SULTAN'S JOURNEY

2 de enero de 2022

Autor: Aaron Escobosa Martín

Grado en ingeniería Informática

TFG-Videojuegos

Consultor: Albert Sánchez Amo

Profesor: Joan Arnedo Moreno

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Sultan's Journey*
Nombre del autor:	*Aaron Escobosa Martín*
Nombre del consultor/a:	*Albert Sánchez Amo*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega (mm/aaaa):	*01/2022*
Titulación o programa:	*Grado en ingeniería informática*
Área del Trabajo Final:	*Videojuegos*
Idioma del trabajo:	*Castellano*
Palabras clave	*Plataformas, videojuegos, 2D*

Resumen del Trabajo (máximo 250 palabras):

El objetivo de este proyecto, es la creación de un videojuego 2D de temática aventura-plataformas clásico, abarcando todas las etapas del proceso de desarrollo, desde la mera fase de conceptualización y diseño, hasta la propia implementación de todo esto a nivel artístico y de programación.

El producto, será generado con el motor de desarrollo de videojuegos Unity y correrá como aplicación de escritorio sobre sistemas operativos Windows.

Dada la multidisciplinariedad requerida para la elaboración de este tipo de productos, se ha considerado el desarrollo de un videojuego como una forma ideal de concluir el grado, ya que ha permitido tanto poner en práctica muchas de las habilidades adquiridas durante el transcurso de este, como combinarlas con muchas otras ramas propias del proceso creativo.

Pese a que se ha diseñado un guion y trama pensados para un juego completo, por motivo del corto ciclo de vida del que se dispone para la elaboración del proyecto al estar este ideado para durar solo un semestre y las barreras que esto supone, se ha limitado la ejecución a una demo jugable y funcional en la que solo se han implementado parte de la totalidad de los elementos planteados.

No obstante, dado que se tiene intención de seguir con el proyecto por cuenta propia, se ha tratado de mantener el diseño tanto a nivel técnico como artístico lo más aprovechable y modular posible, del mismo modo, pese a que en algunos casos se han sentado bases para su desarrollo, se han reservado las implementaciones restantes para líneas de trabajo futuras.

Abstract (in English, 250 words or less):

The main purpose of this project, is the creation of a classical 2D adventure platformer based video game while passing through all the game development states.
From mere design and conceptualization to more elevated stages like implementing and bringing to life all of those concepts, technically and artistically.

Final product, would be developed using Unity game engine as the main tool in order to obtain an executable desktop application that will run on Windows based PCs.

Due to the big amount of disciplines involved in the creation of such products, developing a video game has been chosen as the ideal way to conclude my bachelors degree. Mainly, because it's creation, allowed me to put into use most of the skills acquired during all those years while combining them with many other capabilities, to be precise, those usually related to artistic creations.

Game plot and main story, have been designed to fit into a full videogame, but as a result of the short development stage life cycle and it's time restrictions, mainly due to the project being designed to have duration comprehended into an academic semester, the product has been limited to a functional and playable demo.

Anyway, after completing this first stage, the main goal is to continue expanding the game. In order to facilitate this, game structure, logic and implementation have been designed to be as modular and reusable as possible.
Some of the not implemented elements have been partially designed and created nonetheless, but it's crucial parts will be included on future updates.

Índice

Índice de figuras

Índice de tablas

Dedicatoria

En primer lugar, quisiera dedicar este trabajo a mis padres, por la inmensa paciencia, ternura, amor incondicional y cariño que han demostrado durante toda mi vida. Por ser ese espejo en el que siempre he querido verme reflejado, y por haber dado forma a la persona que soy.

A mi tía Severina, por lo mucho que nos hubiese gustado a los dos celebrar esta victoria, pero, sobre todo, por haber sido ese faro que me llevó a escoger andar este camino profesional que tanto adoro.

A mi mujer Meritxell, por su infinita comprensión desde que empecé esta aventura, por ser un apoyo siempre presente en los buenos y en los malos momentos, y por empujarme a tratar de ser mejor cada día.

A mi gran amigo Jose, por su paciencia y las incontables horas de su tiempo, que altruistamente, ha decidido invertir en ayudarme a hacer más liviana esta carrera de obstáculos.

Por último, a mi perro Sultán, tanto por el tiempo que disfrutamos juntos, como por haber sido el mejor compañero que he podido tener. Pero, ante todo, por doler tanto, tanto como para obligarme a convertir ese dolor en este humilde proyecto.

Este éxito, como todos los que alcance, no es realmente mío, sino vuestro, ya que, sin teneros a mi lado, jamás hubiese llegado donde estoy.

"Aquí reposan los restos de una criatura que fue bella sin vanidad, fuerte sin insolencia, valiente sin ferocidad y tuvo todas las virtudes del hombre y ninguno de sus defectos.

Epitafio a un perro - Lord Byron (1808)

"Se puede vivir sin perro, pero no merece la pena".

Heinz Rühmann.

1. Introducción

1.1 Contexto y justificación del Trabajo

El motivo principal del desarrollo de este proyecto recae sobre dos elementos clave, ambos ligados a mi desde muy pequeño. Por una parte, la pasión por los videojuegos, que como indico, manifesté ya a muy temprana edad, habiendo marcado estos en gran medida mi infancia y desarrollo personal.

En segundo lugar, mi amor incondicional por los perros, animales siempre presentes en mi vida y con los que he convivido desde mis primeros días hasta la actualidad.

El mundo del videojuego ha evolucionado a un ritmo vertiginoso desde mi primer contacto con él a mitad de la década de los 90. Han pasado de estar considerados como un simple pasatiempo absurdo por gran parte de la sociedad, a convertirse en un fenómeno social a la vez que, en una poderosa herramienta, usada en ámbitos tan diversos como la pedagogía, la simulación o incluso como una excelente herramienta narrativa audiovisual.

Cabe destacar también, que los motores de desarrollo actuales han democratizado muchísimo la industria, brindando a casi cualquier usuario con unos conocimientos muy básicos, la capacidad de crear su propio juego. Una clara consecuencia de esto es el “boom” de los juegos Indie que se está viviendo durante los últimos años, llegando estos en muchas ocasiones a cotas de calidad muy elevadas, que les permiten competir cara a cara con videojuegos AAA.

No obstante, no hay que olvidar que el desarrollo de un videojuego actual es un proyecto altamente transversal e interdisciplinar. En consecuencia, desarrollarlo solo requiere prestar atención y cuidado en aspectos muy diversos, desde la parte más matemática o de programación, hasta los aspectos de vertientes más artísticas como el sonido o el aspecto visual.

Concretamente, todo este proyecto se ha visto motivado por la reciente pérdida de mi perro Sultán, tras tres años muy duros combatiendo contra la leishmaniasis.

Por este motivo, decidí aprovechar momento tan remarcable como es el finalizar el grado, para rendirle tributo con la creación de este pequeño videojuego.

1.2 Objetivos del Trabajo

- Aprender a usar de forma correcta y eficaz el motor de juegos Unity
- Diseñar y crear los assets del videojuego, tanto a nivel audiovisual como de código.
- Planificar de inicio a fin el desarrollo del videojuego. Desde el momento en que se instala Unity hasta que el juego sea publicado.
- Mejorar las habilidades de edición gráfica, edición de audio y de ilustración en general.
- Poner en práctica todos los conocimientos y habilidades adquiridos a lo largo del grado.
- Adquirir soltura con el lenguaje C#, con el cual no se había interactuado hasta el momento.
- Iniciarse en el mundo del desarrollo de videojuegos con la intención de lanzar el juego completo en un futuro.
- Rendir tributo a un gran compañero.

1.3 Enfoque y método seguido

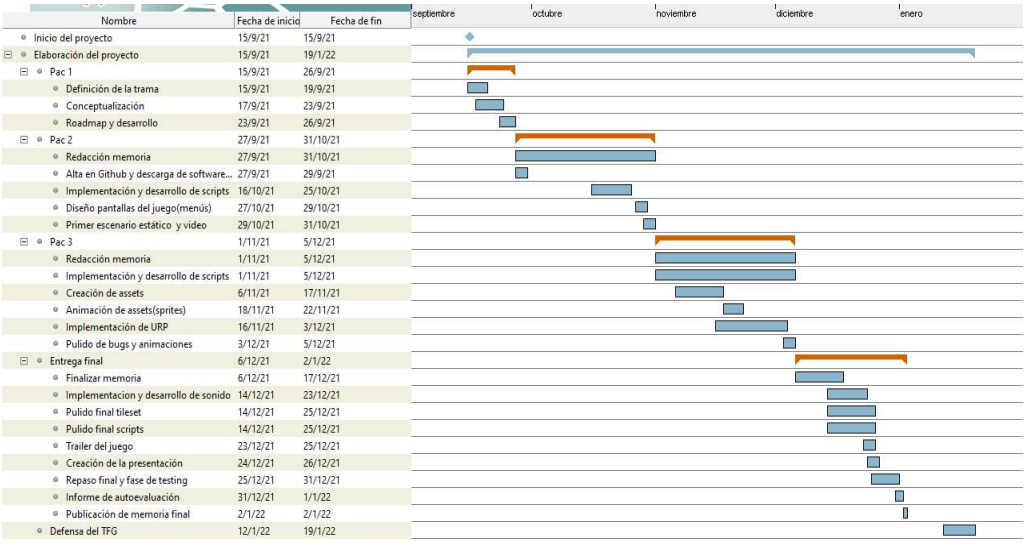
Respecto a la metodología usada para llevar a cabo el proyecto, se ha optado por una estrategia estilo “from scratch”, es decir que, pese a contadas excepciones, se han generado desde cero todas y cada una de las partes del videojuego evitando siempre el uso de material de terceros.

Este enfoque, no era el más adecuado en términos de pura eficiencia a causa del acotado margen temporal y la escasa experiencia en muchos de los ámbitos, pero dado la gran carga emocional que suponía el hilo argumental y la temática del juego en general, se ha optado implicarse de forma personal en cada uno de los apartados por tal de conseguir volcar en cada uno de ellos el esmero y cariño deseados.

1.4 Planificación del Trabajo

Para poder planificar el desarrollo del videojuego, se realizó en la PAC 1 un diagrama de Gantt, que dividía y gestionaba las diferentes tareas correspondientes a la implementación y desarrollo del proyecto entre las diversas entregas parciales o PAC y el calendario establecido para estas.

No obstante, este diagrama sufrió algunos cambios a lo largo de la vida del proyecto y se adjunta su versión final y actualizada



Nombre	Fecha de inicio	Fecha de fin
Inicio del proyecto	15/9/21	15/9/21
Elaboración del proyecto	15/9/21	19/1/22
Pac 1	15/9/21	26/9/21
Definición de la trama	15/9/21	19/9/21
Conceptualización	17/9/21	23/9/21
Roadmap y desarrollo	23/9/21	26/9/21
Pac 2	27/9/21	31/10/21
Redacción memoria	27/9/21	31/10/21
Alta en Github y descarga de software...	27/9/21	29/9/21
Implementación y desarrollo de scripts	16/10/21	25/10/21
Diseño pantallas del juego(menús)	27/10/21	29/10/21
Primer escenario estático y video	29/10/21	31/10/21
Pac 3	1/11/21	5/12/21
Redacción memoria	1/11/21	5/12/21
Implementación y desarrollo de scripts	1/11/21	5/12/21
Creación de assets	6/11/21	17/11/21
Animación de assets(sprites)	18/11/21	22/11/21
Implementación de URP	16/11/21	3/12/21
Pulido de bugs y animaciones	3/12/21	5/12/21
Entrega final	6/12/21	2/1/22
Finalizar memoria	6/12/21	17/12/21
Implementacion y desarrollo de sonido	14/12/21	23/12/21
Pulido final tileset	14/12/21	25/12/21
Pulido final scripts	14/12/21	25/12/21
Trailer del juego	23/12/21	25/12/21
Creación de la presentación	24/12/21	26/12/21
Repaso final y fase de testing	25/12/21	31/12/21
Informe de autoevaluación	31/12/21	1/1/22
Publicación de memoria final	2/1/22	2/1/22
Defensa del TFG	12/1/22	19/1/22

Figura 1 Diagrama de Gantt

1.5 Breve sumario de productos obtenidos

Una vez finalizado el trabajo este es el compendio de productos obtenidos:

- Un ejecutable para PC de la versión final de la demo.
- Memoria del proyecto.
- Videos requeridos en las distintas entregas parciales (PAC 2 y PAC3).
- Video de defensa del proyecto.
- Video tráiler del juego.
- Informe de autoevaluación.
- Repositorio en GitHub con todo el código fuente del juego.

1.6 Breve descripción del resto de capítulos de la memoria

Esta memoria está compuesta por 6 capítulos adicionales al de introducción, que abordan los siguientes aspectos:

1.6.1 Capítulo 2: Estado del arte

En este capítulo se realiza una revisión del género del videojuego a la vez que se analizan diversos entornos de desarrollo.

1.6.2 Capítulo 3: Definición del juego

Este capítulo será usado para documentar los aspectos de diseño creativo del juego, a la vez que se muestran referencias a juegos que han sido usados como inspiración. En esencia podríamos decir que se trata del GDD (game design document).

Adicionalmente, se incluye un pequeño resumen económico sobre los costes estimados de desarrollo.

1.6.3 Capítulo 4: Diseño técnico

Capítulo destinado a la documentación técnica del proyecto. En él, se explican al detalle los diferentes componentes del videojuego, como interactúan estos entre sí, se detallan los requisitos técnicos de desarrollo, se diseccionan todos los assets, se detalla el uso de URP, las animaciones de los distintos elementos, y finalmente, se hace una relación de las herramientas utilizadas.

1.6.4 Capítulo 5: Diseño de niveles

Este capítulo consiste en una pequeña guía sobre el nivel desarrollado para la demo, en él se detalla la ruta óptima a seguir por el jugador con tal de superar el nivel con éxito y se hacen algunas aclaraciones de diseño.

1.6.5 Capítulo 6: Manual de usuario

El manual de usuario, nos sirve como punto de partida para referenciar los requisitos mínimos para correr el juego y como guía explicativa de los diferentes controles y mecánicas del juego.

