sucede en la cinemática, ya que el jugador es "inmortal" y cuando este muere, simplemente vuelve a reaparecer al inicio del mapa.

Como podemos observar, todos los *sprites* miran hacia la izquierda, pero en el juego a veces nuestro protagonista mira a la derecha. Para hacer esto, obviamente no se han creado dos tipos de animaciones. Simplemente se ha cambiado por código la propiedad específica del *SpriteRenderer* que provoca que la imagen se gire, y se mantiene, aunque una animación esté en curso.

4.4.2 Arquero enemigo

Este es el enemigo más recurrente del juego. Ataca a distancia, lo que hace que su localización sea importante, pudiendo molestar mucho al jugador avanzar. Además, su cadencia de disparo es bastante elevada, puede volver a disparar mucho antes de que el escudo y/o la bola de fuego se hayan recargado.

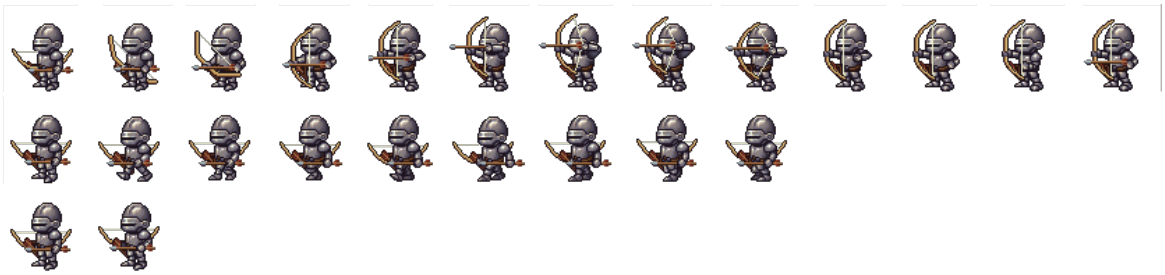

Ilustración 14 Arquero completo

Como ya hemos dicho antes, igual que el jugador, estos *sprites* se han obtenido de la página web http://gaurav.munjal.us/Universal-LPC-Spritesheet-Character-Generator. Como podemos empezar a ver, la tira de caminar usa exactamente el mismo estilo y las mismas posiciones que la tira de andar de Tyson, simplemente se ha cambiado la vestimenta de calle por una armadura y se le ha añadido la flecha y el arco que tiene este enemigo.
Hay que comentar que el arco y la flecha solo están disponibles por defecto en la tira de disparar, pero que no están presentes en todo el resto. Sin embargo, la flecha y el arco sí que están disponibles en la web. Gracias a GIMP y a estos dos materiales se han podido añadir en la posición deseada para que parezca que el enemigo se mueve y hace la animación de *idle* correctamente.
Como podemos suponer, la primera tira corresponde con la animación de disparar del arquero, la segunda, con la animación de andar y la última la animación de *idle.* Creo que no merece la pena explicar cómo y cuándo se decide cuando se ejecuta cada una de ellas ya que es implícito.

4.4.3 Lancero enemigo

Este es la contra parte del arquero. Ataca cuerpo a cuerpo y se mueve, evitando caerse de las plataformas y de estrellarse con los objetos como los pinchos.

Ilustración 15 Lancero completo

Sacado de la misma página que el arquero enemigo, y siguiendo casi casi el mismo proceso. La lanza se ha tenido que añadir posteriormente ya que no viene incluida en alguna de las tiras que podemos ver.

La primera tira corresponde con la animación de *idle.* La segunda, de atacar cuerpo a cuerpo. La tercera, con la animación de andar. La última a la animación de morir.

4.4.4 Riole, *boss* final

El último de los personajes que nos queda por mostrar es el *boss* final, Riole. Se ha obtenido del mismo sitio que los tres anteriores, así que no se va a comentar nada nuevo sobre él.

Ilustración 16 Riole

4.4.5 Objetos y elementos mágicos

Una vez mostrados todos los personajes que aparecen en el juego, vamos a mostrar los objetos que aparecen.

Por escoger algún orden, empezamos de nuevo con aquellos objetos que tienen una relación con nuestro protagonista y luego procederemos con las armas de los enemigos.

Ilustración 17 Bola de fuego

La bola de fuego que nuestro jugador lanza a los enemigos o a las flechas para destrozarlas, es esta que acabamos de ver aquí arriba. Esta obtenida de la web *opengameart,* concretamente del siguiente enlace: https://opengameart.org/content/fireball-vector

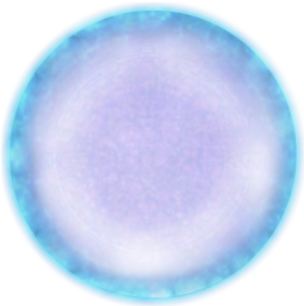

Ilustración 18 Escudo

El escudo que se pone nuestro jugador para protegerse de las flechas como el que se pone Riole en la batalla final es la ilustración 14. Igual que la bola de fuego que hemos podido ver antes, este *sprite* se ha obtenido de la web *opengameart*. https://opengameart.org/content/shield-aura-effect

Ilustración 19 Diamante de fresco

Esta gema es la que permite a nuestro protagonista pueda volver a realizar el *dash*, incluso aunque el jugador no haya tocado el suelo aún. Igual que la bola de fuego que hemos podido ver antes, este *sprite* se ha obtenido de la web de *opengameart*. https://opengameart.org/content/2d-platformer-objects-pack

Ilustración 20 Flecha

Como ya hemos comentado en el apartado de *concept art* (3.4 Concept art), las flechas que disparan los enemigos son un objeto completamente aparte. Este es el s*prite* que se usa para la creación de dicho elemento. Se ha obtenido del LPC (http://gaurav.munjal.us/Universal-LPC-Spritesheet-Character-Generator).

Ilustración 21 Flecha completar *sprites*

Esta flecha es la que hemos necesitado para completar los *sprites* del arquero enemigo. Se ha obtenido también de la web del LPC. No hay objeto por si solo con este *sprite.*

Ilustración 22 Arco para completar *sprites*

Exactamente igual que la ilustración 17. Solo se ha usado para poder completar los demás *sprites.*

Ilustración 23 Lanza para completar *sprites*

Exactamente igual que el arco y la flecha de la ilustración 18 y 17, respectivamente.

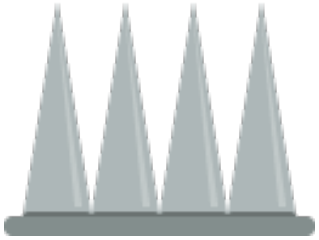

Ilustración 24 Pinchos

Como ya hemos mostrado anteriormente, este *sprite* corresponde con los pinchos que abundan en los niveles que Tyson debe avanzar. Tocarlos supondrá la muerte de nuestro personaje, que deberá volver a empezar. Se ha sacado del paquete que se puede descargar en la siguiente web. https://opengameart.org/content/2d-platformer-objects-pack

Ilustración 25 Engranajes

De nuevo aparece este *sprite* que corresponde a otro tipo de objetos que matarán a Tyson si los toca. En este caso, no están pegados en el suelo, si no que pueden encontrarse flotando. Además, pueden o no moverse en el aire, con distintas velocidades y con un tiempo de reposo cuando llegan a uno de los extremos antes de volver a empezar a moverse. En los niveles que se han entregado solo tienen dos puntos entre los cuales bascula, pero están preparados para tener cuantos se quiera. Se ha sacado del paquete que se puede descargar en la siguiente web. https://opengameart.org/content/2d-platformer-objects-pack

4.4.6 Terreno

Hasta aquí se ha visto todos los elementos dinámicos del videojuego, los caracteres, los enemigos, las flechas… Pero todo esto se debe mantener sobre una base, un suelo o terreno. Como ya se ha comentado antes, se pueden sacar miles de combinaciones con todos los *tiles* o *sprites* que se han importado, pero todos se comportan de la misma forma *in-game*. Por eso se va a mostrar uno de cada tipo.

Ilustración 26 Terreno básico exterior

La ilustración 26 es un conjunto de tres *sprites* que nos permiten formar plataformas tan largas como queramos, con si de una imagen única se tratase. La única condición es que el tamaño mínimo es de dos *sprites* y si se quisiese una, se debería usar el que vamos a mostrar a continuación en vez de la combinación de estos.

Ilustración 27 Terreno básico de un solo *sprite*

Con estos elementos ya se pueden crear la mayoría de las plataformas que aparecen en el nivel tutorial. Para diferenciar el terreno básico con las plataformas que se mantienen en el aire se han añadido las siguientes imágenes, aunque perfectamente se podría haber utilizado estas mismas.

Ilustración 28 Plataformas flotantes tutorial

Estas son las principales plataformas que aparecen en este videojuego. Sin embargo, tienen su homologo correspondiente en los niveles que suceden en interior de la Torre de los Dioses. Por ejemplo, la ilustración 26 se convierte en la siguiente:

Ilustración 29 Terreno básico de la Torre de los Dioses

Otros de las plataformas que se repite a menudo en el juego son las plataformas que desaparecen una vez que el protagonista las ha pisado, para luego volver a parecer.

Ilustración 30 Plataformas trampa

Podemos ver que no ocupan el mismo tamaño de altura que las demás plataformas, sino que son la mitad de altas. Se decidió usar estas plataformas para que el usuario pueda reconocerlas una vez sepa que estas plataformas no son del todo fiables, es decir, una vez haya sido “traicionado” por una de ellas.

Ilustración 31 Plataformas especiales

Esta es el último *sprite* que vamos a mostrar en este apartado. Estas plataformas también son especiales. Pueden ser atravesadas desde abajo, ya sea saltando o con un dash, pero no desde la parte superior.

Estos y todos los otros *sprites* que se han utilizado para formar el terreno se han obtenidos de la página web de *Kenney.* El pack en concreto es el siguiente: https://www.kenney.nl/assets/platformer-art-pixel-redux

4.4.7 *Background*

El fondo de *Unity* es bastante aburrido así que se ha sustituido por un efecto de *parallax scrolling*. Hay que tener en cuenta que esto solo está activado en aquellos niveles externos, y no se ve cuando el nivel sucede dentro de la Torre de los Dioses.

Este efecto está constituido por cuatro capas de imágenes que, cuando se ponen una delante de otra y se mueven con una velocidad adecuada, provoca un efecto de profundidad que realmente no existe.

Ilustración 32 Cielo *background*

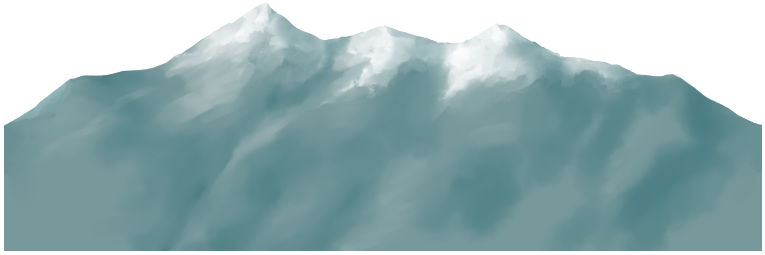

Ilustración 33 Montañas en primera capa

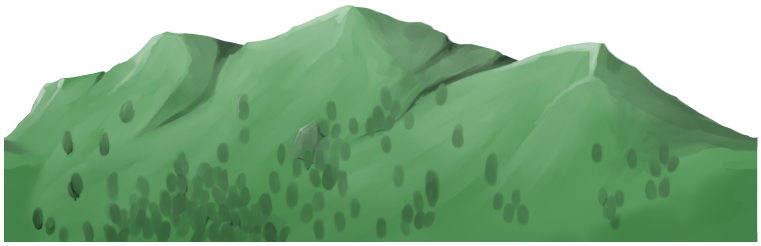

Ilustración 34 Montañas en segunda capa

Ilustración 35 Nubes con distinto tamaño y forma

Estas imágenes se han obtenido desde la propia tienda de *Unity:* https://assetstore.unity.com/packages/2d/environments/painted-hq-2d-forest-medieval-background-97738
Cuando la acción sucede en un escenario interno, estos fondos son sustituidos por los elementos decorativos que vamos a ver en el siguiente apartado.

4.4.8 Elementos decorativos

Los elementos que se van a mostrar aquí no son importantes en el desarrollo del juego, pero si da un poco de vida a los escenarios que, sin ellos, sería muy aburrido y estaría vacío. Todos ellos vienen en el pack de *Kenney.*
Se dividen en dos apartados, aquellos que se usan en escenarios al aire libre y aquellos que se usan en escenarios interiores.

Empezamos primero por aquellos que se utilizar en los niveles exteriores.

Ilustración 36 Varios elementos decorativos exteriores

Ilustración 37 Conjunto de *sprites* que forman un árbol

Y los siguientes son exclusivamente interiores.