1.6.6 Capítulo 7: Conclusiones

El capítulo de conclusiones cierra esta memoria con una reflexión final del proyecto. Adicionalmente, en esta se detallan los aspectos del juego pendientes de desarrollo en un futuro.

2. Estado del arte

2.1 Revisión del género

Sultan's Journey, es un juego de aventura y plataformas 2D.

Los videojuegos de plataformas surgieron en un panorama con claras limitaciones de hardware, lo cual solo permitía diseños de niveles más bien breves y con un numero finito de elementos. De ahí que se usasen plataformas voladoras para diseñar los escenarios, algo simple pero efectivo en cuanto a coste computacional.

Por otra parte, los videojuegos de aventuras, por motivos similares a los de plataformas, empezaron ciñéndose más a las aventuras gráficas, donde el control del jugador sobre el personaje principal era más limitado.

A finales de los 80, las mejoras tecnológicas y la aparición de consolas de 8 bits como NES, permitieron la creación de juegos de plataformas donde los escenarios no se veían tan comprometidos por las limitaciones físicas de computación, pudiendo actuar como un elemento narrativo.

Posteriormente, en la década de los 80 y 90, su popularidad aumento gracias a títulos clásicos como el conocidísimo "Super Mario Bros" o los primeros "Ryman".

Esto dio lugar a grandes títulos como el anteriormente citado "Super Mario Bros" o "Kid Icarus".

Con el paso del tiempo y en parte también gracias a la aparición de los primeros videojuegos 3D, este género híbrido afianzó su dominio como uno de los más populares durante muchos años, aumentando de forma casi constante la lista de clásicos que lo componen.

Destacamos también a parte de las ya citadas, sagas como “Spyro the Dragon”, “Crash Bandicoot”, “Sonic the Hedgehog”, “Donkey Kong” y “Ghosts ’n Goblins” entre otros.

En la actualidad, los juegos de aventura y plataformas 2D han recibido una gran cantidad de títulos de gran calidad como “Ori and The Blind Forest”, “Celeste”, “Cuphead”, “Limbo”, “Gris” o “Unravel”, pese a que este último, no podríamos catalogarlo estrictamente como un videojuego 2D.

Tal como nos indica el término 2D, esto implica que el juego se desarrolla en un solo plano sin eje Z, donde tanto el personaje como la cámara se mueven siempre en los ejes X e Y, no obstante, el caso de “Unravel” es como decíamos algo peculiar, ya que realmente se trata de un juego modelado en 3D pero con un scroll de cámara lateral, siendo este tipo de juegos popularmente conocidos como 2,5D.

2.2 Revisión de la tecnología

Tal como se ha venido citando a lo largo del primer y segundo capítulo, la democratización del mundo del desarrollo de videojuegos, así como la popularidad actual de los juegos indie (especialmente de aquellos de tipo 2D) es tal, que podemos encontrar en el mercado infinidad de herramientas y motores su desarrollo en general, siendo los más populares los que se relacionan a continuación:

- Unity: Este motor, usa C# como lenguaje de programación base y es ampliamente usado en la industria del videojuego, tanto en la escena independiente como la profesional. Podríamos llegar a decir que es casi la navaja suiza del desarrollo de videojuegos dado su enorme abanico de posibilidades.

Uno de sus grandes puntos fuertes, es la ingente cantidad de documentación existente respecto a su uso, tanto a nivel de tutoriales como de troubleshoting en general, reduciendo así la probabilidad de que el proyecto pase por puntos de estancamiento debidos a dificultades técnicas sin solución aparente. No obstante, tanto su curva de aprendizaje como su nivel técnico son algo más elevados que en algunos de sus homónimos dada su complejidad.

Cabe destacar también que cuenta con un plan pro (gratuito para estudiantes que acrediten dicha condición) y con uno gratuito muy completo para desarrolladores más pequeños, siempre que estos no hayan obtenido ingresos superiores a los 100.000$ en los últimos 12 meses

Figura 2 Logo Unity

- GameMaker Studio 2: Otro de los motores más usados, especialmente para juegos 2D. Su principal ventaja radica en la facilidad que brinda a nivel de desarrollo, evita gran parte de la programación y la mayoría de sus funciones son simplemente “drag & drop”. No obstante, tiene algunas limitaciones en cuanto a estabilidad o al producto final, especialmente a la hora de exportar.

  Esta plataforma, está basada en un lenguaje de programación interpretado llamado GML(GameMaker Language) y un kit de desarrollo de software.

Figura 3 Logo GameMakerStudio 2

- Godot: este motor, pese a que más pequeño que Unity aspira a competir directamente con él y ha ganado mucha popularidad en los últimos años.

  Se trata de un motor OpenSource, con una curva de aprendizaje teóricamente mucho más reducida. Su principal inconveniente, radica en que pese a ser muy similar a phyton, hace uso de un lenguaje de programación de alto nivel propio GDScript(Godot Script), requiriendo de aprender un lenguaje nuevo a aquellos que quieran desarrollar proyectos con este motor.

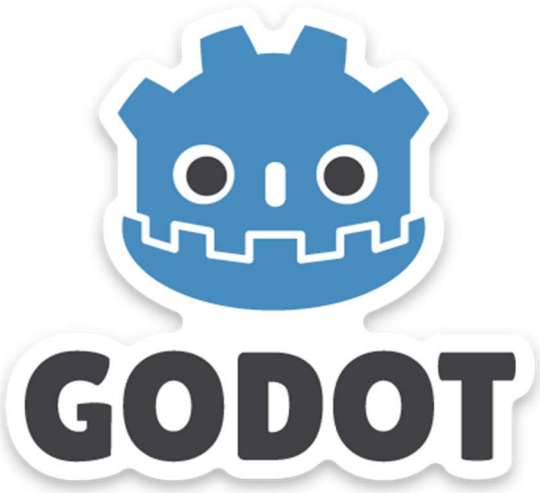


Figura 4 Logo de Godot

3. Definición del juego

3.1 Idea del juego

3.1.1 Breve descripción del juego

"Sultán's Journey", pretende ser un videojuego plataformas 2D de scroll lateral de los que deja huella. La idea principal era que, dadas las limitaciones de tiempo impuestas por el proyecto, este siguiese una línea de Trabajo lo más simple posible y ampliarlo por cuenta propia en un futuro.

En esta aventura, el jugador encarnará a un perro llamado Sultán a lo largo de todo su ciclo vital hasta su inevitable fin a causa de una enfermedad mortal. Por el camino deberá hacer frente a numerosos obstáculos y hacer uso de diversos power-ups para poder hacerles frente.

3.1.2 Subgénero y referencias a videojuegos existentes

En un inicio, se plantearon dos vertientes para este aspecto en el videojuego:

Por un lado, una de ellas era enfocarlo como metroidvania clásico (juegos de acción aventuras) y tomando como principal fuente de inspiración títulos como "Hollow Knight" u "Ori and the Blind Forest".

En este supuesto, el personaje principal avanzaría por distintos escenarios haciendo frente a cierta cantidad de enemigos que entorpecerían en mayor o menor medida que alcanzase su objetivo a la vez que se obtenían ciertos upgrades o mejoras permanentes.

Figura 5 Hollow Knight

No obstante, esta opción decidió abandonarse al principio del desarrollo, tanto por su mayor nivel de complejidad técnica (especialmente en cuanto al tamaño del nivel), como al echo de que era un estilo que no acababa de encajar con el hilo narrativo del juego

Finalmente, se optó por enfocarlo como un juego que pese a tener elementos de plataformas, se centrase más en su vertiente de aventuras y por ende en la historia a contar.

La diferencia principal entre estos dos enfoques radicaba en la gestión de los enemigos en el juego, ya que mientras que en el primero hubiesen sido elementos con los que el jugador se viese forzado a interactuar “matándolos”, en el segundo estos pasaban a ser elementos pasivos que pueblan el escenario y actúan como simples obstáculos a evitar.

Las referencias principales fueron videojuegos como “Gris”, “Unravel” o bien “Limbo”, cuyo objetivo final simplemente es avanzar por el escenario para poder avanzar más en la historia.

Figura 6 Gris

Alternativamente, encontramos títulos que también han sido usados como fuente de inspiración pese a presentar un esquema híbrido entre estas dos vertientes, como por ejemplo “Braid”

Figura 7 Braid

En cualquier caso, la idea principal del título es contar una historia con un elevado nivel de carga emocional, y para ello se ha pretendido hacer uso principalmente del aspecto visual del juego.

De hecho, no es difícil darse cuenta de que los 6 títulos usados como referencia tienen un aspecto gráfico muy cuidado (el cual podríamos catalogar de ciertamente bonito) y que resulta de gran relevancia para la historia.

3.1.3 Tipo de interacción juego-jugador

Como en todo juego plataformas, el jugador deberá mover de forma tanto horizontal como vertical al personaje a la vez que esquiva a los enemigos, haciendo uso de una serie de plataformas y objetos dispuestos por el escenario con tal de poder avanzar.

Tal como se ha comentado anteriormente, los enemigos actúan como meros elementos pasivos que pueblan el escenario. Algunos lo hacen de forma activa, es decir deambulando por el este y eliminando al jugador si los toca, y otros como zonas insalvables hasta que se obtenga el objeto adecuado (barreras que eliminan al personaje al entrar en contacto con ellas).

También se planteó incluir enemigos que actuasen a modo de "némesis" (el conocido personaje de la saga "Resident Evil"), siendo la única opción del jugador en este caso, el hacer uso de los elementos del escenario o bien de ciertos objetos para ocultarse/evitarlos. Finalmente, se decidió posponer su implementación para versiones futuras del juego ajenas a este proyecto dada la complejidad técnica de este tipo de IA.

Figura 8 Némesis

Un gran ejemplo de este modo de gestionar las interacciones con los enemigos, sería “Little Nightmares”, en el que el jugador es constantemente perseguido por estos, pero no tiene ninguna capacidad de atacarles o eliminarlos.

Figura 9 Jugador escondido en Little Nightmares

3.1.4 Plataforma de destino

Tal como se ha indicado en diversas ocasiones a lo largo de los anteriores capítulos, el tiempo del que se dispone para desarrollar el proyecto ha resultado un factor muy limitante en muchos aspectos, por ello se ha decidido centrar los esfuerzos en el desarrollo para PC a nivel Desktop.

No obstante, de cara a un futuro se pretende implementar controles con mando y no se descarta la opción de portarlo a otras plataformas como videoconsolas, lo que si queda descartado en principio sería el desarrollo para smartphone.

3.2 Conceptualización

3.2.1 Historia, trama y ambientación

Toda la trama, gira en torno al personaje principal, Sultán, un pastor alemán usado como perro guardián en una fábrica. Posteriormente, será adoptado a causa del cierre de esta y enfermará al ser contagiado de leishmaniasis.

La historia ha supuesto una gran implicación personal, puesto que será el vivo relato de la vida de Sultán, mi perro, del que nos vimos obligados a despedirnos al finalizar el verano.

El juego final, será narrado en tres actos, pese a que para este proyecto se ha acotado una demo que engloba parte del primero y del segundo, aunando mecánicas de ambos y dejando otras por implementar.

El inicio o prólogo (que será el más breve de ellos), cubrirá la vida de Sultán desde su nacimiento hasta que se convierte en un perro adulto. Durante este acto, manejaremos en primera instancia a un cachorro que estará acompañado en todo momento de sus hermanos y su madre, los cuales irán desapareciendo de forma paulatina según avancemos por este capítulo y Sultán crece. De esta manera se pretende representar como pese a que al inicio eran varios perros, debido a los problemas económicos que sufrió la fábrica, se fueron regalando y terminó por ser el único perro que quedó.

El segundo acto, comprenderá desde el momento en que la fábrica decide dar el cierre, por ende, nuestro protagonista será adoptado por una familia cuando ya parecía no quedar otro futuro que la perrera para él. Así mismo, será aquí donde se marcará el punto de inflexión de la narrativa y se producirá tanto el contagio como el cambio de escenario debido a la adopción. Hay que destacar, que este será el más largo de los tres actos y en el que se centrará el grueso del gameplay.

Según avancemos en este acto, veremos reflejado un deterioro progresivo del personaje a nivel físico, concretamente estoque pretende ser implementado como una mecánica que añada un delay sobre los controles entorpeciendo el gameplay.

El último acto, narrará los últimos meses de vida de Sultán y su lucha contra la enfermedad hasta su irremediable colapso, momento en el que enfrentaremos al boss final, tras lo cual se dará lugar a una última cinemática en que se despedirá de sus cuidadores y volverá a reunirse con su madre y sus hermanos, tras esto, el juego termina dando paso a la escena de créditos.

Podemos apreciar como la trama sigue claramente el esquema del monomito o periplo del héroe de Campbell. Pese a no contar con sus diecisiete etapas, identificamos claramente la salida en el prólogo, la iniciación en el segundo acto del juego y finalmente, el regreso está claramente presente en el último acto, momento en el que sultán “vuelve” a su punto de partida.

3.2.2 Definición de personajes y elementos

Pese a que en un inicio se presentó un esquema de personajes e interacciones notablemente más amplio, y que cubría la totalidad de actores y elementos del juego completo, para esta demo solo aplican los siguientes:

- **Fábrica:** será el escenario sobre el que se desarrollará este acto
- **Maquinaria:** elemento decorativo del escenario y como plataformas móviles de este.
- **Caseta:** será el punto de control.
- **Madre de sultán:** estará presente en los puntos de guardado y aullará para indicarnos que hemos colisionado con ellos.
- **Almacén de la fábrica:** hace las veces de cueva y se presenta como una parte inferior del mapa poco iluminada
- **Lámparas:** Ayudan a iluminar la parte inferior, se mueven de lado a lado describiendo un arco de 45 grados y parpadean de forma intermitente.

- **Muebles:** actuaran como decoraciones de la parte baja del escenario
- **Otros perros:** enemigo que patrulla ciertas zonas del mapa por tierra.
- **Mosquitos:** enemigo que patrulla ciertas zonas del mapa por aire
- **Leishmania:** actúa como barrera disipable que bloquea el progreso al jugador hasta conseguir el objeto adecuado.
- **Medicación:** actúa a modo de llave y permite abrir las barreras de leishmania, siempre que estas coincidan con su color.
- **Pipetas:** vuelve al personaje invulnerable durante cierto periodo de tiempo. La decisión de incluir este elemento y su funcionamiento no resulta fortuita, pretende mostrar dos elementos clave. Por una parte, como aun haciendo invulnerable al jugador, los enemigos no dejan de estar presentes en ningún momento y solo se trata de una solución temporal al problema.

  Por otra parte, se pretende plasmar mediante duraciones asignadas relativamente breves, el corto periodo de efectividad de estos productos.
- **Moriarty:** gato atigrado a cargo del tutorial del juego y de comunicarnos cuando hemos completado el nivel.
- **Comida:** aumentará la velocidad de movimiento temporalmente. Pretende plasmar el “boost” de energía obtenido tras alimentarse.

- **Snacks:** aumentarán la capacidad de salto. Pretende representar los típicos saltos de alegría que daba Sultán incluso en sus peores momentos físicos al ser premiado con estos.

3.2.3 Interacciones entre los actores del juego

A parte del propio protagonista, podemos apreciar tres claros tipos de actores, donde cada uno de ellos tiene un tipo de interacción básica asociada.

- **Enemigos:** patrullan y pueblan el escenario, actúan como obstáculos o barreras que eliminan al jugador al colisionar con ellos.
- **Objetos:** se encuentran distribuidos por el escenario y otorgan beneficios al jugador, ya sea disipando barreras o proporcionándole mejoras temporales.
- **Aliados:** aquí encontramos a Moriarty (el gato) y a la madre de Sultán, ambos ayudan al jugador de forma directa, ya sea indicándole la utilidad de los objetos y los controles (Moriarty en el tutorial) o bien indicando cuando hemos pasado por los checkpoints.

3.2.4 Objetivos planteados al jugador

Los objetivos planteados al jugador en esta demo son muy sencillos, este debe avanzar por el mapa evitando colisionar con los enemigos y disipando las distintas barreras por tal de finalizar el nivel.

Tanto los checkpoints como los distintos objetos, facilitarán esta tarea si son usados de la forma correcta, especialmente los powerups, que deberán ser usados ciñéndose a un buen tempo.

Por otra parte, existen pequeñas trampas para atrapar al jugador, como barreras disipables extra, en las que, si el jugador “gasta” una llave, luego no podrá disipar la correcta y se verá obligado a tomar una ruta más difícil o simplemente rehacer ese trozo de mapa.

3.2.5 Concept art

Figura 10 Boceto Sultán

Figura 11 Boceto primer acto

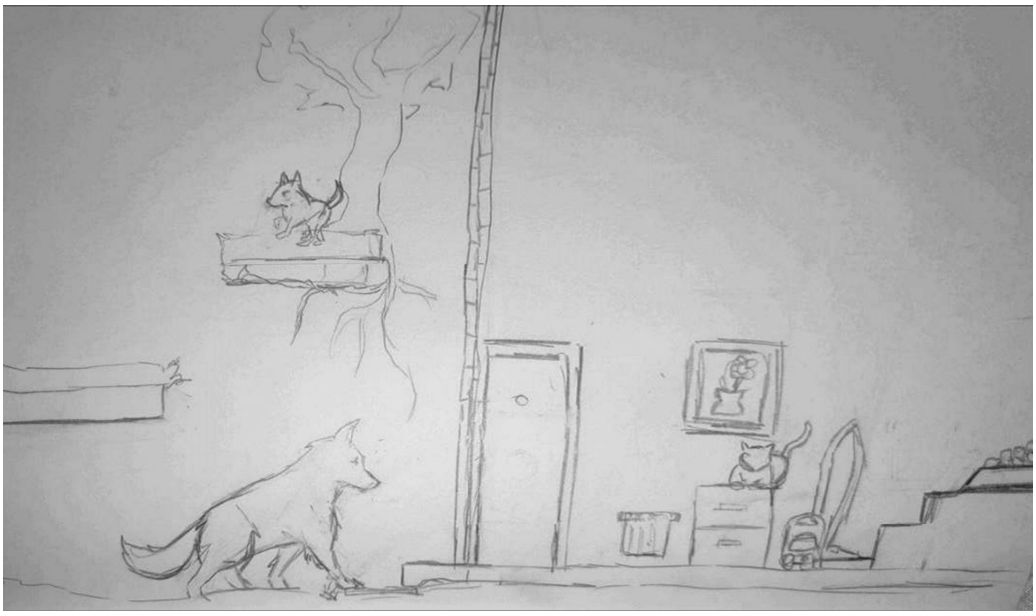


Figura 12 Boceto segundo acto

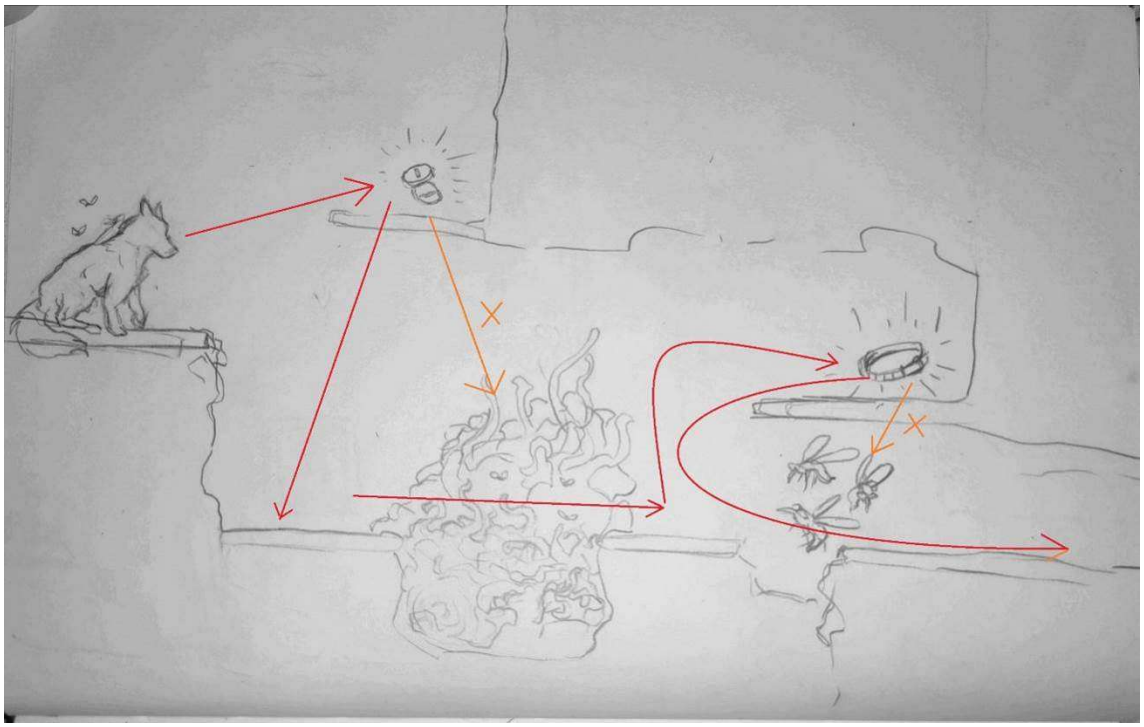


Figura 13 Boceto nivel

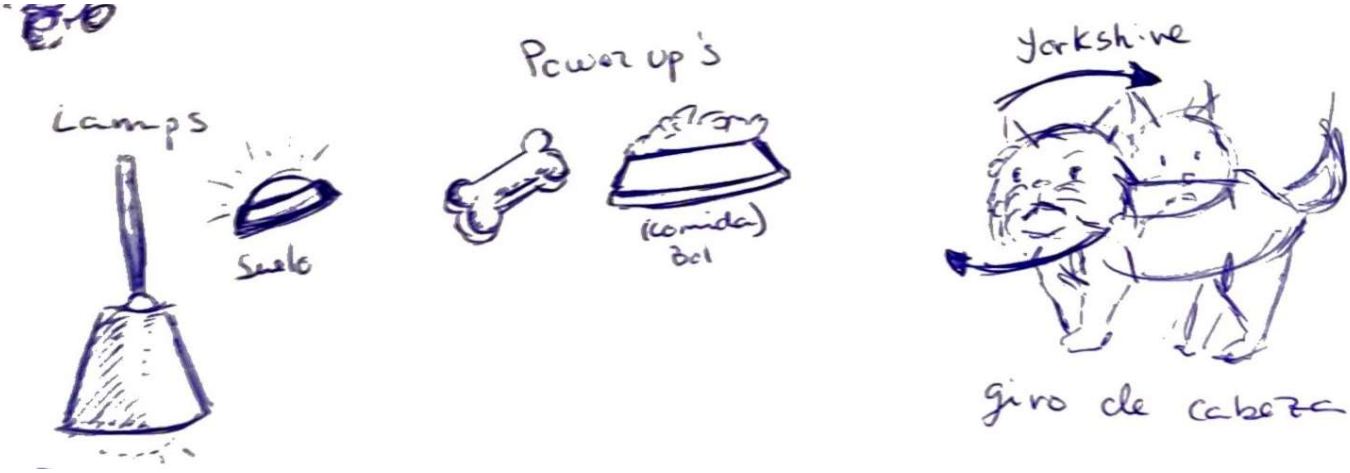


Figura 14 Boceto enemigo, powerups e iluminación

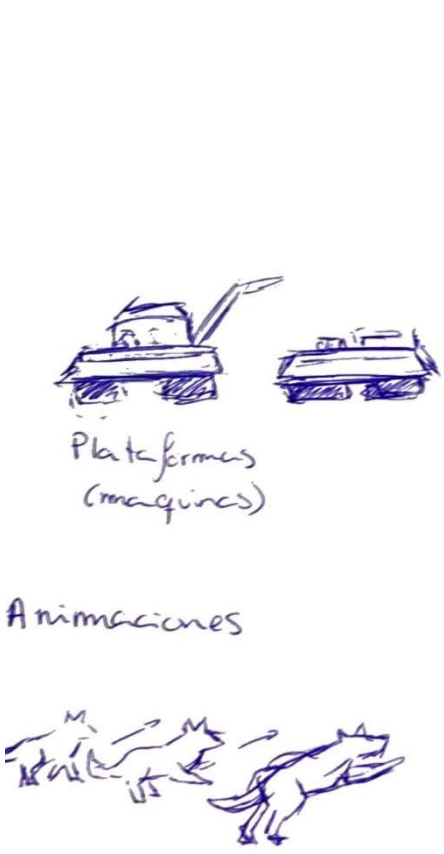


Figura 16 Boceto animación de salto y plataformas

Figura 15 Boceto mapa

3.3 Desarrollo y roadmap

3.3.1 Evaluación de motores y kits de desarrollo

En el apartado 2.2 de la presente memoria, se han analizado ya los motores más populares (Unity, Godot y GameMaker 2). Finalmente, para el desarrollo de este proyecto se ha optado por el uso de Unity.

El motivo principal que sustenta esta decisión es que, dada la nula experiencia en el campo del desarrollo de videojuegos, se ha preferido usar aquel motor del que se disponga de mayor cantidad de documentación y tutoriales en la red.

3.3.2 Planificación de objetivos

Pese a que en el apartado 1.4 de la presente memoria ya se ha incluido un diagrama de Gantt con la planificación temporal y de recursos a modo de roadmap, se adjunta aquí un breve resumen.

El objetivo final del proyecto es tener lista una demo que como hemos dicho, aúne mecánicas tanto del primer acto como del segundo.

Se hará especial hincapié, en el apartado artístico del videojuego ya que se pretende que sea el más notable de todos, pese a que en ningún momento se descuidarán otros más técnicos, a fin de cuentas, el presente proyecto no deja de ser uno de desarrollo de software.

Se implementarán menús, tanto uno principal como uno de pausa que permitan gestionar el flujo de la aplicación en todo momento (salir de ella, volver al menú principal, etc) y ajustar sus opciones.

Se implementará también una UI que nos informe en todo momento de cuáles son los objetos de tipo power-up que tenemos activos, así como las pastillas(llaves) que poseemos.

Finalmente, serán implementados todos los personajes y elementos del juego anteriormente mencionados, así como los correspondientes efectos de audio (tanto a nivel de SFX como de banda sonora del juego).

3.4 Costes derivados del desarrollo del proyecto

Para finalizar este capítulo y tal como se había indicado en el resumen, se adjunta un pequeño desglose económico del coste de desarrollo del proyecto. Para aquellos elementos como por ejemplo el ordenador o el teclado, que ya estaban adquiridos al inicio de este, se adjudicara un coste orientativo en función de su valor de adquisición.

Para calcular los costes de desarrollo, se ha adjudicado un coste por hora equivalente a la parte baja de la horquilla de un salario medio de un programador y un ilustrador profesionales.

3.4.1 Hardware

Elemento	Descripción	Coste
Msi Dominator 7RD	Ordenador portátil de alta gama	2100€
Tableta gráfica Huion Inspiroy	Tableta gráfica	200€
Teclado electrónico Yamaha	Teclado electrónico de gama media	615€
Total Hardware		2915€

Tabla 1

3.4.2 Software

Elemento	Descripción	Coste
Unity	Motor del Juego	0€
Photoshop CC2019	Programa de edición de imagen e ilustración	120€
Visual Studio 2019	Entorno de programación	0€
Adobe Audition	Programa de edición de audio	200€
Total software		320€

Tabla 2

3.4.3 Assets

Elemento	Descripción	Coste
Música y SFX	Archivos de audio del juego	0€
Sprites	Archivos de imagen del juego	0€
Total Assets		0€

Tabla 3

3.4.4 Desarrollo

Elemento	Descripción	Coste
Programación	Coste aproximado de la programación para unas 350h de desarrollo y un salario medio de 20€ la hora.	7.000€
Desarrollo de assets	Coste aproximado del diseño de assets (tanto de audio como de ilustración) para unas 250h de desarrollo y un salario medio de 35€ la hora.	8.750€
Total desarrollo		15.750€

Tabla 4

3.4.5 Total

Elemento	Descripción	Coste
Hardware		2915€
Software		320€
Assets		0€
Desarrollo		15.750€
TOTAL		18.985€

Tabla 5

Finalmente, tras hacer todos los cálculos obtenemos un coste total de 18.985€. Este se ha visto enormemente incrementado por los elementos audiovisuales del juego, ya que, si se hubiesen usado elementos ya creados y gratuitos, la reducción de costes hubiese sido muy notable. No obstante, no es objeto de esta memoria ni este proyecto discurrir sobre estos aspectos.