Ilustración 38 Elementos decorativos interiores

Ilustración 39 *Background* interior

La ilustración 39 muestra el fondo que se ve cuando el nivel sucede en el interior de la torre de los dioses. Para que no sea tan repetitivo y vacío, se añaden los elementos que hemos comentado y mostrado en la ilustración 36. Sin embargo, de vez en cuando, también se sustituye este *sprite* por uno de los que podemos ver a continuación.

Ilustración 40 *Alt background* interior

4.4.9 Sonidos

Sonido 1 Sonido de las pisadas del protagonista

Es el sonido que el protagonista hace al caminar. Se han hecho mediante eventos asociados en la animación. Se explicará por encima en el siguiente apartado. El sonido se ha obtenido de https://opengameart.org/content/sfx-the-ultimate-2017-8-bit-mini-pack

Sonido 2 Sonido del salto del protagonista

Es el sonido que el protagonista hace al saltar, ya sea cuando salta sobre un enemigo, una pared vertical o un salto desde el suelo. El sonido se ha obtenido de https://opengameart.org/content/sfx-the-ultimate-2017-8-bit-mini-pack

Sonido 3 Sonido al hacer dash

Es el sonido que el protagonista hace al hacer un *dash*. El sonido se ha obtenido de https://opengameart.org/content/superpowers-assets-sound-effects

Sonido 4 Sonido al disparar la bola de fuego

Es el sonido que el protagonista hace al disparar la bola de fuego. El sonido se ha obtenido de https://opengameart.org/content/spell-4-fire

Sonido 5 Sonido clic menú

Siempre que se cicle sobre un elemento del menú, se escuchará este sonido. Se ha obtenido de https://opengameart.org/content/superpowers-assets-sound-effects

Sonido 6 Sonido victoria

Sonido al pasarse uno de los niveles del videojuego. Se ha obtenido de https://opengameart.org/content/superpowers-assets-sound-effects

Sonido 7 Sonido muerte jugador

Sonido que podemos escuchar cuando nuestro protagonista muere. El sonido se ha obtenido de https://opengameart.org/content/sfx-the-ultimate-2017-8-bit-mini-pack

Sonido 8 Sonido impacto flecha contra escudo

Sonido que podemos escuchar cuando una flecha impacta sobre el escudo mágico. El sonido se ha obtenido de https://opengameart.org/content/sfx-the-ultimate-2017-8-bit-mini-pack

Sonido 9 Sonido disparo flecha

Sonido de las flechas al ser disparadas por los arqueros enemigos. Se ha obtenido de https://opengameart.org/content/arrow-hit-twang

Sonido 10 Sonido al atravesar un diamante

Sonido cuando alcanzamos un diamante que nos refresca los *dashes* hechos. Se ha obtenido de https://opengameart.org/content/superpowers-assets-sound-effects

4.5 Arquitectura del juego

Dentro de un proyecto de *Unity,* la arquitectura está bastante sujeta a lo que el motor gráfico nos impone. La libertad que existe en muchos otros proyectos, como por ejemplo el desarrollo web, donde existen diversas estrategias para abordar el problema a realizar, está limitada en este caso. No es algo único de *Unity* o de cualquier otro motor gráfico, siempre que se trabaje en cualquier *framework,* se debe seguir su arquitectura, sino el problema es mucho más difícil de resolver.

La intención de *Unity* es que los proyectos se organicen, no orientados a objetos como se hace en la mayoría de los demás proyectos de *software,* sino orientados a componentes. La idea de los componentes es solucionar un pequeño problema, dividiendo el total en otros más pequeños [1].

Por encima de todos y cada uno de los componentes u objetos hay uno, acesible desde todos los lados, dado que es estático y *singleton*[13]. Este es el encargado de asegurar que el flujo del juego funciona correctamente y, ademas, obtiene información que puede ser útil para los demas componentes.

[13] Estrategia en la cual una clase es accesible desde cualquier parte del código y, además, se asegura que solo puede existir una.

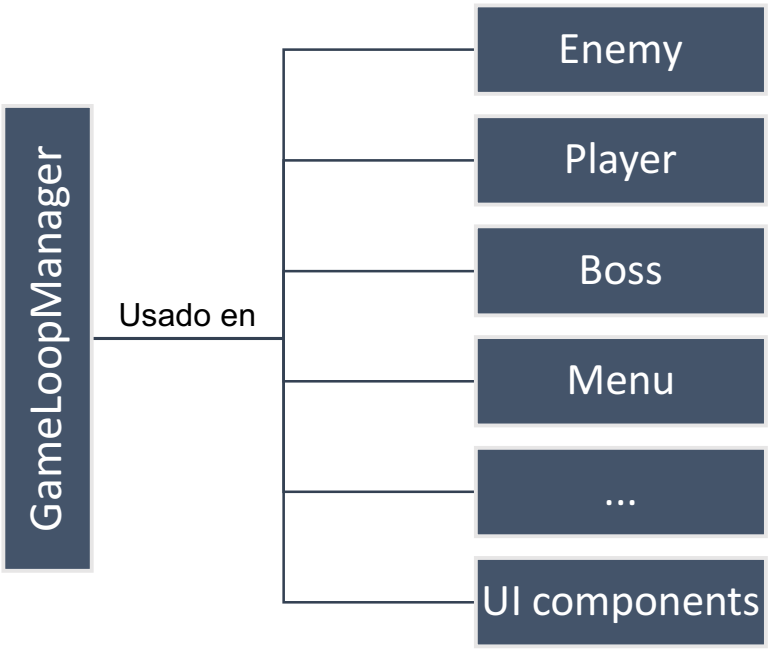

Ilustración 41 GameManager arquitectura

4.5.1 Jugador

El personaje principal tiene bastantes acciones a realizar: moverse, el *dash,* la bola de fuego, el escudo… Se podría poner todas estas acciones en un solo controlador. Pero se ha decidido utilizar la siguiente estructura.

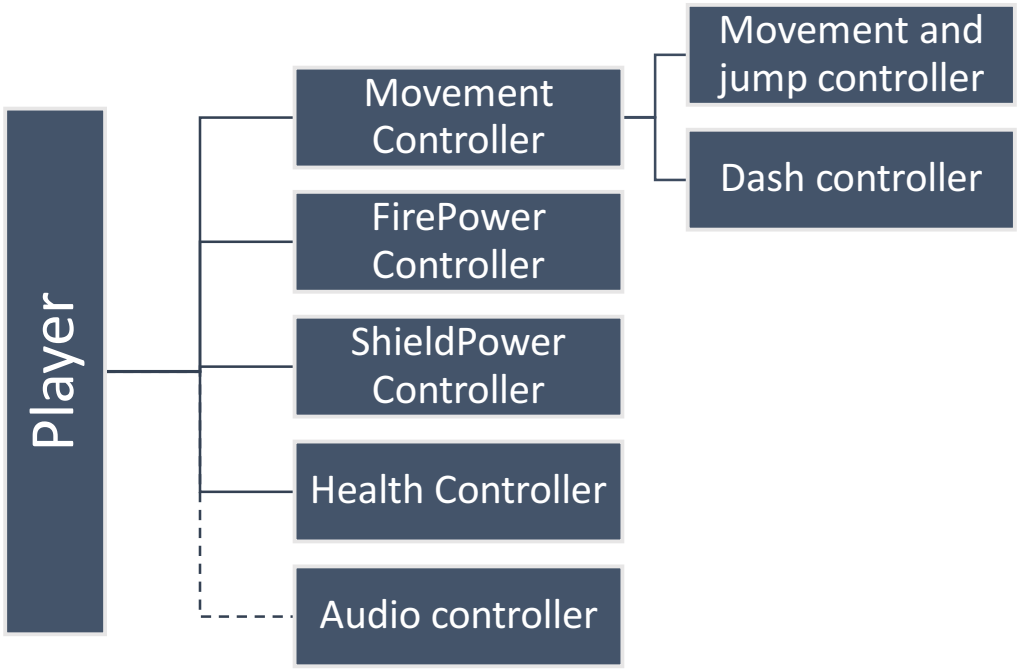

Ilustración 42 Arquitectura jugador

Podemos ver que el jugador se separa en 5 componentes o *scripts.* Cada uno de ellos se encarga de una única cosa, aunque se pueden comunicar entre ellos. Por ejemplo, el *ShieldController* se encarga de activar o desactivar el escudo y el *HealthController* de matar al jugador cuando sea golpeado por un elemento

dañino. Sin embargo, el segundo debe actuar solo cuando el escudo este activo, así que tendrá que preguntar al otro *script* si el escudo está activo o no.
El *MovementController* se encarga tanto del movimiento como del *dash*. Esto se debe a que son dos acciones que tienen que ir de la mano o, mejor dicho, se chocan entre ellas. Que dos o más componentes se complementen entre ellos es algo normal, pero abusar de ello puede ser síntoma de que ambos componentes debería ser solo uno [2].
El *AudioController* está en línea discontinua dado que es un componente reutilizado, no es propio del jugador.

4.5.2 Enemigos

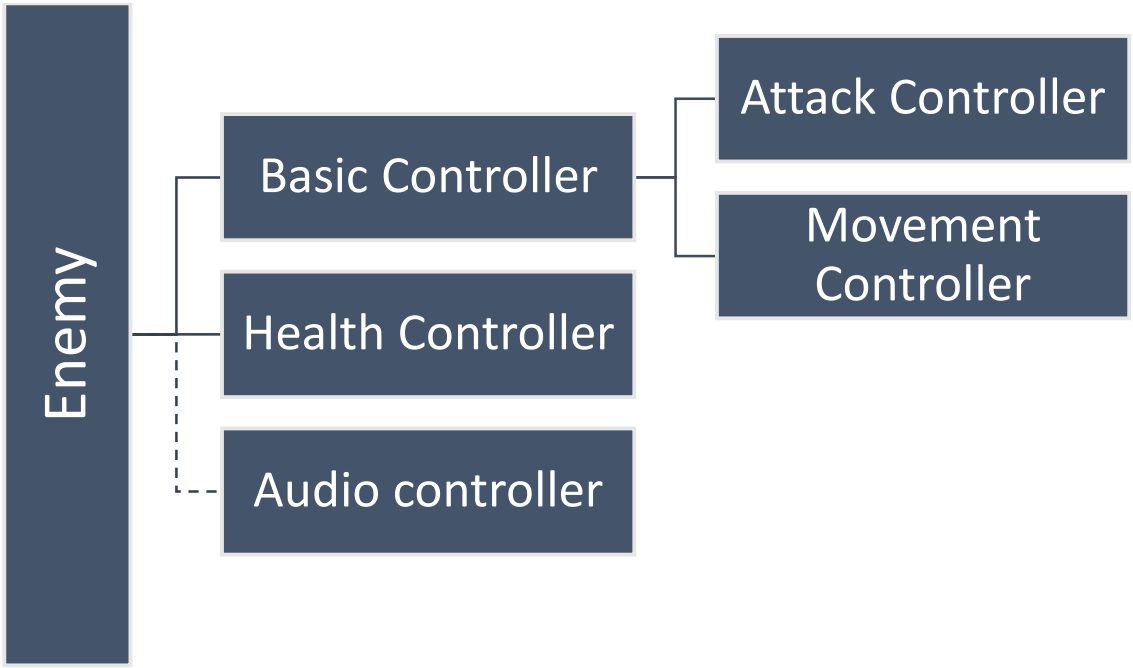

Ilustración 43 Arquitectura enemigo

Esta arquitectura funciona tanto para los enemigos cuerpo a cuerpo como a distancia. Funciona de forma parecida que el del jugador. El *BasicController* hace la función de atacar y de moverse ya que el enemigo no puede hacer ambas al mismo tiempo. Se podrían haber separado y simplemente consultar esta propiedad.

4.5.3 *Boss*

A diferencias de los demás, este solo tiene un *script* o componente que se encarga de todo. Esto se debe a que la vida, los ataques, los movimientos y el total del enemigo están muy relacionados. Se ha decidido separar por fases[14] y la estrategia de tener todos los elementos en el mismo *script* para poder modificar su comportamiento según la fase, era mucho más sencilla. Otra estrategia podría haber sido separar la fase en distintos *scripts*. Sin embargo, la

[14] Las veremos más adelante, en el siguiente apartado

poca complejidad del enemigo en cuestión hacía esa aproximación un poco sobre exagerada.

4.5.4 Demás elementos

Todos los demás objetos del videojuego reutilizan distintos *scripts*. Por ejemplo, el *script* llamado *DeadlyObject*, que se encarga de matar al jugador si colisiona con el objeto al que este adjunto, se usa en las flechas, en los pinchos y en las ruedas que hay por el nivel. No se van a comentar todos, pero sí que se quiere recalcar que cada uno de ellos hace una tarea pequeña y la mayoría de ellos no supera las cincuenta líneas de código.