4. Diseño técnico

4.1 Entorno

Tal como se ha especificado en el punto 3.3.1 de la presente memoria, el entorno de desarrollo escogido en cuanto a motor ha sido Unity, concretamente la versión 2020.3.23f1.

Como se indica en este apartado, el motivo principal ha sido la abundancia de documentación y guías existentes, así como el uso de un lenguaje de programación con cierto bagaje como es C#.

Como IDE de desarrollo de todo el código del juego, se ha decidido usar Visual Studio 2019, principalmente debido a que, al ser estudiante, la licencia era gratuita, es un IDE de interfaz bastante sencilla y amigable y además por su integración directa con Unity, siendo este último punto algo muy positivo en cuanto a comodidad.

4.2 Requisitos

En este punto encontramos dos tipos de requisitos mínimos pese a que ambos pueden ser encontrados en la página web de Unity, por un lado, los necesarios para correr el editor en sí, y por otro los necesarios para Unity player.

4.2.1 Unity Editor

Unity Editor system requirements

This section lists the minimum requirements to run the Unity Editor. Actual performance and **rendering** quality may vary depending on the complexity of your project.

Minimum requirements	Windows	macOS	Linux (Support in Preview)
Operating system version	Windows 7 (SP1+), Windows 10 and Windows 11, 64-bit versions only.	High Sierra 10.13+	Ubuntu 20.04, Ubuntu 18.04, and CentOS 7
CPU	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support
Graphics API	DX10, DX11, and DX12-capable GPUs	Metal-capable Intel and AMD GPUs	OpenGL 3.2+ or Vulkan-capable, Nvidia and AMD GPUs.
Additional requirements	Hardware vendor officially supported drivers	Apple officially supported drivers	Gnome desktop environment running on top of X11 windowing system, Nvidia official proprietary graphics driver or AMD Mesa graphics driver. Other configuration and user environment as provided stock with the supported distribution (Kernel, Compositor, etc.)
	For all operating systems, the Unity Editor is supported on workstations or laptop form factors, running without emulation, container or compatibility layer.		

Figura 17 Requisitos editor

4.2.2 Unity Player

Unity Player system requirements

This section lists the minimum requirements to build and run the Unity Player. Actual performance and rendering quality may vary depending on the complexity of your project.

Desktop

Operating system	Windows	Universal Windows Platform	macOS	Linux
Operating system version	Windows 7 (SP1+), Windows 10 and Windows 11	Windows 10, Xbox One, HoloLens	High Sierra 10.13+	Ubuntu 20.04, Ubuntu 18.04, and CentOS 7
CPU	x86, x64 architecture with SSE2 instruction set support.	x86, x64 architecture with SSE2 instruction set support, ARM, ARM64.	x64 architecture with SSE2.	x64 architecture with SSE2 instruction set support.
Graphics API	DX10, DX11, DX12 capable.	DX10, DX11, DX12 capable GPUs.	Metal capable Intel and AMD GPUs	OpenGL 3.2+, Vulkan capable.
Additional requirements	Hardware vendor officially supported drivers. For development: IL2CPP scripting backend requires Visual Studio 2015 with C++ Tools component or later and Windows 10 SDK.	Hardware vendor officially supported drivers. For development: Windows 10 (64-bit), Visual Studio 2015 with C++ Tools component or later and Windows 10 SDK.	Apple officially supported drivers. For development: IL2CPP scripting backend requires Xcode. Targeting Apple Silicon with IL2CPP scripting backend requires macOS Catalina 10.15.4 and Xcode 12.2 or newer.	Gnome desktop environment running on top of X11 windowing system Other configuration and user environment as provided stock with the supported distribution (such as Kernel or Compositor) Nvidia and AMD GPUs using Nvidia official proprietary graphics driver or AMD Mesa graphics driver.
	For all operating systems, the Unity Player is supported on workstations, laptop or tablet form factors, running without emulation, container or compatibility layer.			

Figura 18 Requisitos player

4.3 Inventario de herramientas usadas

Se presenta a continuación una relación de las diversas herramientas de software usadas a lo largo de la vida del proyecto.

UNITY 2020.3.23f1

Motor usado para el desarrollo y la creación del videojuego.

ADOBE PHOTOSHOP CC2019

Herramienta usada para toda la edición visual del juego y la creación de las ilustraciones.

ADOBE AUDITION

Programa de edición de audio usado para la creación y edición de los archivos de audio del juego.

VISUAL STUDIO 2019

IDE usado como entorno de programación para el código de scripts

4.4 Inventario de assets y recursos del juego

4.4.1 Gráficos

Tileset:

Mapa principal del nivel, el cuerpo principal, está compuesto por una sola imagen a la que posteriormente se añadió un elemento collider. Adicionalmente, existen dos modelos de islas pequeñas y una más grande situada al final del nivel.

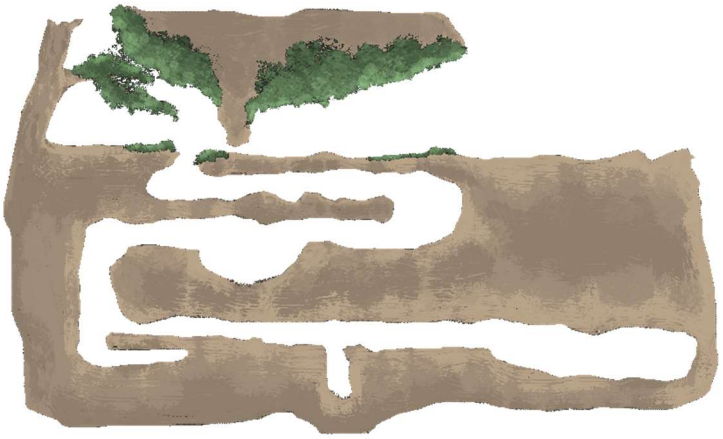


Figura 14 Mapa principal

Figura 15 Isla final

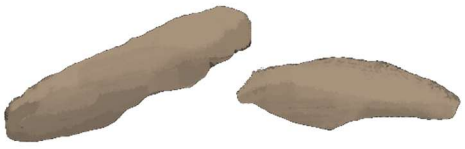


Figura 16 Isla 1

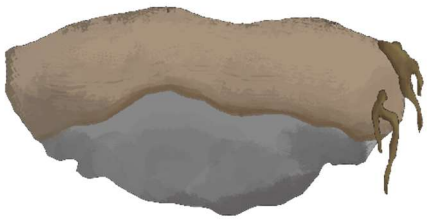


Figura 17 Isla 2

Background:

En este apartado, encontramos todos aquellos sprites dedicados tanto a la creación del background de la parte superior del nivel como la inferior.

Para su creación, se ha jugado bastante con las transparencias, especialmente con las montañas. Debido al color blanco de las nubes, se les ha tenido que aplicar un relleno de fondo sólido por tal de visualizarlas en este documento, dicho relleno no forma parte del modelo ni del juego.

Figura 19 Hierba

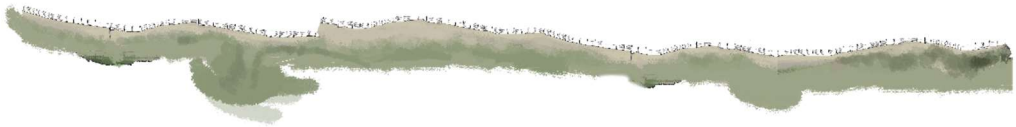


Figura 20 Muro

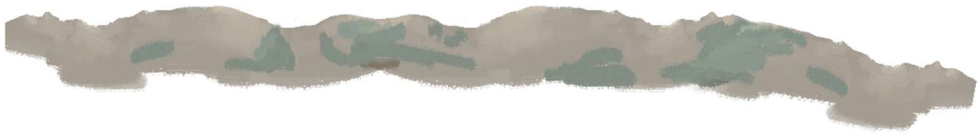


Figura 21 Montañas

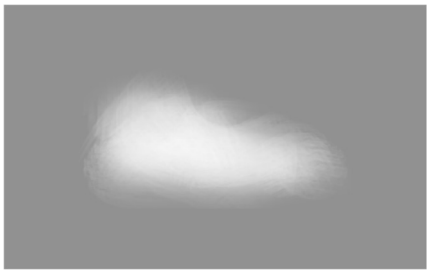


Figura 23 Nube 1

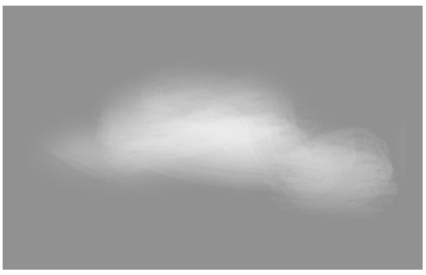


Figura 22 Nube 2

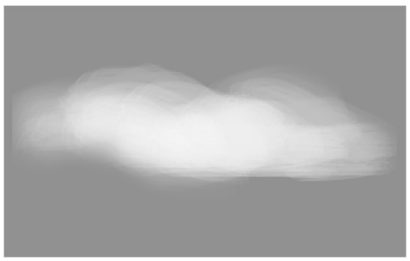


Figura 25 Nube 3

Figura 24 Cielo

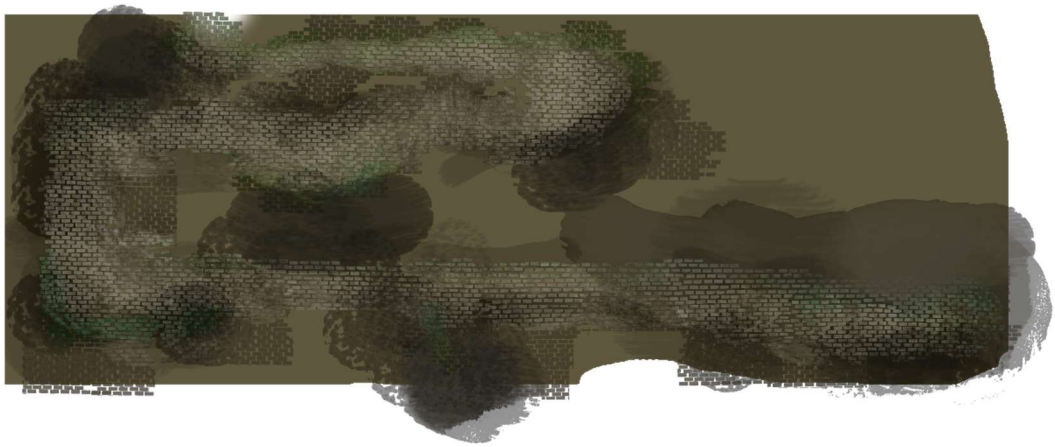


Figura 26 Fondo zona sótano

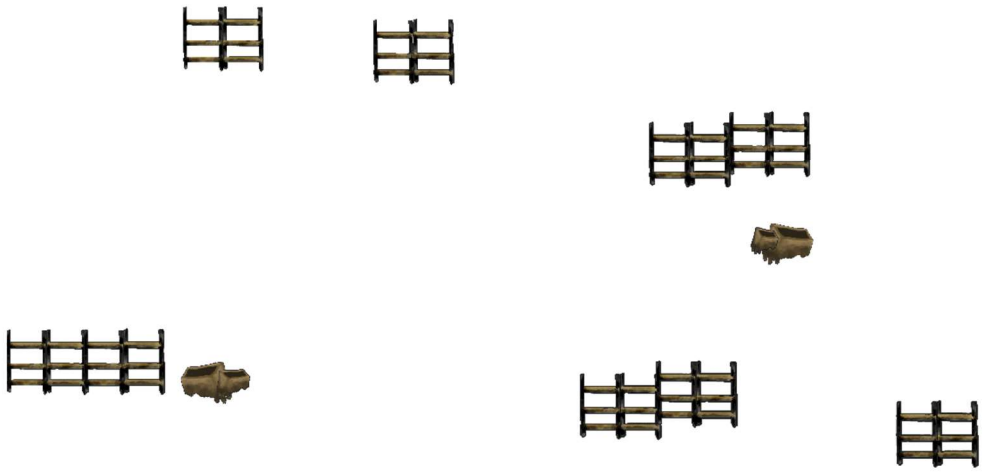


Figura 27 estanterías y cajas.

Aliados

En este apartado quedan englobados todos los personajes que ayudan a Sultán a lo largo de su aventura, es decir, encontramos los spritesheets de Moriarty y el de los checkpoints.

Figura 28 Moriarty Idle

Figura 29 Moriarty Meow

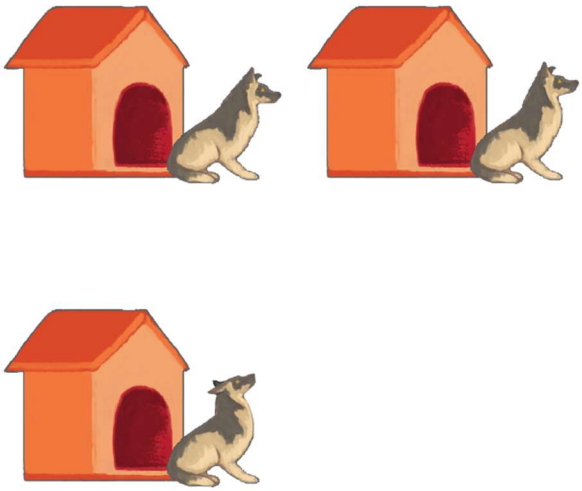


Figura 30 Animación checkpoint

Decoraciones

Relación de todos los elementos decorativos creados con tal de dar poblar el escenario y darle vida. El conjunto está compuesto en su mayoría por elementos de follaje, como plantas y arbustos, así como algunas rocas.

Figura 31 Conjunto de decoraciones

Enemigos

Compendio de todos los enemigos del juego y sus diferentes animaciones.

Figura 32 SpriteSheet perro caminando

Figura 33 SpriteSheet perro detenido

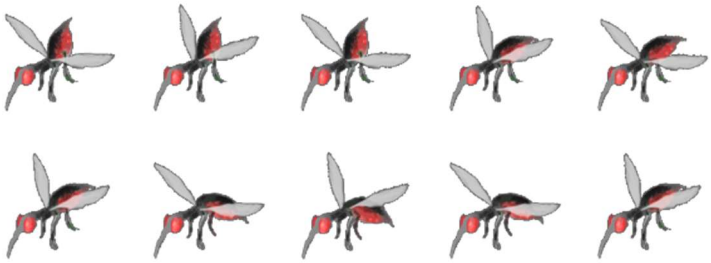


Figura 34 SpriteSheet mosquito

Figura 35 SpriteSheet leishmania

Medicación y powerUps

Listado de todos los objetos coleccionables por parte del jugador, tanto llaves como powerUps.

Figura 36 conjunto de llaves del juego

Figura 37 conjunto de power ups del juego

Plataformas

En este apartado se adjuntan los diseños de los diferentes tipos de plataforma, tanto las traspasables de color naranja, como las móviles, de color gris.

Figura 38 plataformas estáticas

Figura 39 plataformas móviles

Jugador

Compendio de los spritesheets correspondientes a cada una de las animaciones del personaje principal, Sultán.

Figura 40 Spritesheet correr

Figura 41 Spritesheet aterrizaje

Figura 42 Spritesheet idle

Figura 43 Spritesheet caida

Figura 44 Spritesheet Iddle 2

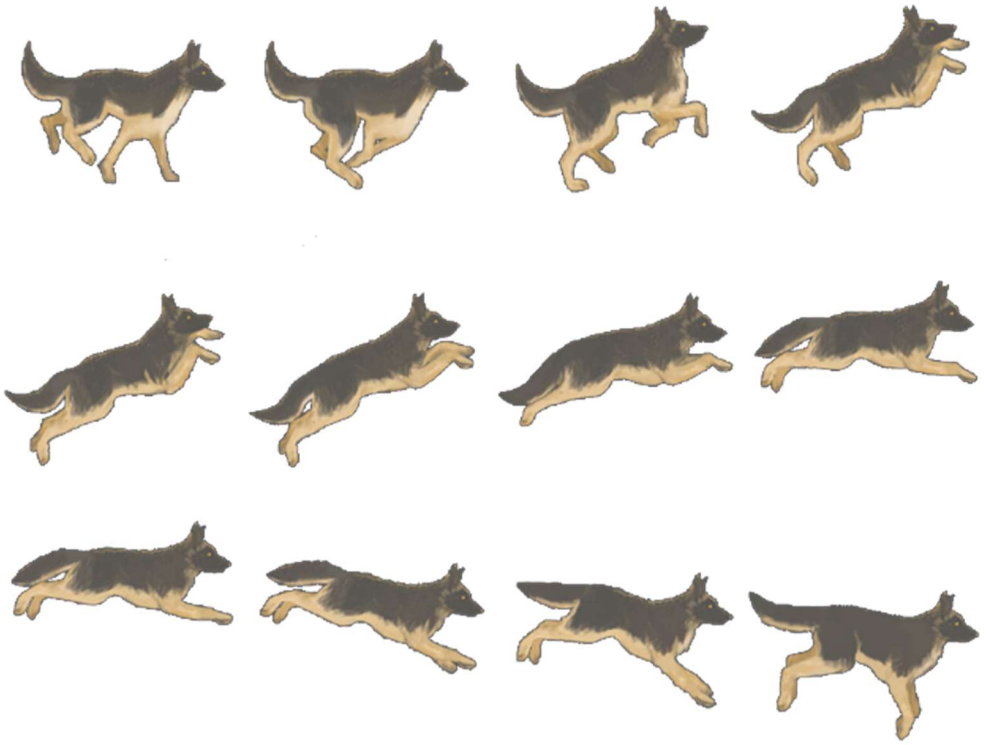


Figura 45 Spritesheet salto

Menús

Relación de los diferentes elementos gráficos usados para la creación de los menús y submenús del juego, tanto el principal como el de pausa y el de opciones. Para poder visualizar correctamente en este documento alguna de las imagines, se ha tenido que seguir el mismo procedimiento que anteriormente con las nubes.

Figura 46 Cursor y títulos de submenús

Figura 47 Título menú principal

Figura 48 Background menú principal

4.4.2 Audio

El audio, ha sido el único apartado de todo el proyecto en el que no se han generado todos los assets de forma propia. De echo, dadas las dificultades encontradas a la hora de gestionar la edición y grabación de este primer archivo, se decidió que el resto de assets de audio serían de terceros.

Se trata de una cover en piano de la canción “For the Good times” de Kris Kristofferson.

Concretamente, la cover, está hecha sobre la versión de la misma canción interpretada por Johny Cash, debido a su tempo ligeramente más pausado.

Este tema, es el usado en el menú principal del juego.

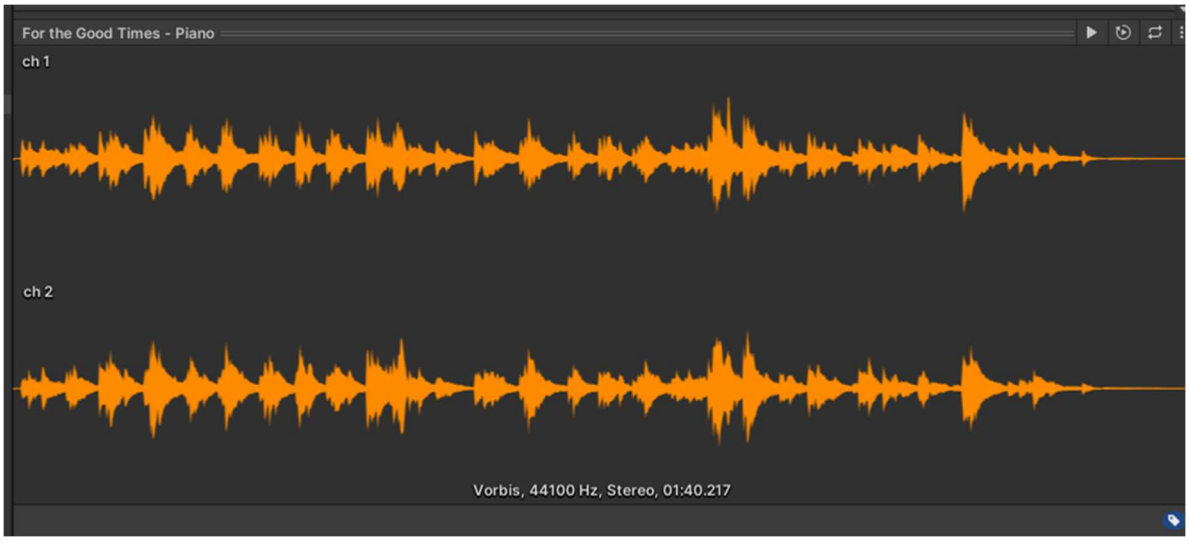


Figura 49 Tema menú principal

4.4.3 Scripts

Los scripts, forman la espina dorsal de todo juego, unity permite asignarlos como un componente más a nuestros objetos para definir por código como deseamos que se comporte este.

Se ha tratado de mantener siempre un estilo de programación lo más simple, organizado y modular posible por tal de facilitar la legibilidad y comprensión del código, tanto durante el proyecto como en sus futuras ampliaciones.

Una de las máximas que se ha tratado de seguir en todo momento, es que la mayoría de las variables, fuesen públicas o al menos, que siempre estuviesen serializadas. Este sencillo paso, permite que puedan ser modificadas desde el mismo editor de unity, facilitando enormemente la tarea de testeo, desarrollo y ajuste de mecánicas.

Por tal de preservar la legibilidad de la memoria y mantenerla organizada, se ha decidido agruparlos de forma muy similar a la que se ha usado en el proyecto, resultando la relación final la que sigue:

Jugador: aquí encontraremos los scripts que afecten directamente al personaje principal

- **PlayerController**

  Este es uno de los scripts más complejos de todo el proyecto, es el encargado de gestionar todo lo referente al personaje, haciendo uso de diversas funciones a las que llama frame a frame en el método Update.

Sus principales funciones son, gestionar los inputs del teclado y traducirlos a movimiento del jugador, gestionar mediante variables booleanas las transiciones entre las diferentes animaciones, rotar al jugador para que este siempre mirando hacia la dirección que se mueve, y finalmente, el checkeo constante del ángulo en el que se encuentra el suelo respecto al jugador.

Mediante esta última funcionalidad y el uso de diversos RayCast, sabemos en todo momento si el jugador está en terreno plano, de subida o de bajada y en caso de estar en una pendiente, si esta excede el ángulo máximo como para andar por ella.

No obstante, esta comprobación genera algunos bugs de colisiones bastante molestos que se detallaran más adelante

- **PlayerRespawn**

  Pequeño script que almacena la posición del último checkpoint con el que se ha colisionado. En caso de morir, reapareceremos en las coordenadas almacenadas.

- **Checkground**

  Este script es muy simple, pero indispensable para el correcto funcionamiento del juego. Mediante el uso de etiquetas y colisiones, almacena si el jugador se encuentra en el suelo o en aire.

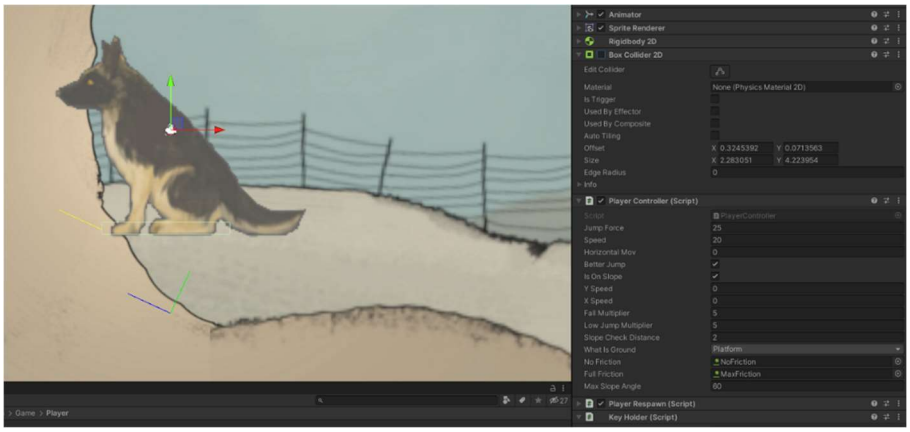


Figura 50 Raycasts, checkground y variables de PlayerController

Enemigos: se relacionan aquellos scripts que afecten a los enemigos, especialmente en lo que respecta a la IA

- **BasicAI**

  Es el encargado de gestionar las transiciones de animaciones, almacenar los distintos puntos de patrulla establecidos para los enemigos, así como su velocidad de desplazamiento y el tiempo que permanecen detenidos en cada punto.

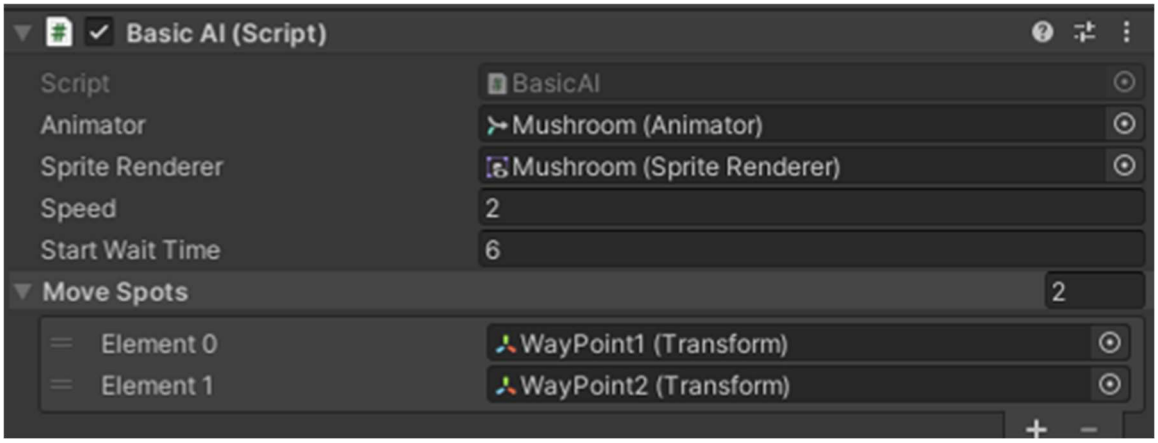


Figura 51 Puntos de ruta IA

- **DamagerObject**

  Esta parte del código interactúa directamente con el script PlayerRespawn, en caso de detectar una colisión con el jugador, reinicia la escena actual con el jugador en las coordenadas del último checkpoint.

Otros elementos: compendio de scripts que afectan al resto de objetos y elementos con los que el jugador puede interactuar (power ups, llaves, checkpoints, etc.)

- **Checkpoint**

  Actúa como trigger de la animación y el sonido al colisionar con un punto de control.

- **PowerUp**

  Este script, es el encargado de modificar los stats correspondientes del jugador en función al tipo de power up y multiplicador que corresponda.

- **CollectedFood**

  Al colisionar con un power up, dispara la animación de recogida de objetos y se comunica con el AudioManager con tal de reproducir el sonido correspondiente.

- **CollectedKey**

  Actúa de manera análoga al anterior, pero al colisionar con una llave.

- **Key**

  Permite asignar a cada llave un tipo de llave de los de una lista mediante el editor.