4.5.6 Eventos

Otra de las estrategias con las que nos invita a trabajar *Unity* es con los eventos [3]. Esta forma de trabajo consiste en crear una función o procedimiento, que espere a ser ejecutado cuando cierta acción, programada, suceda. Es muy útil cuando se usa conjuntamente con las animaciones. Cuando cierto *frame* de la animación se ejecute, este lanzará el evento, que será recogido por el objeto asignado y se ejecutará.
El ejemplo más claro que se puede ver este videojuego es el ataque de los arqueros. Cuando el jugador es avistado, este empieza la animación de disparo y, en cierto momento, esta lanza el evento, que es recogido de nuevo por el controlador y, en ese momento, instancia la flecha.

4.6 IA de los enemigos

Hay tres tipos de enemigos que tienen una pequeña inteligencia artificial asociada. Estos son los arqueros, los lanceros y Riole, el jefe final.

4.6.1 Arqueros y lanceros

Juntamos estos dos enemigos, ya que comparte el mismo *script* y simplemente cambiando dos variables podemos hacer que cambie su comportamiento.
La idea del movimiento es realmente sencilla, pero funciona muy bien. Haciendo un resumen, el enemigo siempre intentará avanzar de frente. Solo si se encuentra con un obstáculo, por ejemplo, unos pinchos o caída “libre”, dejará de avanzar.
Para que avance hacia el jugador y no de forma aleatoria, nos aseguramos que siempre este “mirando” al jugador. Esto se consigue restando la X de la posición del jugador y la X de la posición del *NPC.* Según si este resultado es positivo o negativo, giraremos al enemigo.
Para calcular los objetos, se hace uso de *Raycast.* Lanzamos un rayo horizontal hacia delante una corta distancia, y si choca con algo, es que hay un objeto. Lo mismo para detectar si hay un precipicio delante o no. Lanzamos de nuevo un rayo, esta vez verticalmente, a una pequeña distancia enfrente de nuestro enemigo en cuestio. Si este rayo colisiona con el terreno, significa que puede avanzar sin caerse.

El ataque también usa la misma técnica. Lazamos un rayo, cuya distancia dependerá si el ataque es a distancia o no. Si choca con el jugador, iniciamos la animación de atacar.
Es importante señalar que, si el personaje está atacando, no se puede mover.

4.6.2 Riole

La IA del *boss* es bastante sencilla. Se ha separado en 4 fases. Una de ataque, otra de reposo y otras dos intermedias, que permiten pasar de una fase a la otra.
En la fase de ataque, el enemigo puede recibir daño, lo que hace que su escudo baje, mientras él tira flechas en todas las direcciones. Cuando el escudo llega a cero, pasa a la siguiente fase.
En esta segunda fase, el enemigo simplemente se mueve hasta el suelo y ya no recibe daño de las bolas de fuego, solo de los *dashes.* Cuando llega al suelo, cambia de fase a la de reposo.
En esta fase, su escudo se regenera poco a poco y es también inmune al daño de las bolas de fuego. Cuando se le haga daño de *dash,* cambiará de fase a otra que simplemente le hace subir.
Cuando la fase de subir acabe, pasará de nuevo a la fase de ataque.
Estas fases se repetirán hasta que el jugador mate al *boss.*

4.7 Mecánicas

Aunque no se van a explicar todas las mecánicas, si se pondrán las más importantes, que caracterizan este juego en concreto.

4.7.1 Movimiento

La mecánica más básica que posee el jugador. Permite mover al jugador hacia la derecha o a la izquierda, pero no hacia arriba ni hacía abajo. En el caso de que el jugador se encuentre en el aire, también podrá moverse libremente hacia los laterales.

4.7.2 Salto

Otra mecánica básica del jugador, que le permite moverse hacia arriba momentáneamente, para luego empezar a caer hasta que se encuentre con el suelo. Esto permite al jugador acceder a plataformas y esquivar desniveles que el movimiento no es capaz de superar.

4.7.3 Salto contra la pared

Mientras que la anterior mecánica permite al jugador moverse verticalmente cuando esta posado sobre el suelo, esta permita moverse verticalmente al estar contra una pared. Este salto no es solo vertical, sino diagonal, siempre hacia arriba y hacia el lado contrario al cual se encuentra la pared (con respecto al jugador)

4.7.4 *Dash*

Es movimiento hacia cualquier dirección que desplaza al jugador una pequeña distancia de forma casi instantánea. Tiene un rango establecido y no se puede parar a mitad de camino, aunque dado que no permite atravesar el terreno, este movimiento se acabará al colisionar.
En caso de colisionar contra un enemigo, lo matará. El *dash* se puede realizar un total de dos veces sin tocar el suelo. Cuando el jugador toca el suelo, podrá realizar los dos *dashes* de nuevo.

4.7.5 Escudo

Hace inmune al jugador de casi todo el daño que se le pueda ocasionar. Tiene un tiempo hasta que desaparezca, y un tiempo para poder volver a ser usado.

4.7.6 Bola de fuego

Un proyectil que mata a los enemigos que impacta. Tiene un tiempo hasta poder ser usado de nuevo. Si colisiona con una flecha, tanto la flecha como la bola de fuego desaparecerán.

4.7.7 Enemigos

Enemigos que atacarán al jugador. Un solo golpe de ellos será el fin, y deberá comenzar de nuevo.

4.7.8 Pinchos

Trampas mortales para nuestro jugador. Tocar una de estas supone morir. Existen dos tipos, las estáticas y las móviles.

4.7.9 Plataformas

Suelo sobre el que el jugador se posa. Existen varios tipos. Algunas de ellas son trampa, ya que desaparecen unos instantes después de que el jugador se pose. Otras se pueden traspasarse por debajo pero no por arriba.

4.7.10 Diamantes

Refrescan los *dashes* del jugador, permitiéndole volver a hacerlos sin necesidad de tocar el suelo.

4.7.11 La ausencia de *checkpoints*

Cuando el jugador muere, no vuelve a reaparecer en un *checkpoint* cercano, sino que deberá comenzar de nuevo la pantalla en la que se encuentra actualmente.

4.7.12 Combinaciones

La mecánica más importante de todas estas, es la conjunción y la combinación de todas ellas, dando lugar a miles de escenarios, cada uno de ellos distintos del anterior.

5. Diseño de niveles

En este apartado se va explicar cómo avanzar por un nivel de ejemplo. Hay que tener en cuenta que no se ha hecho el *walkthrough* en una sola partida, ya que para hacer las capturas se ha tenido que pausar el juego, lo que hacía complicado continuar jugando sin perder el hilo de los que se estaba haciendo. Por eso mismo, el contador de segundo puede no coincidir con el orden de las capturas, y lo mismo pasa con las muertes.

Explicar el nivel tutorial es un poco absurdo ya que no hay dificultad alguna, y el nivel del *boss* es demasiado simple, así que se va a explicar el segundo nivel de esta demo que se presenta.

Empezamos el nivel en una plataforma grande, donde hay pared a la izquierda y vacío a la derecha. Si nos pegamos a la derecha, quedándonos en el filo se puede ver una plataforma flotando. Es hacia a ella donde nos tenemos que dirigir para empezar nuestra aventura.

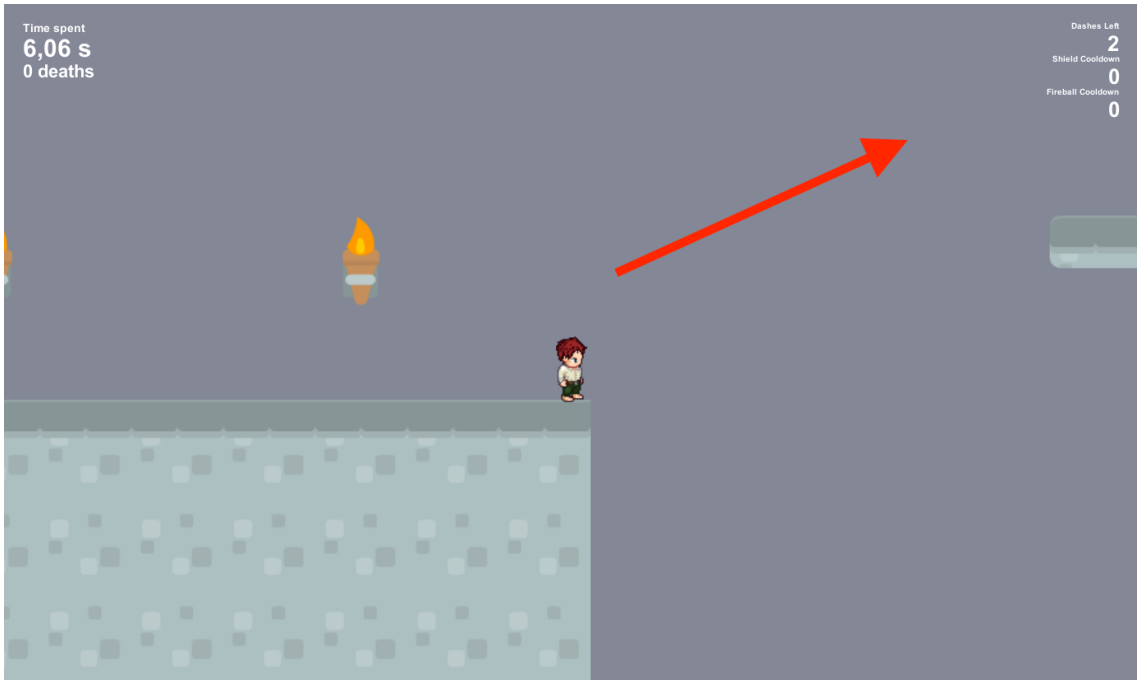

Ilustración 44 *Walkthrough* fase 1

Se llega fácilmente a la plataforma con un salto y un doble *dash,* pero hay que tener cuidado ya que es una plataforma trampa. Esta plataforma desaparecerá unos instantes después de pisarla. Tenemos que ir avanzando por ella, para evitar caernos la vacío. Si se quiere calcular el tiempo, podemos dejar caernos al pisarla y, cuando empecemos a caer volver a la plataforma de inicio. A los 5 segundos se regenerará, y podemos volver a intentarlo, sabiendo esta vez el tiempo que tenemos para avanzar.
Continuamos por la plataforma, hacia la derecha y, cuando lleguemos al final, podremos ver en la esquina superior derecha otra plataforma de este tipo. Debemos saltar hasta a ella. Una vez alcancemos esta plataforma deberemos saltar verticalmente como muestra la siguiente ilustración (Ilustración 45

Walkthrough fase 2), hacer *dash* vertical y otro lateral hacia la izquierda, para alcanzar la plataforma que se puede ver.

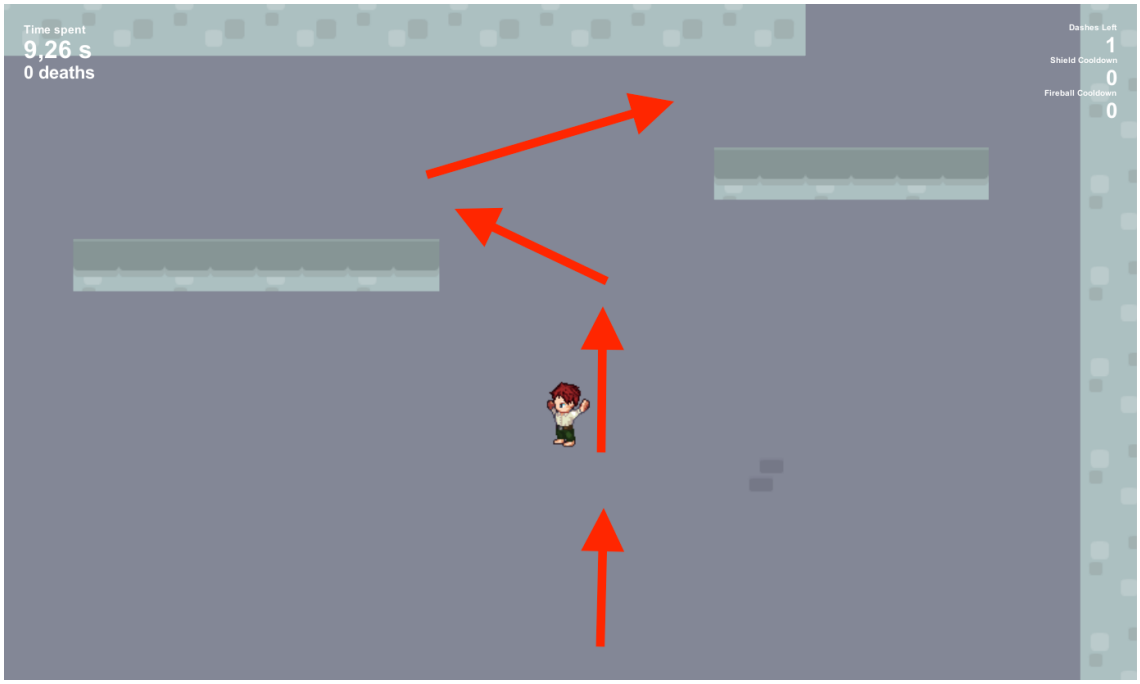

Ilustración 45 *Walkthrough* fase 2

Una vez en dicha plataforma, se puede acceder fácilmente a la siguiente, tal y como muestran las flechas. Continuamos andando y saltamos verticalmente hacia arriba, ayudándonos de los *dashes* para poder alcanzar la plataforma.