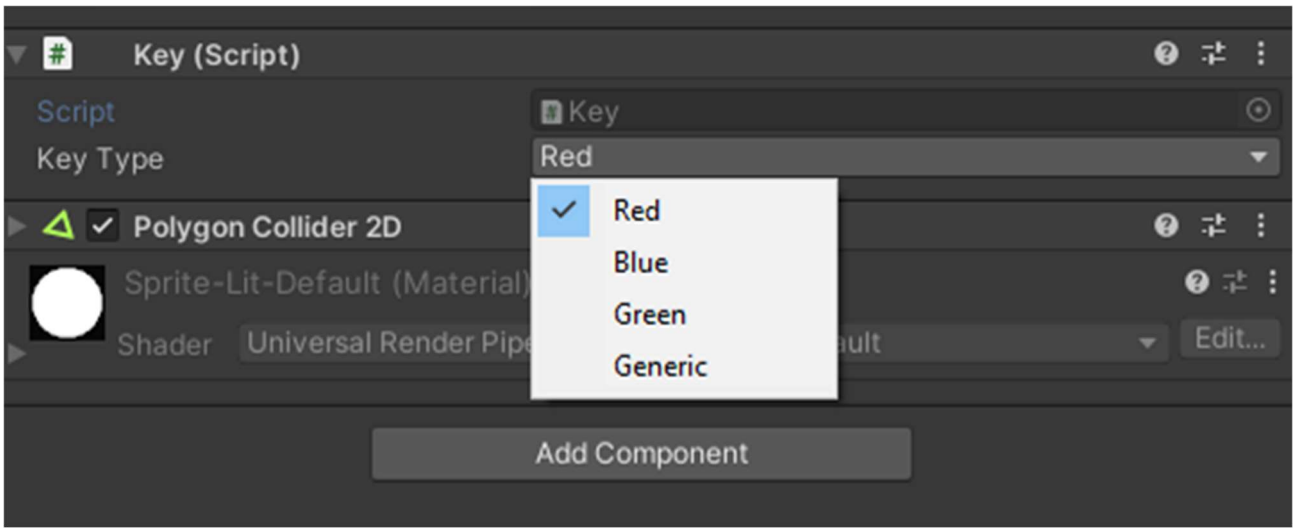


Figura 52 script Key

- **KeyDoor**

  Se encarga de diversos aspectos, en primer lugar, permite asignar el tipo de llave que abre la puerta comunicándose con el script anterior, define la animación por la que se disuelve el obstáculo y finalmente se comunica con AudioManager para reproducir el sonido asignado.

- **KeyHolder**

  Script asignado al personaje, se encarga de la comunicación con el script Key para gestionar el inventario de llaves que se posee, adicionalmente, al colisionar con una puerta, se comunica con keyDoor para disparar sus eventos.

- **Revive**

  Este controlador, esta asignado a los grupos de plantas marchitas alrededor de cada barrera de leishmania, una vez la barrera que tienen asignada se disipa, las plantas recuperan su color verde original para representar el resurgir de la vida.

- **MovingPlatform**

  Permite a las distintas plataformas móviles recorrer una serie de puntos de manera análoga a la IA de los enemigos.

- **Platform**

  Permite que las plataformas puedan ser traspasadas haciendo uso de un componente de tipo Effector.

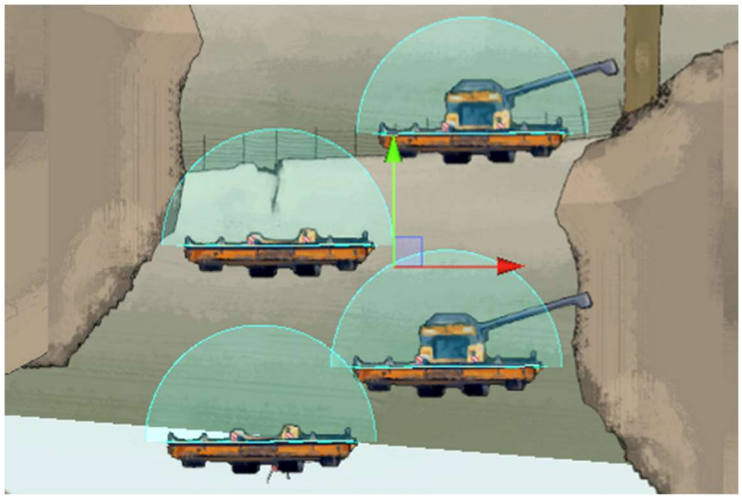


Figura 53 Effector

- **Moriarty**

  De forma análoga al resto de scripts, gestiona las transiciones entre las animaciones y los sonidos en función de las colisiones. Adicionalmente, es el encargado de activar y desactivar los diferentes textos emergentes mediante los que Moriarty se comunica con el jugador.

Iluminación: listado de los scripts usados para los efectos de iluminación.

- **FlickeringLight**

  Permite a los elementos a los que este asignado, parpadear por instantes de tiempo aleatorios.

- **GlowFlicker**

  Dota a las luces a las que este asignado de un efecto de incandescencia en el que su radio de iluminación aumenta y disminuye.

- **LightZone**

  Permite gestionar mediante colisiones y el cambio de intensidad de la iluminación global, disminuyendo esta hasta el mínimo al entrar en la parte inferior del mapa.

Audio: conjunto de scripts usados para definir el comportamiento de los archivos de sonido, ya sea accediendo a sus distintos componentes como volumen y pitch o simplemente estableciendo cuando debe dispararse.

- **AudioManager**

  Gestiona la mayoría de los efectos de sonido del juego. Permite almacenar una lista de sonidos para que el resto de los scripts, vayan comunicándose con este indicándole cual debe reproducir.

- **AudioZone**

  Actúa de forma análoga al script LightZone pero con respecto al cambio de música al entrar en la parte inferior.

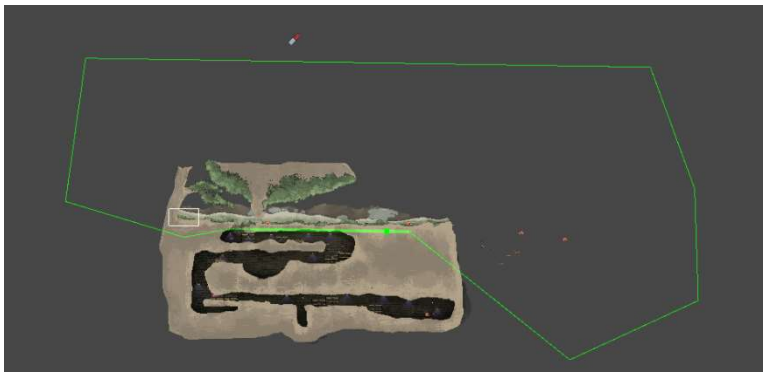


Figura 54 Collider que divide las zonas

- **Sound**

  Esta clase, permite definir cuáles son las variables de la fuente de sonido que queremos que sean visibles y modificables por AudioManager.

Interfaz: scripts que afectan directamente a la UI, mostrando que objetos tenemos en cada momento

- **TimerController**

  Activa y desactiva los temporizadores radiales asociados a cada power up, mostrando mediante estos su duración restante en la interfaz.

- **UI_KeyHolder**

  Se comunica con los scripts Key y KeyHolder, por tal de mantener la interfaz sincronizada con el inventario de llaves del jugador, mostrando en todo momento las que posee.

Menús: conjunto de scripts que definen el comportamiento de los distintos menús y permiten al usuario navegar por ellos

- **MainMenu**

  Permite al menú cargar las distintas escenas y cerrar la aplicación. Al cerrar la aplicación, borra todos los datos relativos a posición del jugador almacenados por los checkpoints. Este último detalle, permite solucionar el bug encontrado por el consultor en anteriores entregas en que Sultán hacia spawn en zonas que no debía imposibilitando jugar.

- **PauseMenu**

  Permite acceder al menú de pausa en mitad del juego y detiene el tiempo al hacerlo.

- **SettingsMenu**

  Permite comunicarse al slider de volumen con el mixer principal, posibilitando que el volumen se ajuste en función de la posición física de dicho slider.

4.4.4 Assets externos

Tal como se ha indicado en el apartado 4.4.2, la mayoría de los archivos de sonido del juego son assets de terceros, estos se han obtenido de las siguientes fuentes:

Free Orchestral Music Pack – Unity Asset Store
https://assetstore.unity.com/packages/audio/music/orchestral/free-orchestral-music-pack- 189885

- **Aspiration Woods:** Tema usado como principal del mapa. Cuando accedemos a la parte inferior se modifica por script el pitch del mismo consiguiendo un sonido más ronco y lúgubre.

GameaudioGDC 2017 - Sonnis.com

https://sonniss.com/gameaudiogdc

- **Button click:** sonido usado cada vez que se pulsa un botón.
- **Cat Meow:** maullido de gato que suena cuando colisionamos con Moriarty.
- **Door Open:** sonido “viscoso” que es usado cada vez que una barrera de leishmania se disuelve

- **Eat:** sonido de masticado, usado cada vez que recogemos un powerUp

- **Howl:** aullido de perro, este sonido se usa cuando colisionamos con un checkpoint y se dispara su animación.

- **Mosquito:** zumbido de mosquito, sonido que emiten todos los mosquitos al volar. Tiene asignada un área de efecto, y si el jugador está dentro de ella será capaz de escucharlo. A medida que nos alejamos la intensidad disminuye.

- **Small dog bark:** ladrido de perro de tamaño pequeño. Es emitido por todos los enemigos de tipo perro del mapa cuando estos entran en la animación Idle. Tiene asignada un área de efecto, y si el jugador está dentro de ella será capaz de escucharlo. A medida que nos alejamos la intensidad disminuye.

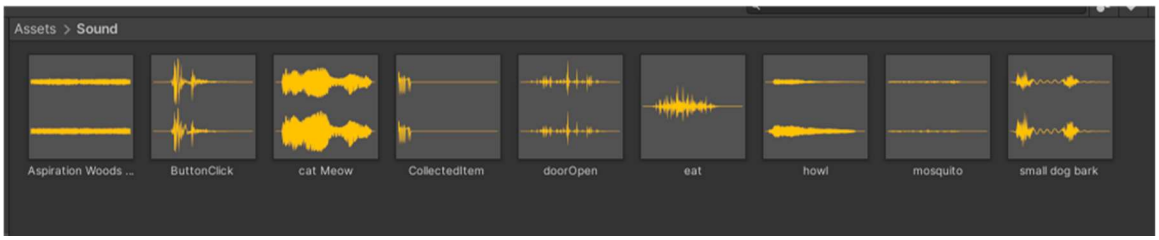


Figura 55 Assets audio externos

4.4.5 Bugs conocidos

- **Rampas:**

  Debido a que se ha usado un tileset que contiene rampas, se ha tenido que gestionar cómo se comportan las físicas en estos lugares. No obstante, la solución implementada tiene ciertas limitaciones, y si el personaje se queda detenido justo en la “cima” de una de estas rampas, es decir, justo en el punto exacto de inflexión respecto a las pendientes, se queda bloqueado y no puede moverse. La única solución en estos casos es saltar por tal de “desatascarse”.

- **Transición de animación de caída**

  Debido a problemas de sincronía entre el método Update y el encargado de actualizar las animaciones, muchas veces pese a estar cayendo, el personaje mantiene la animación correspondiente a Idle en lugar de la de caída.

  La principal causa, es que la caída es tan corta, que antes de que se haga la comprobación del frame, el personaje ya ha aterrizado. Por el momento no se ha encontrado ninguna solución.

4.5 Arquitectura del juego

El presente apartado, detallará en profundidad los aspectos más relevantes de la arquitectura del juego.

4.5.1 Diagrama de flujo menús

Esta figura detalla el flujo seguido por el jugador al correr la aplicación.

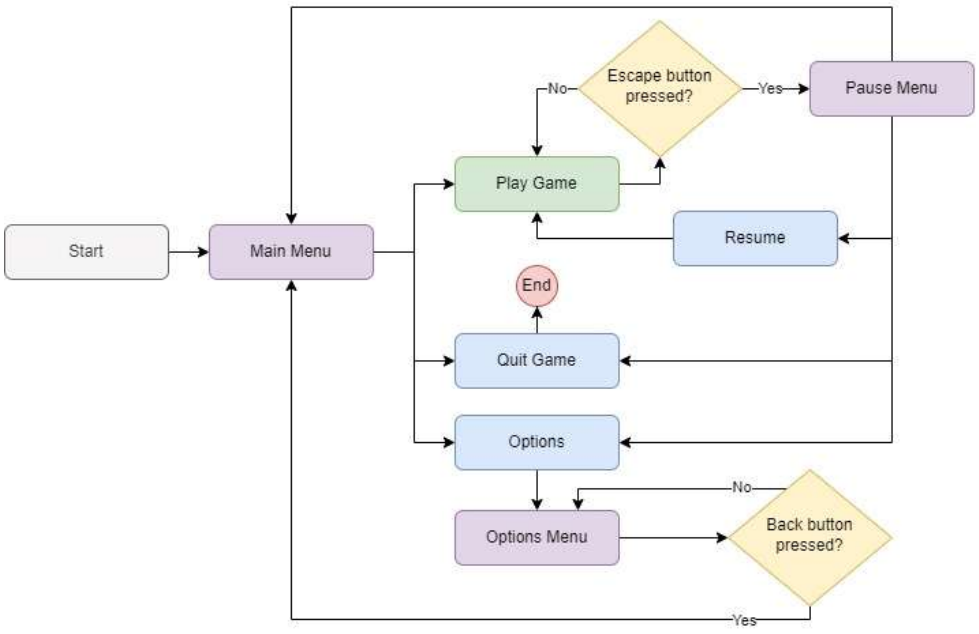


Figura 56 Diagrama de flujo menús

4.5.2 Diagrama de flujo escena jugable

Este diagrama, muestra el flujo seguido por la escena principal de la demo, este termina al llegar al punto final del mapa obtener el mensaje de Moriarty informándonos de que hemos finalizado la demo.

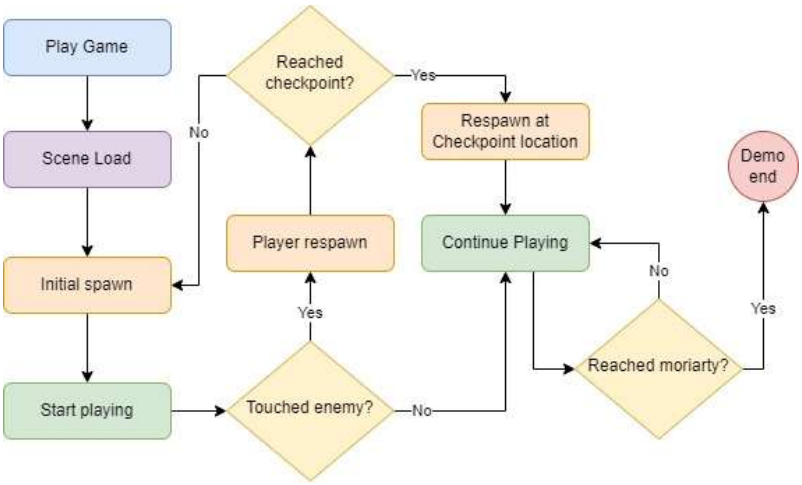


Figura 57 Diagrama flujo escena

4.5.3 Diagramas UML de subclases.

Este diagrama, nos muestra las diferentes subclases de cada uno de los prefabs más importantes del juego.

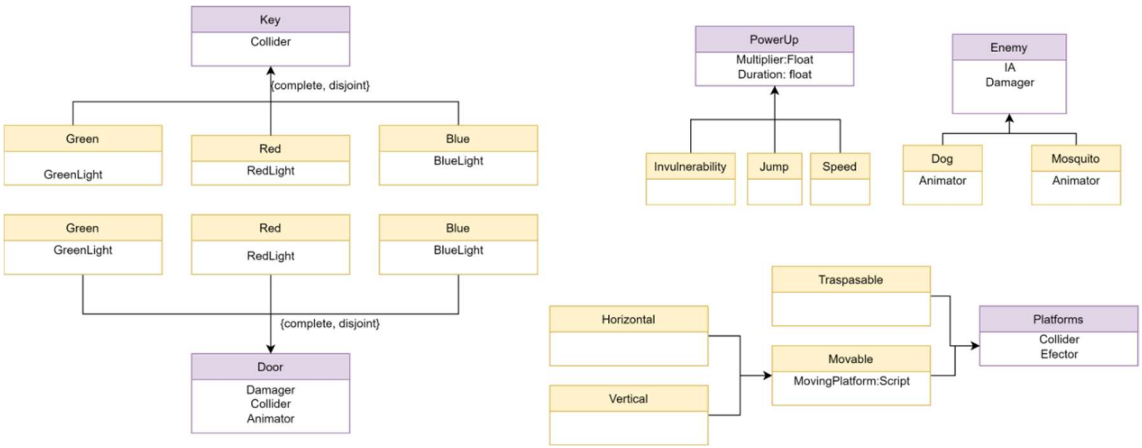


Figura 58 UML subclases

4.5.4 UML General

Este diagrama, muestra cómo se relacionan las clases más importantes del juego entre sí, sus multiplicidades y de que elementos depende cada componente. Destacar que por mantener el diagrama legible y de un tamaño apto, se ha obviado el indicar que todos los objetos que tengan un componente spriteRenderer y no sean de la UI, se ven afectados por IlluminationController. Tampoco se ha incluido el tileset ni sus colliders, ya que estos solo interactúan con el propio player, los perros enemigos (al tener componente RigidBody2D) y evidentemente, con la iluminación.

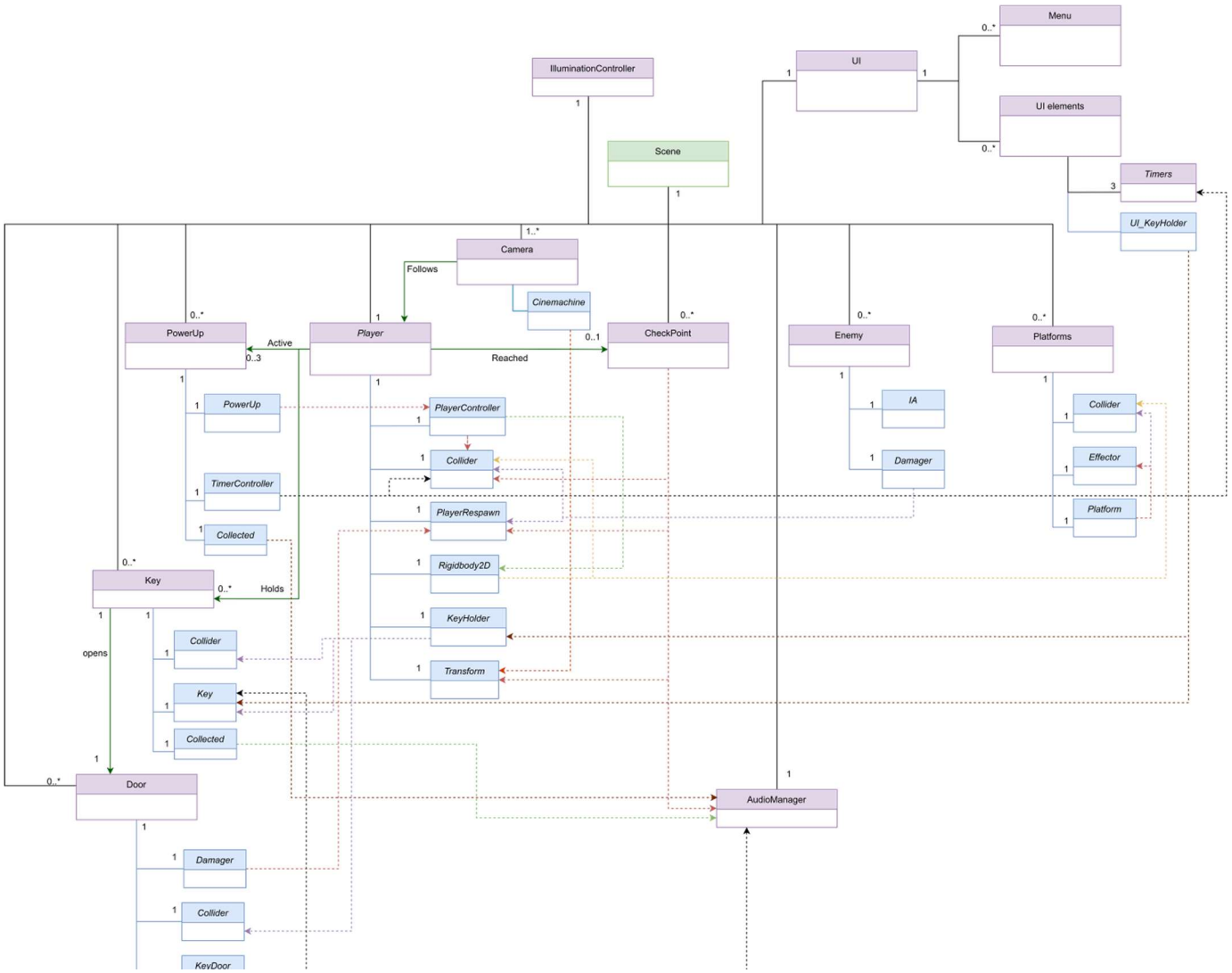


Figura 59 UML general

4.6 Animaciones

A continuación, se relacionan y detallan los árboles de animación más relevantes del proyecto. Aquellos elementos que poseían animaciones de un solo estado, es decir aquellos que, pese a estar animados, no eran capaces de transicionar a otras animaciones, han sido excluidos de esta relación por motivos evidentes.

4.6.1 Moriarty

Moriarty cuenta con un árbol de animación bastante simple y que actúa de forma muy similar al de los checkpoints. Permanece en estado Idle hasta que el jugador colisiona con él. En ese momento se lanza una animación en que este maúlla (acompañada del correspondiente sonido), cuando esta animación finaliza, regresa al estado Idle.

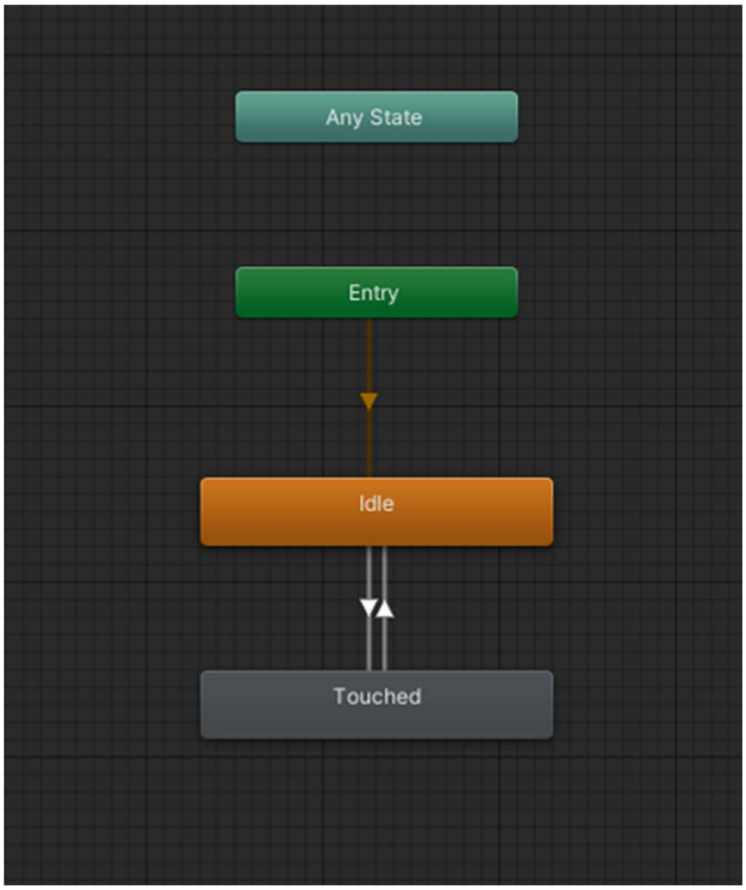


Figura 60 Árbol Moriarty

4.6.2 Mosquito y perros enemigos

Tanto los mosquitos como los perros enemigos tienen un árbol de animación idéntico, mientras se desplazan permanecen en una animación y en el momento que se detienen pasan a la animación Idle. Una vez retoman la marcha hasta el siguiente punto, se vuelve a transicionar a la animación correspondiente.

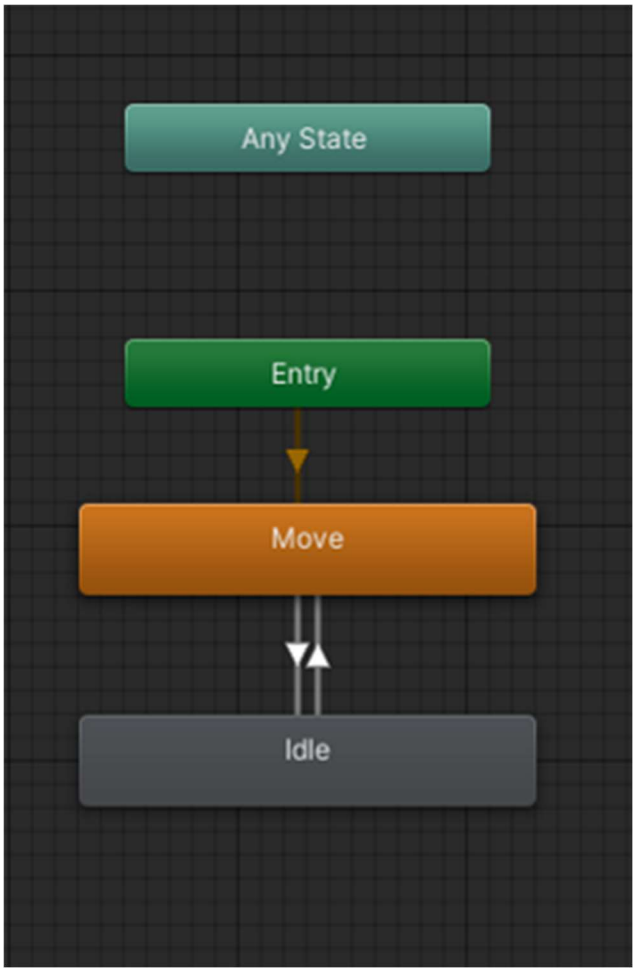


Figura 61 Árbol mosquito

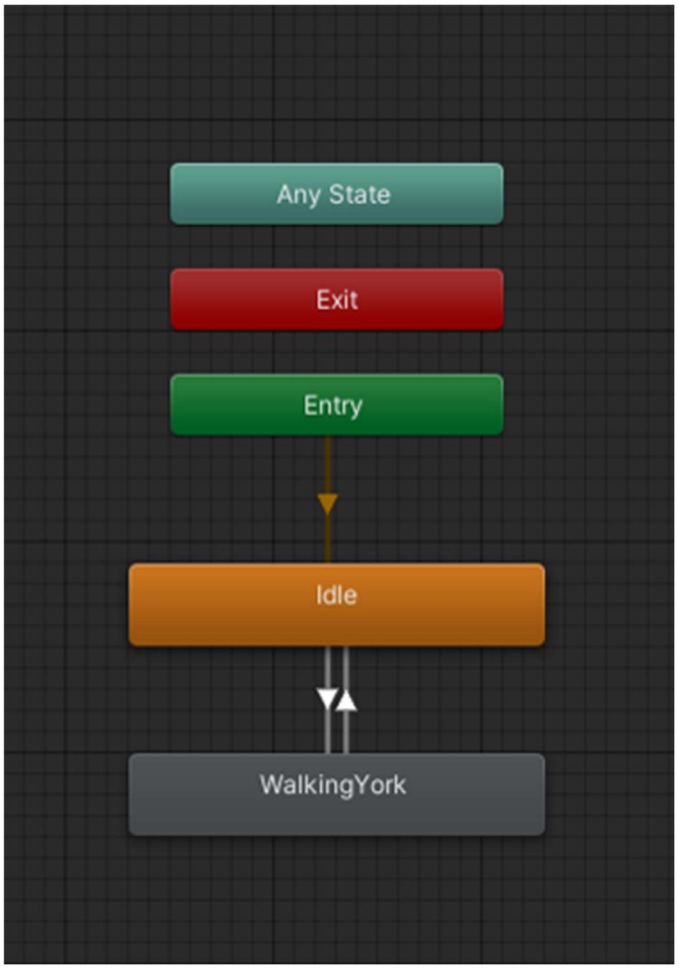


Figura 62 Árbol perro

4.6.3 Sultán

Tal como se puede apreciar en la figura, Sultán posee el árbol de animaciones más complejo de todo el juego.