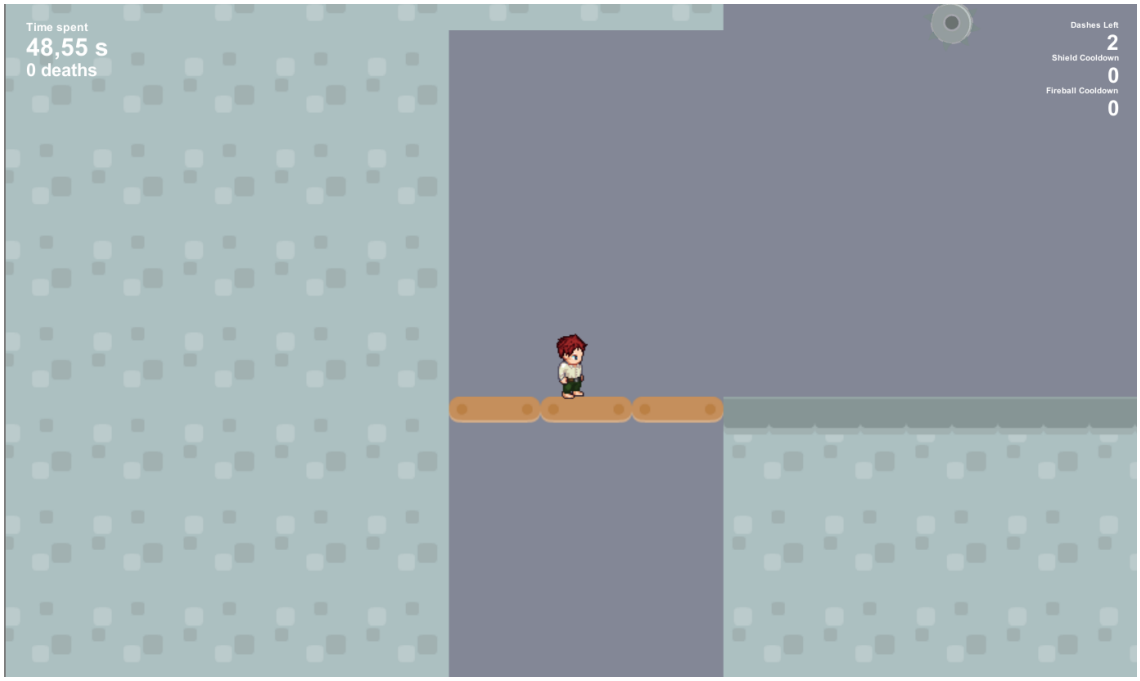

Ilustración 46 *Walkthrough* fase 3

Avanzamos hacia la derecha, esquivando los pinchos que hay en nuestro camino. Simplemente saltando vertical y un *dash* lateral se pueden esquivar,

mientras estos suben. Después de esquivar los más sencillos, nos enfrentamos a unos pinchos “dobles”.

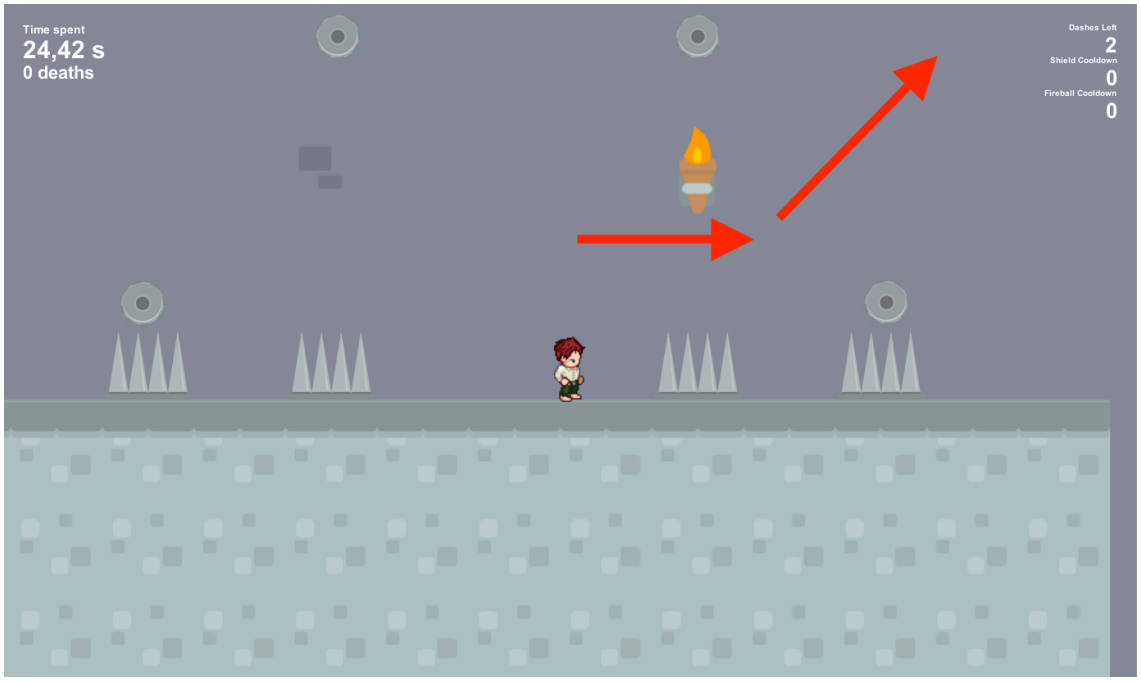

Ilustración 47 *Walkthrough* fase 4

La forma más fácil de esquivarlos es como se ha mostrado en la ilustración 47. Saltar verticalmente cuando el primer pincho está subiendo, para realizar un *dash* lateral a la derecha y, justo cuando acabe, hacer otro en diagonal hacía arriba y derecha. También se puede superar este obstáculo haciendo un único salto y un único *dash,* para acabar entre los dos pinchos. Sin embargo, requiere controlar mejor las distancias que hará el personaje al realizar todos y cada uno de los movimientos.

Una vez superada esta parte, nos encontramos ante un precipicio al cual nos tenemos que lanzar, esquivando los primeros pinchos que nos encontramos. Podemos dejarnos caer para superar también los que se encuentran casi al final de la caída, pero hay que tener en cuenta que nuestro personaje gana mucha velocidad mientras va cayendo, así que puede ser difícil acertar y esquivarlos.
Por eso mismo, se recomienda hacer una mini parada en el medio, donde podemos encontrar dos huecos en la socavados en la pared. Sin embargo, no es fácil entrar en ellos, ya que también tienen una barrera que superar. Estos pinchos no van a la par, mientras uno de los dos está en un extremo, el del otro lado se encontrará justo en el medio de la entrada. Con un poco de ojo, se debe ver cual de los dos tiene la “puerta” abierta. Recordad que podemos hacer dos *dashes* y que no hemos usado ninguno. Puede ser una buena idea realizar un primer *dash* verticalmente hacia arriba, simplemente para ganar tiempo y ver en si debemos ir a la derecha o a la izquierda. Una vez decidido entramos en la que veamos que tiene una mayor apertura.

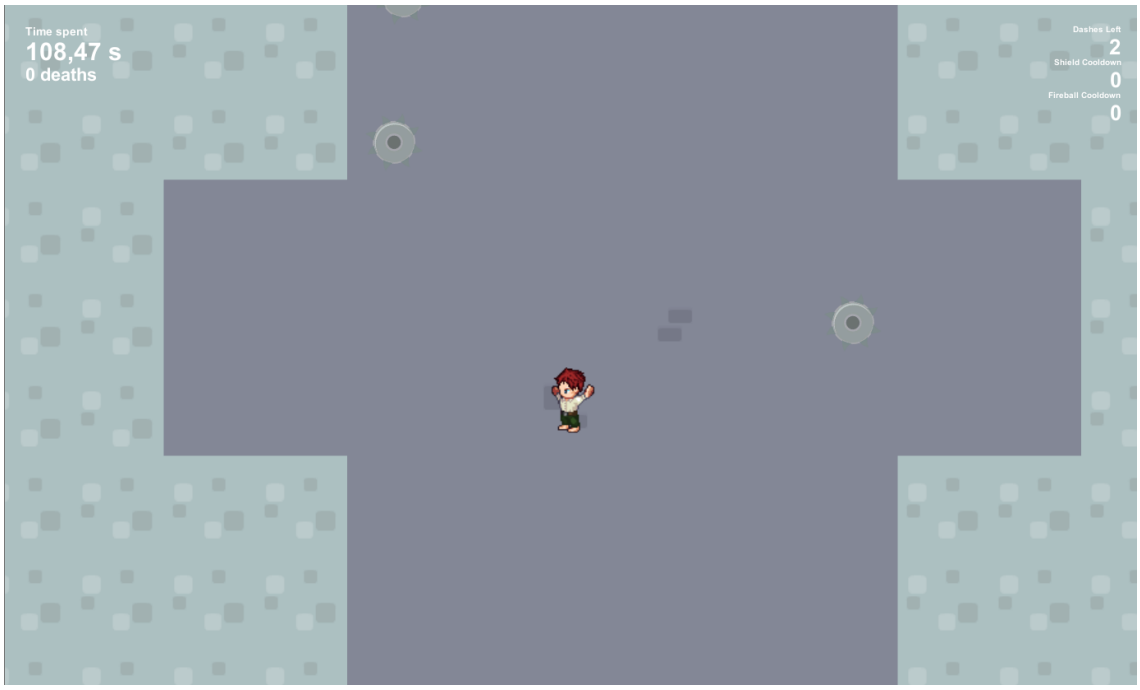

Ilustración 48 *Walkthrough* fase 5

Después de llegar al suelo, nos encontramos con un pasillo, donde no hay ningún obstáculo y podemos descansar nuestra mente y nuestros dedos, aunque no por mucho tiempo.
Al llegar al final de dicho pasillo, deberemos saltar hacia arriba para llegar a una plataforma doble. Estas dos plataformas tienen unos pinchos que suben y bajan, mientras que en el lado de la izquierda estará arriba, el de la derecha estará abajo y viceversa, siempre alternándose. No hay que desesperar, ya que da tiempo a cambiarse de un lado a otro para no morir. Hay un total de 5 plataformas con este tipo de obstáculos que habrá que obstaculizar. Se intentará explicar en las siguientes imágenes como esquivar una muerte segura. Probablemente se necesite más de un intento.

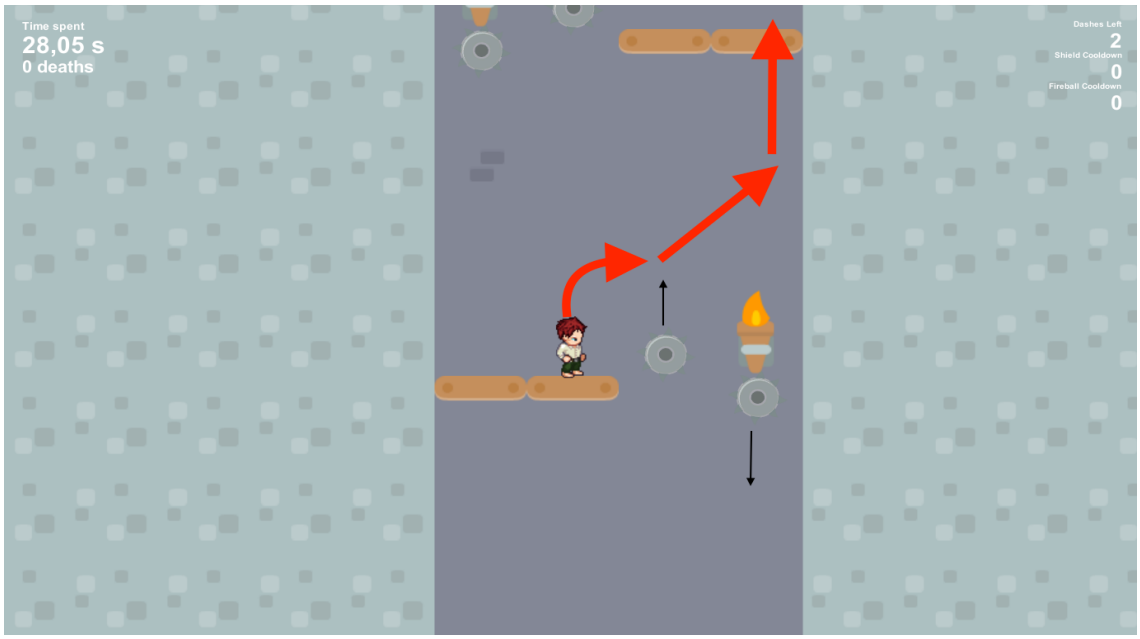

Ilustración 49 *Walkthrough* fase 6

Cuando estamos en las plataformas colocadas en la parte de la izquierda (Ilustración 49 *Walkthrough* fase 6), debemos esperar a que los pinchos se crucen, cuando el más cercano a nosotros este subiendo, tal como muestra la ilustración referenciada anteriormente. Es en ese momento cuando hay que saltar y realizar dos *dashes*, uno en dirección arriba-derecha y el siguiente solo hacía arriba. En este momento, el pincho de la derecha de esta plataforma, se encontrará arriba, mientras que el izquierdo estará abajo. Podemos ir turnando de lado para evitar los pinchos, hasta que nos veamos capaces de saltar hasta la otra plataforma.

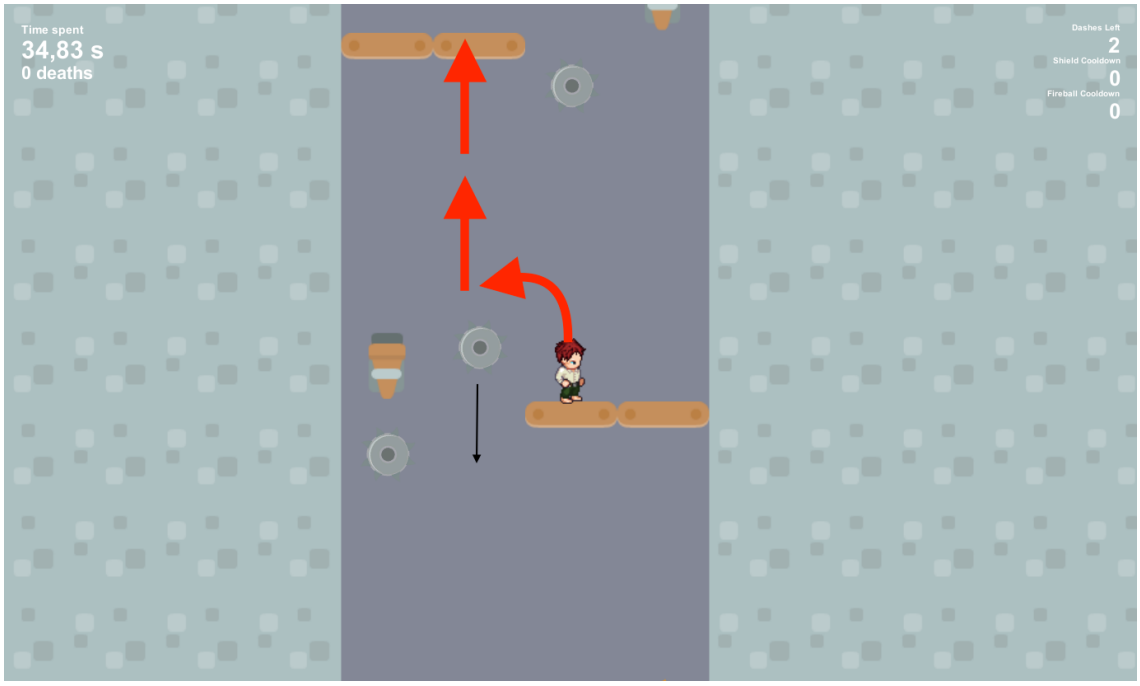

Ilustración 50 *Walkthrough* fase 7

En este lado, subir es mucho más sencillo que la anterior. De nuevo, nos ponemos en el filo y, cuando el pincho más cercano a nosotros este bajando saltamos sobre él y realizamos dos *dashes* para llegar a la plataforma superior. El pincho del lado derecho, lado en el que estamos, estará arriba, pero probablemente empiece a bajar antes de que nos de tiempo de subir a la siguiente plataforma. Por eso es recomendable cambiar de lado.

Para seguir subiendo, solo debemos realizar estos pasos de nuevo, hasta que lleguemos al final. Cuando superemos esta zona nos encontraremos en una zona abierta, donde habrá 6 arqueros que nos dispararán sus flechas. Podemos matarlos o simplemente ignorarlos. Debemos ir hacia la derecha, hasta que lleguemos a una pared vertical. Se recomiendo no perder tiempo, ya que algunas flechas todavía nos pueden dar.

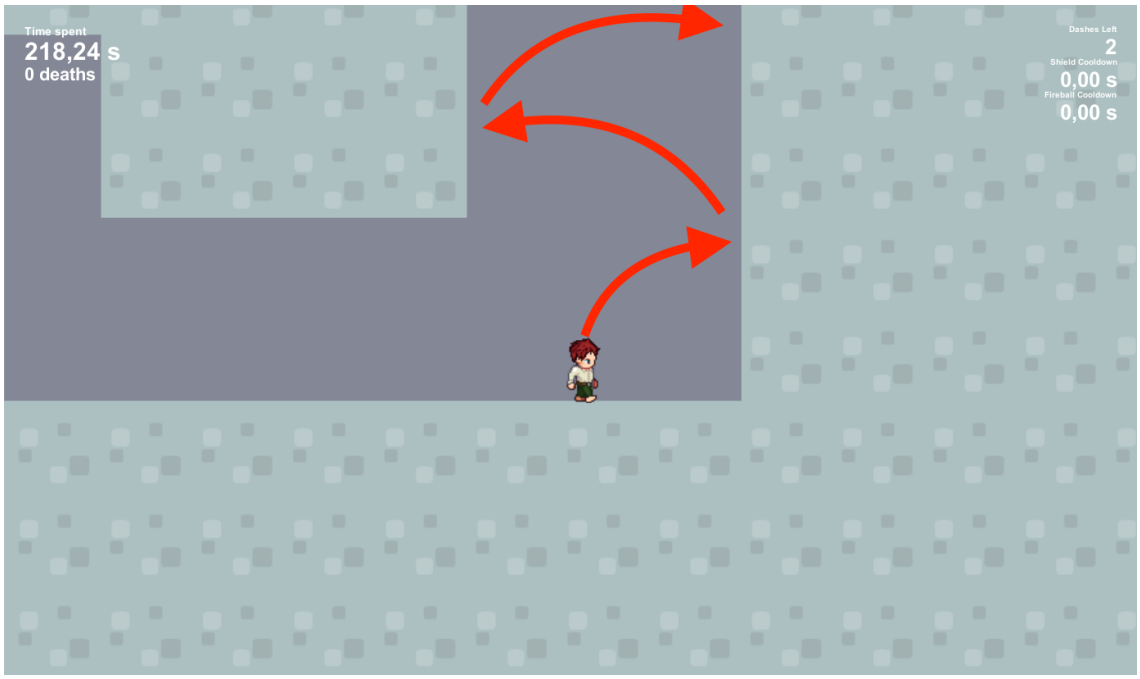

Ilustración 51 *Walkthrough* fase 8

Por fin, nos encontramos con un obstáculo que fallar no supondrá la muerte y tener que volver a empezar. Lejos de eso, si fallamos aquí, solo caeremos hasta el punto de partida. Aunque de nuevo, no nos durmamos ya que podemos ser alcanzados por alguna flecha.
Para poder subir esta pared simplemente hay que hacer el salto contra la pared, lo que nos impulsará hacía atrás. Saltando constantemente podremos subir toda la pared vertical. Podemos hacer uso de los *dashes* para no tardar tanto o si fallamos un salto, pero no son necesarios para pasarnos este punto.
Arriba, nos encontraremos con una pequeña “curva” y después otra pared vertical en la cual tendremos que repetir este proceso. Sin embargo, hay que tener cuidado ya que esta es traicionera. Al final, en la plataforma que tenemos que alcanzar, hay pinchos. Hay que tener cuidado no caigamos sobre ellos.
Una vez estemos a salvo en este punto, enfrente de los pinchos, deberemos saltar a la izquierda, por encima de los pinchos, y realizar dos *dashes* para alcanzar la pared de la izquierda.

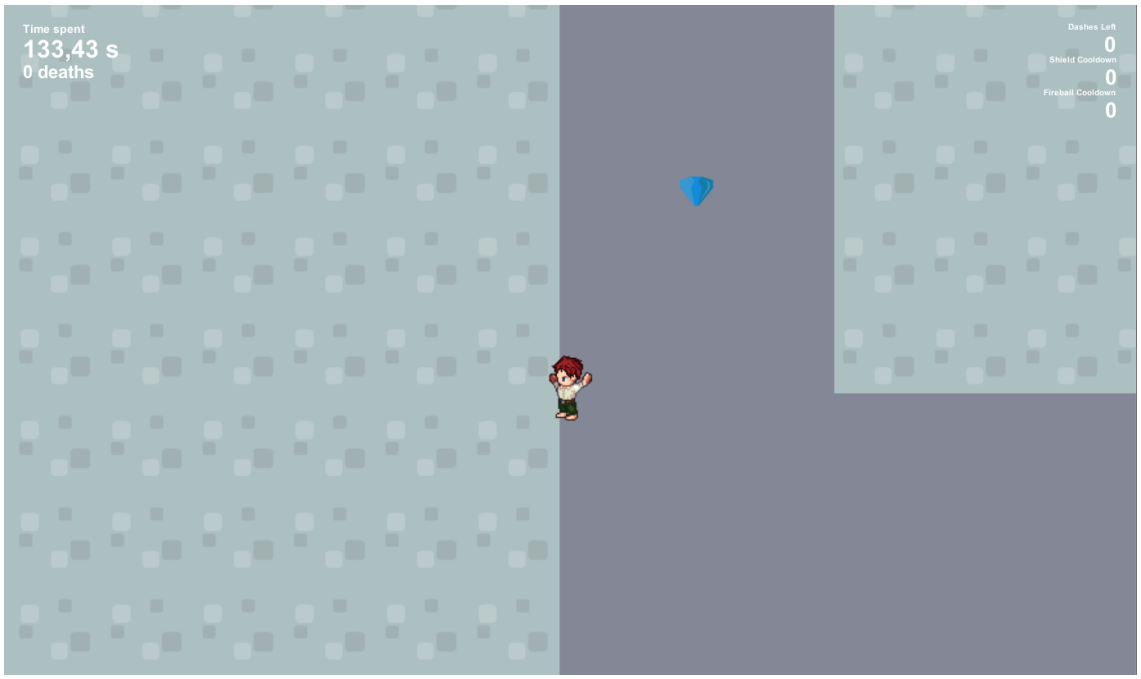

Ilustración 52 *Walkthrough* fase 9

Aquí deberemos saltar otra vez usando las paredes para impulsarnos hacia arriba. Podemos coger el diamante, que nos da una segunda oportunidad en caso de fallar, otorgándonos la oportunidad de esquivar los pinchos. Si controlamos perfectamente los saltos, podemos usar los *dashes* simplemente para llegar antes a la plataforma.
El siguiente obstáculo es también bastante sencillo, como podremos ver.

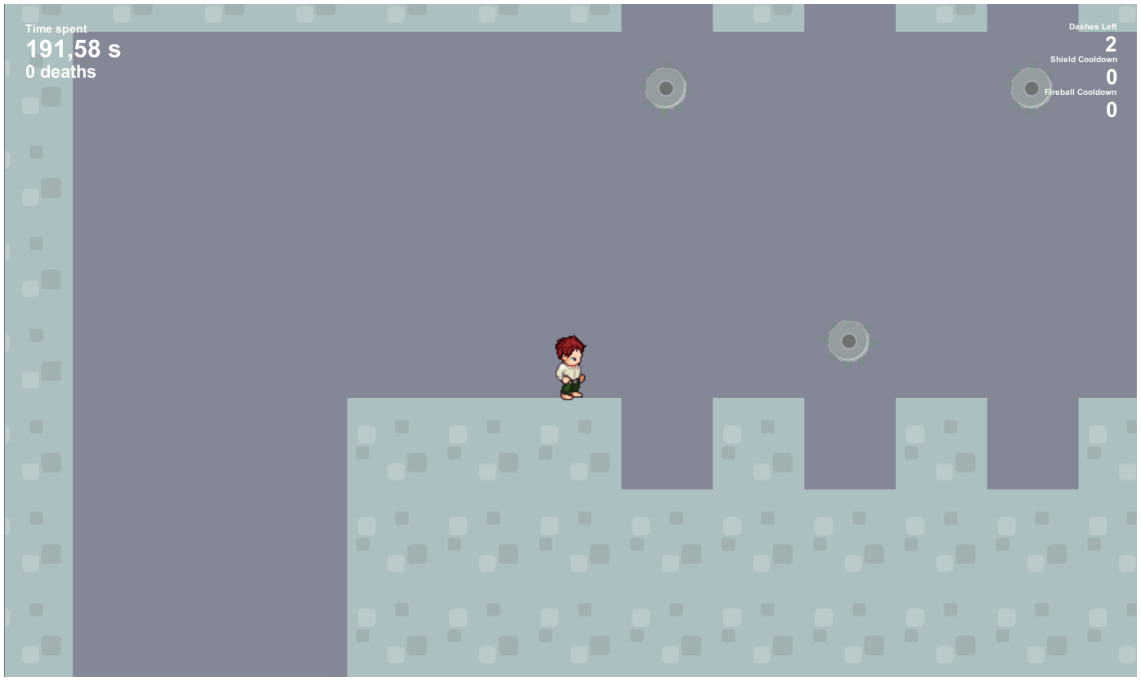

Ilustración 53 *Walkthrough* fase 10

Debemos esperar a que los pinchos suban para simplemente avanzar, ya sea con un salto o con un *dash.* Si somos rápidos, podremos escapar de una muerte segura en el caso de que caigamos en uno de los huecos.

La siguiente fase, es difícil de dar una técnica que funcione en todos los casos, es mejor esperar hasta entender el patrón de movimiento de los pinchos y, una vez se haya entendido, saltar en consecuencia. Aunque sí que es verdad que en la mayoría de los casos se puede aplicar la estrategia o, mejor dicho, el movimiento que hay que hacer para poder pasar sin morir. Simplemente debemos acercarnos al límite y realizar un *dash* en diagonal hacia abajo y al lado que corresponda, tal y como se indica en la siguiente imagen. Al ser posible cuando los pinchos se alejan de la forma que se indica en la misma ilustración.

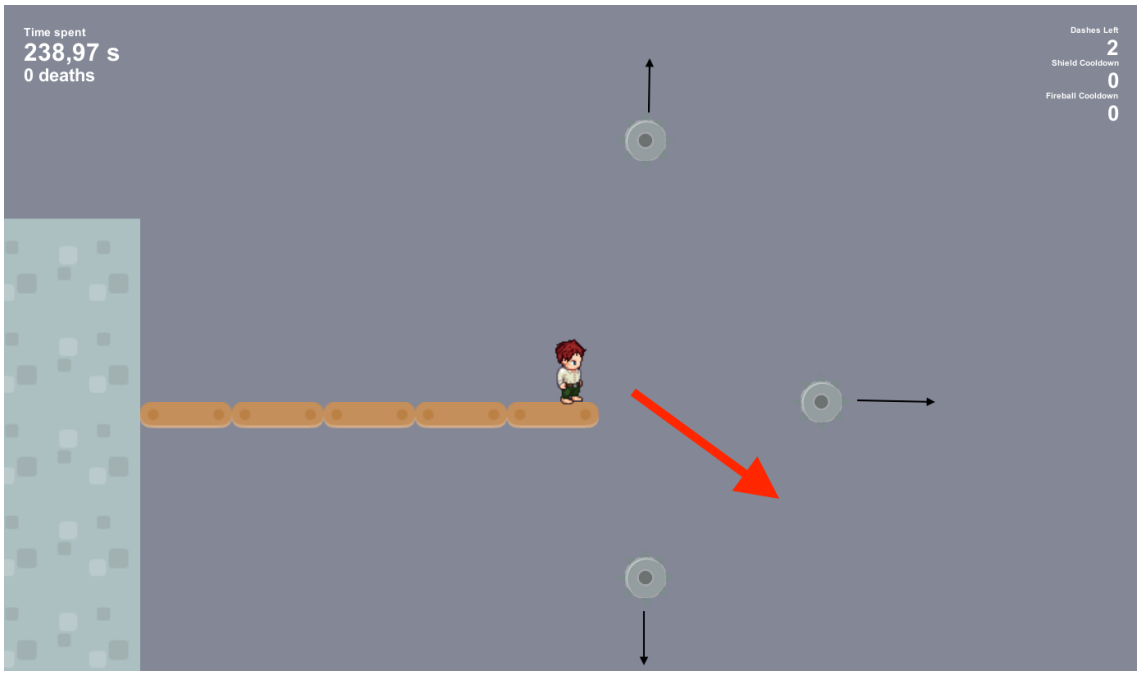

Ilustración 54 *Walkthrough* fase 11

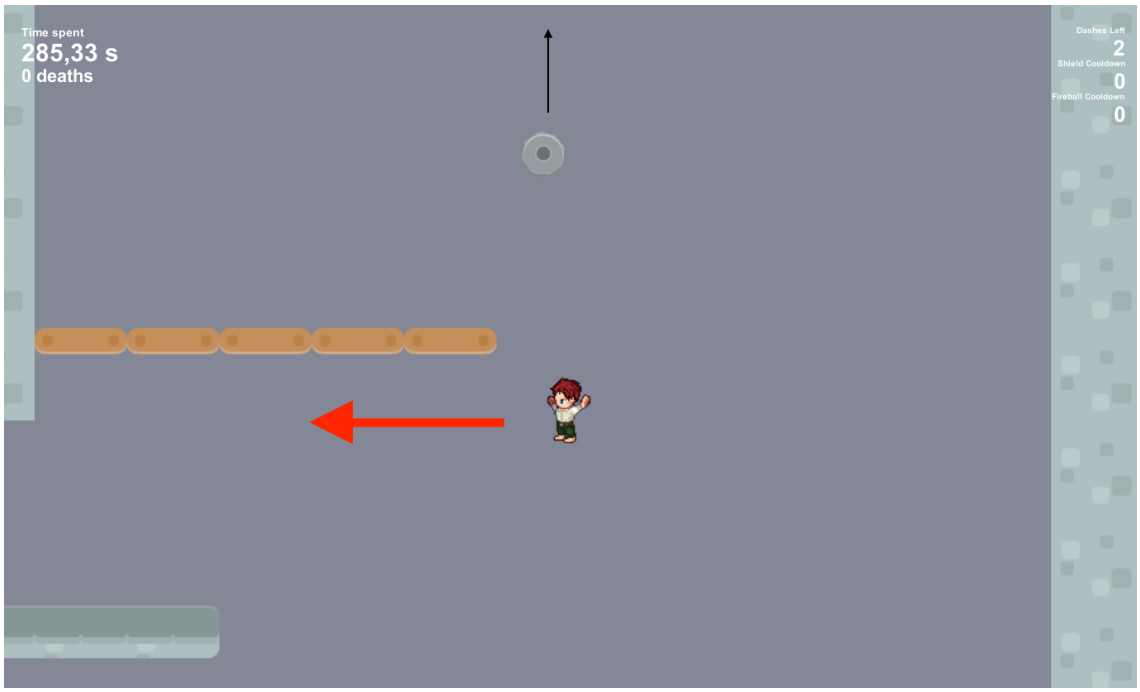

Ilustración 55 *Walkthrough* fase 12

En la plataforma final, solo debemos dejarnos caer y luego alcanzar la plataforma que vemos en la ilustración. Hay que tener cuidado, ya que no solo la plataforma va a desaparecer cuando nos posemos sobre ella, sino qué, además, al final del todo, hacia la izquierda, hay un arquero que empezará a dispararnos. Dado que es solo un enemigo, no es difícil de esquivar sus ataques y matarlo. Debemos recordad que las flechas se pueden evitar con el escudo o disparándole una bola de fuego, lo que las destruirá.
Una vez eliminado el arquero, solo debemos dejarnos caer entre los pinchos, esquivándolos como se muestra en la siguiente ilustración.

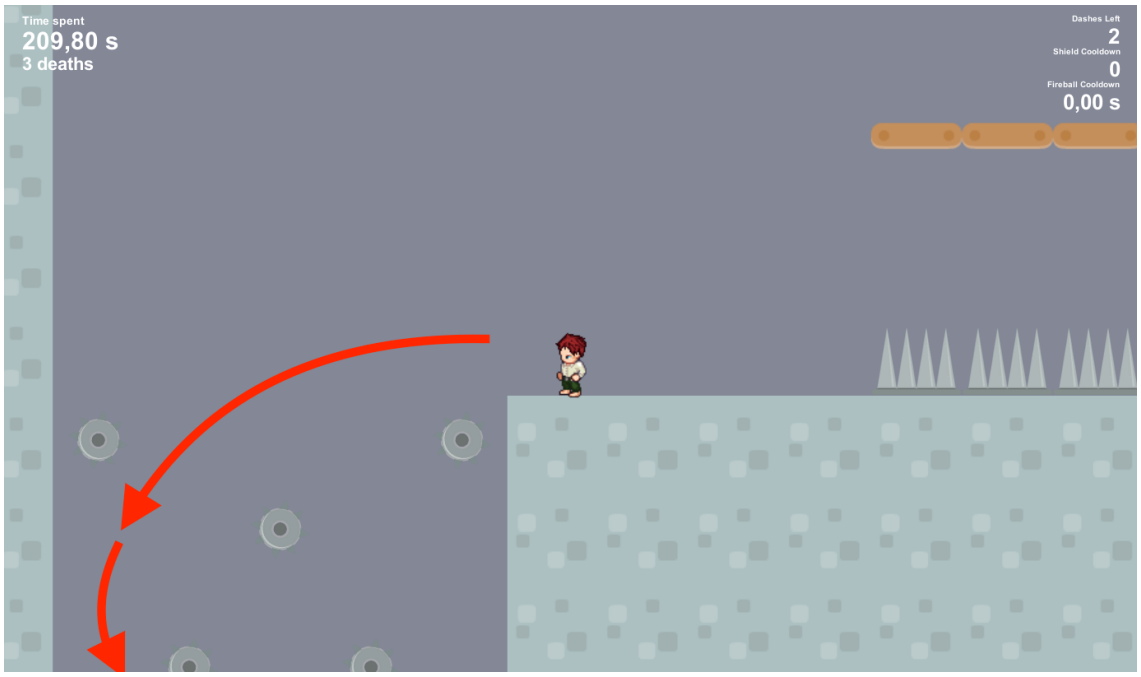

Ilustración 56 *Walkthrougn* fase 13

No hace falta usar ningún *dash* en este punto de la partida. Simplemente debemos saltar y movernos hacia la izquierda. Una vez superada la tercera línea de pinchos, debemos movernos hacia la derecha para esquivar los que nos encontraremos. Es un conjunto de pinchos casi simétrico en ambos ejes, así que, si nos dejásemos simplemente caer, acabaríamos muriendo y teniendo que repetir todo.
De nuevo, hay que tener en cuenta que nuestra velocidad de caída aumentará, así que a medida que caemos, tendremos menos tiempo de esquivar la muerte. Intentar esquivar los primeros pinchos pasando lo más cerca ellos puede ser una buena idea para luego tener más tiempo de esquivar los otros.

Al salir, nos encontraremos sobre una plataforma, con un sinfín de pinchos debajo nuestra. Tendremos que volar a través de ellos haciendo uso de los *dashes* y de los diamantes que nos permiten volver a hacerlos.

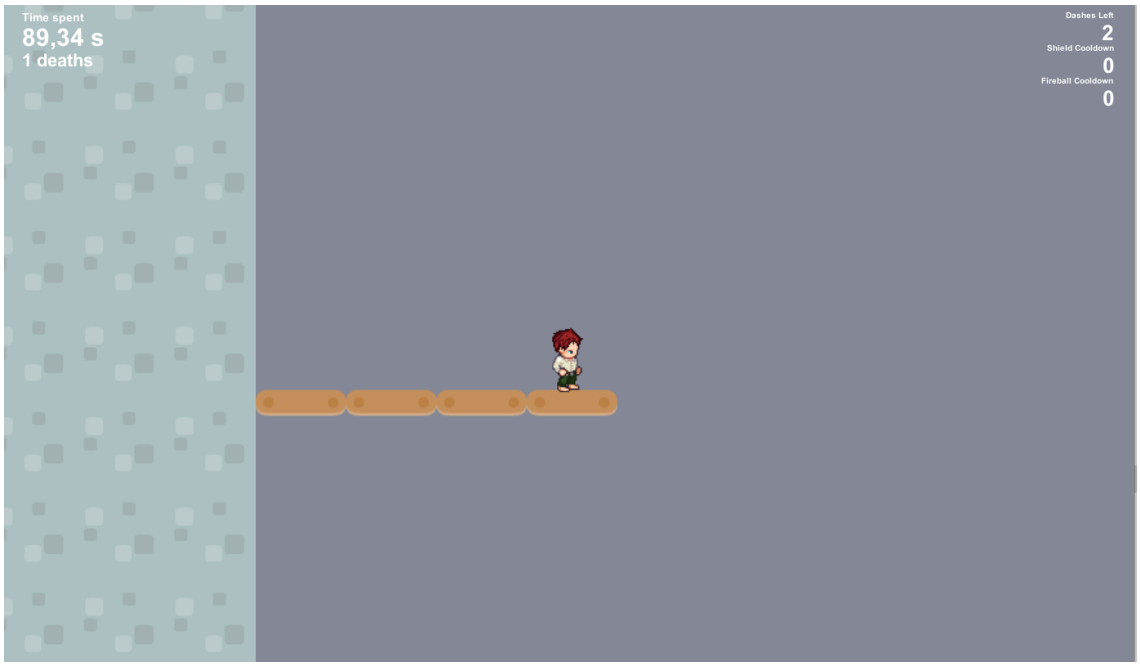

Ilustración 57 *Walkthrough* fase 14

Saltamos hacia el vacío y, cuando nos encontramos más o menos al mismo nivel que la plataforma de origen, realizaremos el primer *dash,* hacia arriba y a la derecha. Nos dejamos caer de nuevo para volver a realizar el segundo *dash.* Si se ha hecho en el momento oportuno, caeremos sobre un diamante. Nada más cogerlo deberemos volver a repetir el proceso hasta alcanzar el siguiente diamante.
Al coger este diamante, podemos ver que el siguiente se encuentra sobre nosotros. Debemos realizar un *dash* vertical y otro en diagonal, para poder cogerlo. El siguiente diamante está a la misma altura que el que acabamos de coger, así que debemos volver a realizar los dos *dashes* diagonales, dejándonos caer entre el primero y el segundo.
Una vez alcanzado, solo nos queda uno por coger. Este, se encuentra al mismo nivel que los primeros, así que debemos dejarnos caer un poco y realizar el *dash*, evitando que nuestro personaje pille velocidad debido a la caída. Recordad que, cuando estamos mucho tiempo en el aire la fuerza de la gravedad sobre nuestro personaje se incrementa, lo que provoca que cada vez pillemos velocidad más rápido. El *dash* o un salto resetea este valor.
Aunque no es muy difícil, si no se ha hecho una primera vez o se sigue una guía, lo más probable es que muramos alguna vez en el intento. Realizar los saltos al azar puede provocar que no llegues a coger uno de los diamantes o que los sobre pases. No pillar uno de estos diamantes supone caer en los pinchos sin poder evitarlo.
Una vez superado, estaremos a salvo sobre una plataforma similar a la del inicio.

Ilustración 58 *Walkthrough* fase 15

En este punto, debemos volver a realizar la estrategia usada en la fase 8 y 9 (Ilustración 51 *Walkthrough* fase 8 e Ilustración 52 *Walkthrough* fase 9, respectivamente). En esta ocasión, solo saltar no nos sirve, ya que no llegamos a la pared del lado contrario para poder subir. Debemos hacer uso de los *dashes* para poder subir.
Justo al realizar un salto contra la pared, debemos realizar el *dash* que nos permite subir un poco y llegar a la pared. Los diamantes están contados, hay uno cada dos saltos. Si fallamos, podemos saltar de lado a lado y esperar a que vuelva a salir el diamante. En este caso, nuestro personaje no sube, pero tampoco cae, se mantiene más o menos en el mismo nivel siempre.
Al llegar arriba, nos encontramos una serie de plataformas sobre las que hay que ir subiendo. No hay peligro alguno, ni pinchos ni enemigos, solo plataformas. Caernos de ellas tampoco supone tener que volver a empezar y nada por el estilo, es un momento de tranquilidad.

La ilustración siguiente muestra lo que nos encontramos al superar la subida vertical. No tenemos descanso alguno, ya que no podemos apoyarnos en ningún lado. Tenemos que hacerlo todo seguido.
La combinación idónea para salvar esta fase es la siguiente. Debemos saltar verticalmente, evitando tocar los pinchos. Saltar usando la pared como impulso, lo que nos hará alcanzar la pared del lado derecho. Utilizar esta pared para volver a saltar hacia el otro lado. Sin esperar a empezar a caer, hay que realizar los dos *dashes* hacia el diamante que hay flotando sobre los pinchos. Nada más cogerlo, debemos volver a realizar los dos *dashes* que el diamante nos ha otorgado. El primer *dash* nos permitirá alcanzar la pared de la derecha y el segundo es simplemente para llegar un poco más alto, facilitando los siguientes saltos. Debemos impulsarnos con las paredes para llegar arriba. En el caso de fallar un salto y caer, debemos movernos hacia la derecha lo más rápido que podamos, ya que podremos llegar a las plataformas y evitar caer sobre los pinchos.

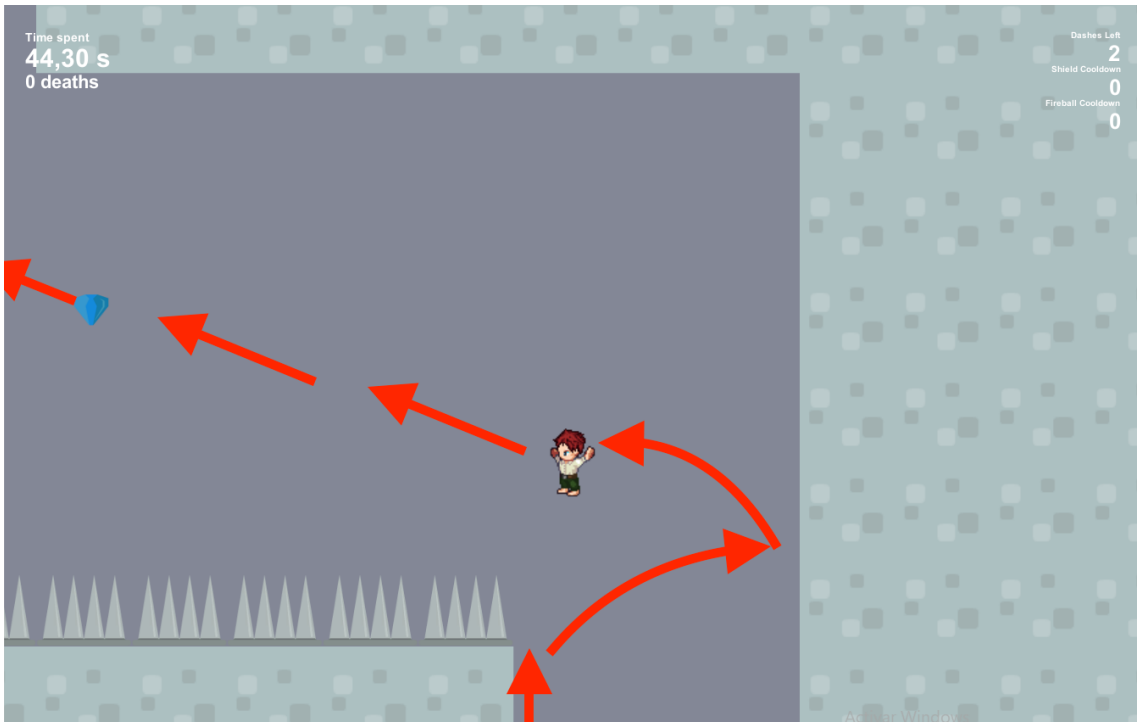

Ilustración 59 *Walkthrough* fase 16

Nos encontramos ante el último de los retos para superar este nivel. Teniendo reflejos y sabiendo la distancia que recorremos al hacer un *dash* esta fase es de lo más fácil que hay. En caso contrario, no dejaremos de morir en la caída. Se va a poner la secuencia a seguir, en forma de lista, donde cada punto equivale a una línea de pinchos, ya que es más fácil y rápido de entender.

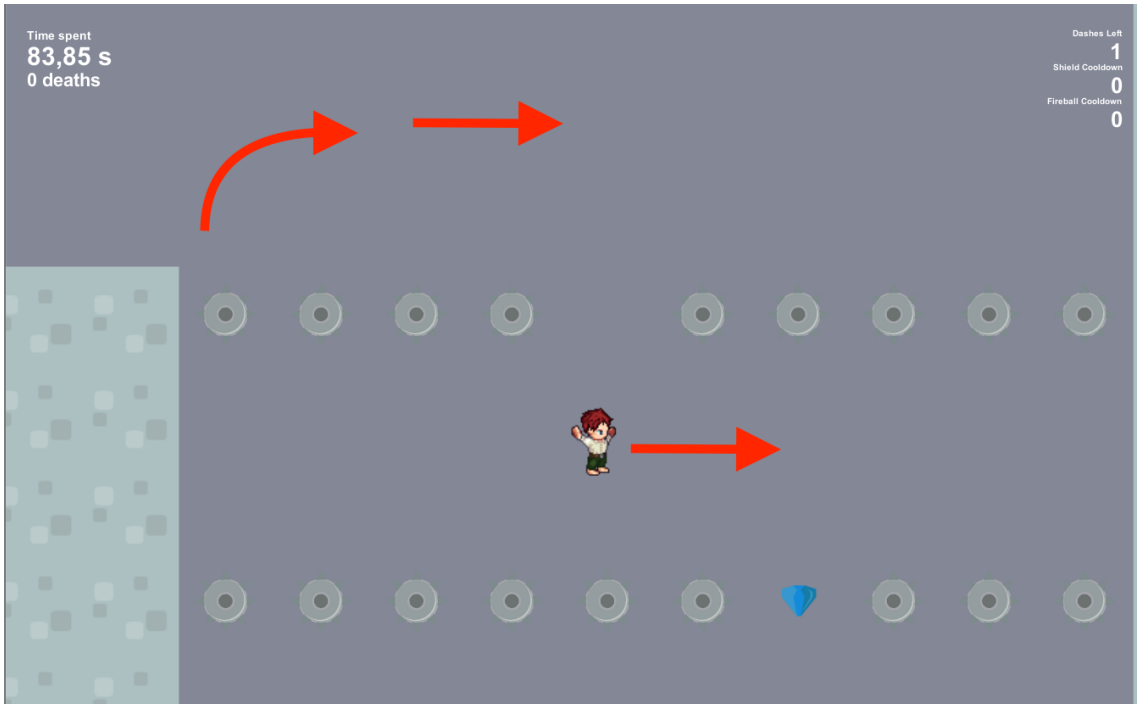

Ilustración 60 *Walkthrough* fase 17

- Salto y *dash* a la derecha
- *Dash* a la derecha
- *Dash* a la izquierda
- *Dash* a la izquierda
- Moverse a la derecha, no hacer *dash*
- Doble *dash* a la derecha
- Moverse a la izquierda, no hacer *dash*
- Doble *dash* a la izquierda
- Moverse a la derecha, no hacer *dash*
- Moverse a la izquierda, no hacer *dash*
- Moverse a la derecha, no hacer *dash*
- *Dash* a la derecha
- *Dash* a la izquierda
- *Dash* a la derecha
- *Dash* a la derecha
- *Doble dash* a la derecha

Caeremos sobre un pasillo, que lleva al final de la pantalla.

5.1 Testeo y balanceo

El diseño de los niveles no es algo que se haga sobre papel y sea fácil llevar al videojuego, sino que son fruto de muchos intentos. Todos y cada uno de los "rompecabezas" se ha debido probar cautelosamente, para asegurar que se pueden superar.

Mientras que la palabra *testeo* se refiere más a la detección de *bugs* y la solución de los mismos, la palabra balanceo se refiere a conseguir un punto intermedio en el cual los niveles no sean muy fáciles de superar ni tampoco imposibles.

5.1.1 Testo

El testeo se consigue simplemente jugando tantas veces como sea posible, intentando todas las combinaciones posibles, incluso si no tienen sentido para pasarse el nivel. Algunos de los ejemplos más comunes consisten se basan en exactamente esto, cosas que los desarrolladores no habían pensado que los jugadores hiciesen.
Como ya se ha dicho, la única forma de detectarlos es jugando. Para ello, no solo yo, el desarrollador, he jugado mi videojuego, si no que se ha presentado a un número de jugadores que han funcionado como *testers*. En un proyecto de verdad, un gran número de personas llevarían a cabo esta tarea, durante varios meses en los cuales se irán arreglando todo lo que se vaya reportando.
En este caso, aparte de mí, se ha pedido ayuda a dos personas para que testeasen el juego, siendo uno de ellos un jugador habitual, y otro un jugador ocasional, casi sin experiencia alguna.

Algunos ejemplos de bugs reportados son los siguientes:

- Se podía cambiar la dirección del *dash* si en medio de este se movía el joystick de movimiento.

- El salto parecía no funcionar del todo bien, dando la impresión que saltaba un poco más tarde de los esperado o incluso no saltaba. Se arregló cambiando el método del *FixedUpdate* a *Update.*
- Dañar al *boss* con el *dash* tenía un comportamiento extraño. El empujar hacia atrás el jugador podía interrumpirse completamente
- Morir, sobre todo en la pelea final, podía provocar que reaparecieses y que una flecha te golpease sin dar tiempo alguno a reacción.

5.1.2 Balanceo

De la misma forma que con el testeo, la única forma de asegurar que hay un balanceo en las mecánicas es jugar. Por esa misma razón, se ha vuelto a presentar el juego, modificando el comportamiento para que cada vez que se muriese, se volviese a empezar en frente del rompecabezas a testear.
Sin embargo, este testeo lleva mucho más tiempo, ya que se debe "desarrollar" distintas versiones para testear concretamente unas partes del nivel.

Por eso mismo, casi todo el balanceo se ha llevado a cabo por mí, a medida que desarrollaba el videojuego. Aunque la intención es que en mayoría de los casos solo haya una forma de superar el desnivel que se le presenta al jugador, este no debe ser fruto de azar, sino de que el jugador conozca los límites, las distancias y los movimientos de los que disponen.

Algunos ejemplos son:
- Velocidad de los pinchos.
- Distancia a la que se encuentra la siguiente plataforma.
- Cooldowns demasiado altos

6. Manual de usuario

6.1 Controles

Aunque la plataforma objetivo del juego es el ordenador, este tipo de videojuegos es mucho más fácil y placentero jugarlos con un mando o controlador. Nosotros recomendamos jugarlo con el controlador de la Xbox ya que ha sido con el que se ha efectuado el desarrollo.

Con el teclado, los controles son los siguientes: **"A"**, "**W**", "**S**" y "**D**" para moverse hacia la izquierda, arriba, abajo y a la derecha, respectivamente. Como en la inmensa mayoría de los juegos, el **espacio** se usa para realizar los saltos. La tecla **Shift** se utiliza para hacer el *dash.* El **botón izquierdo** del ratón para disparar las bolas de fuego, y el **botón derecho** del ratón para activar el escudo. Cuando se dispare una de las flechas de fuego, la posición del ratón indicara hacia donde ira la misma.
Por último, la tecla "**P**" sirve para pausar el nivel.

Todos estos controles tienen su contraparte en el controlador de la Xbox.

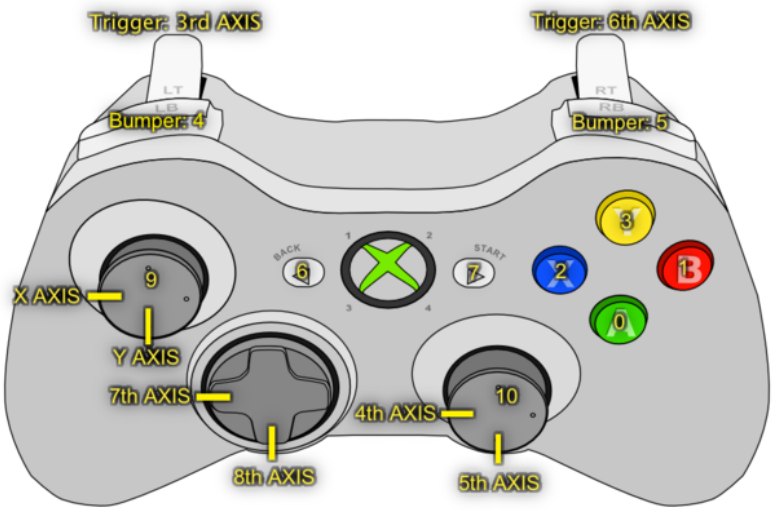

Ilustración 61 Controlador de la Xbox[15]

Como es de esperar, el **Joystick izquierdo** (X y Y AXIS) es el encargado del personaje. El botón **LB** (Bumper 4) activará el escudo y el **LT** (Trigger 3rd AXIS) disparará las bolas de fuego. Dado que cuando usamos no hay un cursor con el cual podamos apuntar, se ha usado el **Joystick derecho** (4th y 5th AXIS) para

[15] Imagen obtenida de https://wiki.unity3d.com/index.php/Xbox360Controller

calcular donde irá la bola, es decir, se apunta con este Joystick. Los otros dos gatillos que quedan se usan para el salto y el *dash*, El gatillo **RB** (Bumper 5) se usa para saltar, mientras que el **RT** (Trigger 6th AXIS) se usa para realizar los *dashes*.

7. Conclusiones

7.1 Lecciones aprendidas

Como se ha repetido varias veces en la introducción, la intención principal de este trabajo es aplicar y poner en practica todo aquello que se ha aprendido a lo largo del master. No nos referimos a la parte técnica y al uso del motor de videojuego que se haya decidido usar para realizar el proyecto, sino todo aquello previo al desarrollo, y también durante el mismo, que es necesario para realizar un videojuego.

Establecer un documento de diseño del videojuego, donde se especifique todo aquello que se quiere realizar es el primer punto que hemos tenido que tratar en este trabajo. Aunque obviamente este documento se modificará mientras se desarrolla el videojuego, tenerlo como referencia es una gran ayuda para saber qué debemos hacer y de cuánto tiempo disponemos. Cuando el desarrollo del videojuego es de una sola persona, puede no parecer tan importante, debido que dicha persona sabe lo que él desea para realizar su videojuego. Ahora bien, si este juego lo desarrollan un grupo, por pequeño que sea, que todos los integrantes tengan el mismo objetivo y tengan claro la meta es un punto muy importante para que el juego evolucione correctamente. No es el único uso de este documento. Cuando el juego se presenta a una distribuidora o al público, este documento sirve como exposición para dar a conocer como es el juego y a quien está dirigido.
Debido a la importancia de este documento, esta es tal vez la lección más importante que se ha aprendido con el desarrollo del trabajo. Pero no es el único.

Diversos componentes de *Unity* se han desarrolla en este videojuego que no se habían puesto en práctica antes: El uso de *Tiles* y *Grid* y el uso del *DirectorCinematics* son los ejemplos más claros de las lecciones aprendidas referidas en cuanto al uso de *Unity.*

7.2 Logros alcanzados

A principio de curso, se pretendía crear un videojuego del estilo plataformas 2D y, en gran parte, se ha conseguido realizar este objetivo. Se han creado diversas mecánicas que, juntándolas pueden crear un sinfín de rompecabezas y obstáculos que el jugador deberá superar, todos diferentes. Los *dashes,* los saltos contra las paredes, los diamantes, enemigos, pinchos, plataformas… Todas estas mecánicas permiten realizar múltiples combinaciones y añadir un solo elemento nuevo (por ejemplo, una plataforma que te empuja hacia un lado o hacia arriba al posarte sobre ella) incrementaría considerablemente el número de recursos que se tendría para realizar los mapas del videojuego.

Por lo general, casi todos los objeticos que se plantearon se han podido alcanzar en la medida deseada, cumpliendo con aquellos que se había establecido en el documento inicial.

Así pues, muchos de los objetivos se han podido alcanzar, muchos se han añadido a posteriori, sin estar planificados en el inicio del proyecto, ya que se veía un futuro en ellos. Pero también hay algunos que no se han podido completar. Todos estos ejemplos se comentarán en los siguientes dos puntos.

7.3 Seguimiento y análisis

El tiempo que se necesitaba para cada uno de los puntos que se querían implementar se pueden ver en el punto correspondiente[16], el 1.4 Planificación del Trabajo. Cumplir al cien por cien con lo establecido no ha sido posible, aunque sí que se han podido realizar la mayoría de las tareas y de los puntos.

No cumplir con el *planning* no significa haber eliminado cosas, sino que se han tenido que modificar los tiempos establecidos.
La búsqueda de arte no acabo el día 10/4, sino que se ha tenido que ir buscando nuevos elementos a lo largo del desarrollo del videojuego. La creación de las cinematografías y los diálogos no se presenta por primera vez en esta entrega, si no que se logró presentar en el PEC 2, avanzando así toda la planificación. Sin embargo, se añadieron nuevas mecánicas para esta entrega, como por ejemplo los diamantes que resetean los *dashes* o el poder saltar en las paredes. El tiempo que se ganó por un lado se ha podido invertir en estas mecánicas no planificadas.

Como siempre, la documentación se ha empezado más tarde de lo debido. No se ha empezado a crear este documento el día 28/4, sino a finales de marzo, sobre el 20/5. Aunque no se ha dejado para lo último, si que ha ralentizado el tiempo disponible para desarrollar los últimos elementos, como por ejemplo el *boss.*

7.4 Mejoras

Tal y como se ha comentado anteriormente, el punto que ha quedado más flojo es el *boss* final. Aunque es verdad que al principio no estaba planificado y se decidió implementar a pocas semanas de la entrega, se había pensado más elementos a implementar. Alguna de las ideas que se podrían haber implementado son:

1. Un mapa más completo sobre el cual el *boss* avance cuando le quede poca vida, con plataformas, pinchos, trampas...
2. Ataque de carga. Se abalanza sobre el jugador, con una felicidad considerable

Muchas otras mejoras a las que no se le han dado importancia se podrían hacer sobre el juego presentado. Las animaciones son una de ellas. Dado que para el trabajo cuenta más este documento y la parte técnica que no la parte del arte,

---

[16] El presentado aquí se ha modificado ligeramente al que se presentó en su momento en la PEC 2, se han añadido dos o tres cosas y cambiado alguna de las fechas.

se han dejado un poco de lado, aunque sí que se han implementado las opciones básicas, con el fin de usar todo lo que *Unity* nos proporciona.

8. Glosario

Asset: Cualquier elemento o recurso que se va a usar durante el desarrollo del videojuego. Puede ser imágenes, audio, materiales…
Background: Literalmente, fondo. En este caso, fondo de la escena del videojuego.
Boss: Jefe. Se suele usar para referirse a enemigos más fuertes que los demás, normalmente situados al final de la pantalla o en la suya propia
Bug: Errores en un software que provoca un comportamiento no esperado
Checkpoints: Traducido literalmente punto de control. En el ámbito de videojuegos, se refiere a un punto o posición donde el jugador puede volver a reaparecer cuando muere
Dash: Movimiento rápido, casi instantáneo. Durante el proceso puede hacer invulnerable al jugador o no, en este caso no.
Demo: Programa informático, en este caso, un videojuego, que es una versión reducida del programa o videojuego completo. Se suele usar como evaluación antes de su compra total.
Frame: Fotograma, una imagen en concreto del conjunto que compone la animación o el videojuego.
Framework: Un marco de trabajo, conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar. [4]
Hitbox: Espacio de colisión que tiene un objeto o personaje.
IA: Inteligencia artificial. Código que mueve a los personajes no jugables, NPCs. Suele intentar emular cómo reaccionarían los humanos, aunque no tiene por qué ser así.
Indie: De la palabra independiente. Persona o grupo que desarrolla un videojuego sin la inversión, ya sea monetaria o de marketing, de una distribuidora o de una organización con más poder adquisitivo o reputación.
NPC: Del inglés, Non-Playable Character. Jugador no controlado por una persona, sino por la IA.
Open Source: Programa cuyo código es accesible para ver, editar o redistribuir por cualquier persona.
Parallax scrolling: Técnica usada en informática donde las imágenes del fondeo se mueven más lentamente que las imágenes más cercanas a la cámara, creando una ilusión de profundidad en una escena 2D, añadiendo la sensación de inmersión en la experiencia. [5]
Raycast: Técnica que consiste en lanzar un “rayo” para detectar las colisiones con el entorno. El rayo es simplemente una línea reacta.
Script: Texto, escrito. En el ámbito de la informática, programa o secuencia de instrucciones que realizan una tarea.
Singleton: Técnica que asegura que una instancio u objeto solo existirá una vez. Útil cuando se necesita que un objeto controle a lo largo del sistema y que no se creen inconsistencias.
Software: Conjunto de informacion y de instrucciones que indican al ordenador como actuar. En este caso, nos referimos al videojuego
Sprite: Imagen.

Static: Indica que dicha función o variable no es única de cada instancia de este objeto, si no que se comparten entre ellos. De nuevo, es útil para compartir información entre todos los objetos de la misma clase.
Tester: Persona cuyo trabajo es jugar al juego, para poder reportar todos los errores que encuentre
TFM: Trabajo de Final de Master
Unity: Motor de videojuego desarrollado por *Unity Technologies.*
Unreal Engine: Motor de videojuego desarrollo por *Epic Games*
Walkthrough: Tour o demostración, en este caso, del nivel. No tiene que ser exactamente una explicación de cómo hacerlo o superarlo, sino mostrar como es.

Bibliografía

[1] Wikipedia, «Wikipedia,» 4 Junio 2019. [En línea]. Available: https://en.wikipedia.org/wiki/List_of_Unreal_Engine_games. [Último acceso: 10 Junio 2019].

[2] Unity3D, «Unity3D,» Diciembre 2018. [En línea]. Available: https://unity3d.com/how-to/architect-with-scriptable-objects. [Último acceso: 2 Junio 2019].

[3] R. T. Bonet, «Gamasutra,» 7 Marzo 2018. [En línea]. Available: https://www.gamasutra.com/blogs/RubenTorresBonet/20180703/316442/A_better_architecture_for_Unity_projects.php. [Último acceso: 2 Junio 2019].

[4] Wikipedia, «Wikipeda,» 17 Mayo 2019. [En línea]. Available: https://es.wikipedia.org/wiki/Framework. [Último acceso: 28 Mayo 2019].

[5] Wikipedia, «Wikipedia,» 25 Enero 2019. [En línea]. Available: https://en.wikipedia.org/wiki/Parallax_scrolling. [Último acceso: 14 Junio 2019].