Mientras no reciba ningún input y esté en el suelo, permanecerá en Idle, si permanece más de cierto intervalo de tiempo en esta, transaccionará automáticamente a la animación Sit, idéntica a la anterior, pero en la que mueve la cola (de esta manera se consigue que el juego no parezca estar “congelado” pese a estar parado).

En el momento que el jugador pulsa la tecla correspondiente a saltar y siempre que no esté ya en el aire, el personaje transiciona a la animación Jump, y una vez empieza a descender (es decir, comprobamos si su velocidad en el eje y es negativa) lo hace a Fall. En el momento en que toca el suelo y por tal de suavizar la transición de Fall a Move, se ha implementado la animación auxiliar Landing, que contiene una pequeña transición entre ambas animaciones.

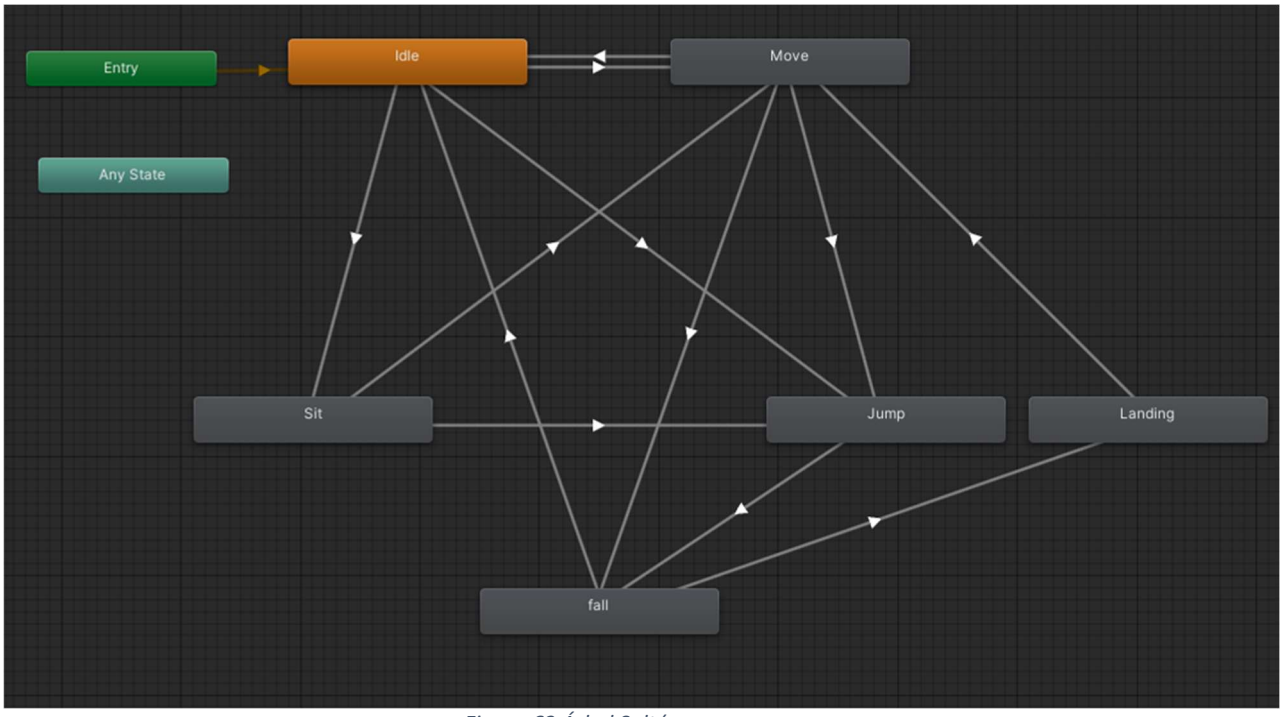


Figura 63 Árbol Sultán

4.7 URP

Uno de los grandes puntos a fuertes de Unity como motor, es sin duda la integración directa con herramientas muy poderosas de forma gratuita. En este caso, hablamos de URP (Universal Render Pipeline), se trata de una potente pipeline de renderizado y postprocesado usada para generar efectos de iluminación.

La mayoría de sus funcionalidades referentes a la iluminación 2D, siguen en fase experimental, no obstante, se han obtenido unos resultados mejores de lo esperado sin ver sacrificados el rendimiento o la estabilidad.

El uso de iluminación dinámica ha sido uno de los mayores cambios respecto a la planificación inicial del proyecto. Su inclusión, ha permitido funcionalidades que de otra manera hubiesen sido casi imposibles de lograr (o por lo menos bastante más complicadas), como generar los efectos de las luces parpadeantes, la luminiscencia de las pastillas y barreras de leishmania así como la creación de dos zonas muy diferenciadas dentro de un mismo mapa. Todo esto, ha llevado el aspecto visual del juego un nivel por encima, haciendo que este luzca mucho mejor con un coste de horas relativamente asequible.

Otro aspecto muy destacable, es la posibilidad de crear renderers customizados y shaders, algo que sin duda se explotará de cara a versiones futuras del juego.

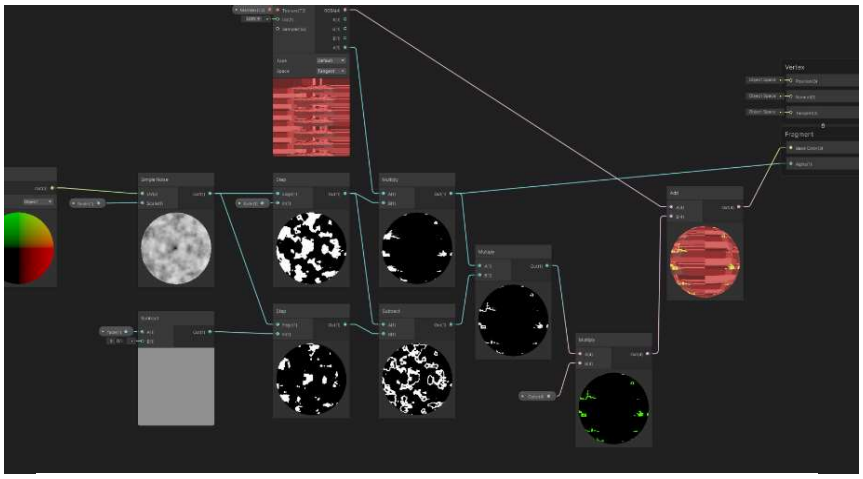


Figura 64 Shader custom

5. Diseño de niveles

Pese a que no se ha diseñado un nivel demasiado grande, se ha tratado de obligar al jugador a tenerlo que recorrer de forma no lineal para alargar la duración. Es decir, antes de poder acceder a la zona inferior, deberá avanzar un poco por la superior y posteriormente, cuando llegue al final de la zona inferior deberá volver a la superior antes de dirigirse a la zona final.

En la siguiente figura, se adjunta la ruta necesaria a seguir por el jugador junto a una pequeña leyenda. Más adelante en esta memoria se profundizará en el resto de las zonas del mapa mediante imágenes más detalladas.

Cabe destacar, que, aunque no se han indicado en esta guía por ser elementos que es preferible que descubra el jugador por su cuenta, existen ciertas zonas ocultas donde se pueden obtener powerups muy beneficiosos o incluso evitar completamente la zona inferior del mapa.

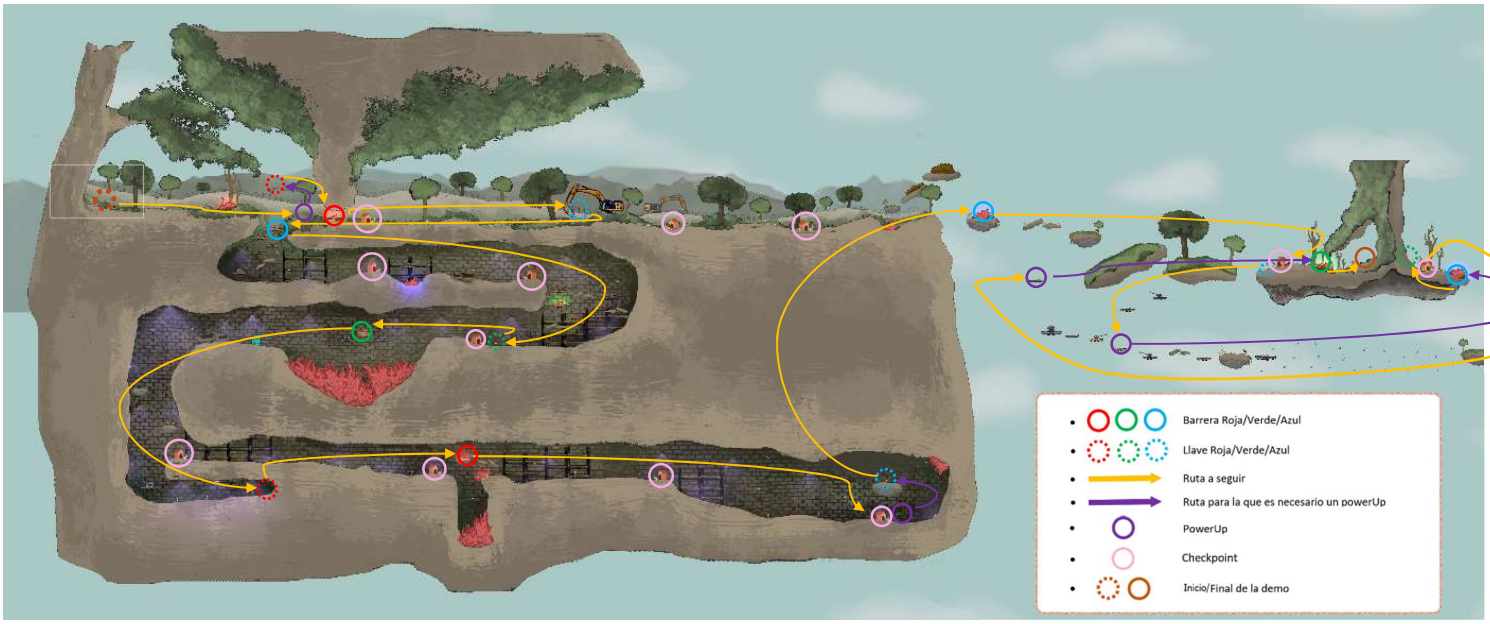


Figura 65 Guía completa del nivel

5.1 Tutorial

Nada más empezar el nivel, nos encontraremos en la zona del tutorial. En esta, Moriarty nos indicará los controles y mecánicas básicos del juego. Una vez disipemos la primera barrera de leishmania empezará la aventura.

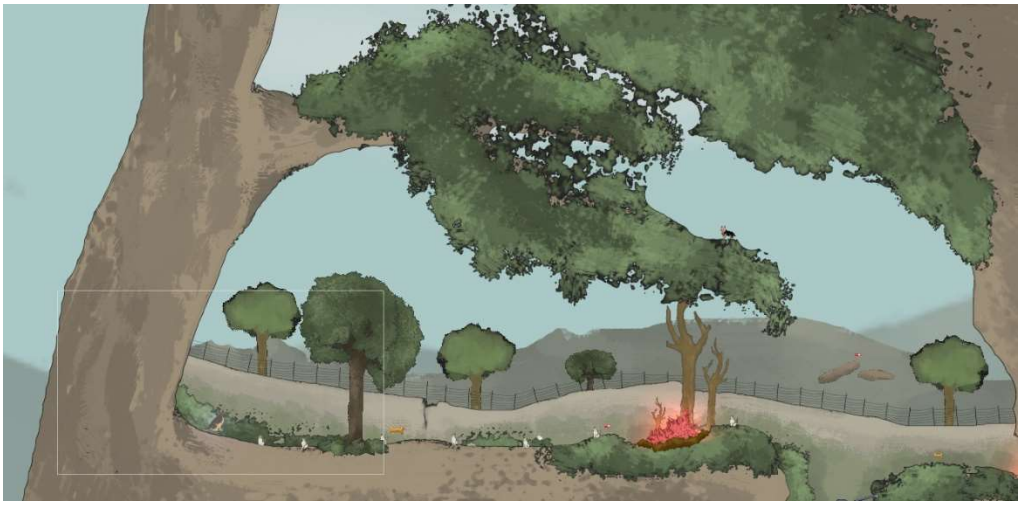


Figura 66 Zona de tutorial

5.2 Parte superior del mapa

En esta zona, apenas encontramos barreras de leishmania, por el contrario, nuestro mayor enemigo serán los mosquitos. En la primera visita deberemos avanzar hasta la excavadora y recoger la pastilla azul por tal de poder acceder a la parte inferior del mapa. Posteriormente deberemos volver a pasar por esta zona para llegar a la parte final.

Figura 67 Zona superior del mapa

5.3 Parte inferior del mapa

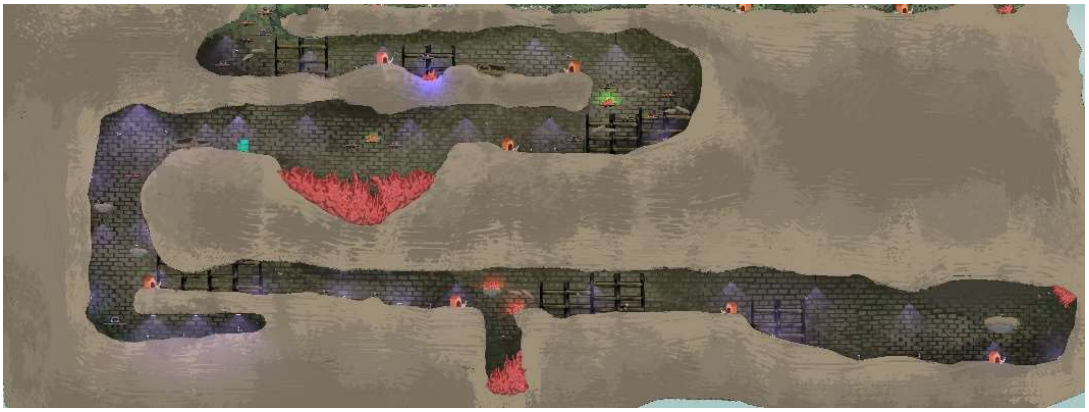


Figura 68 Zona inferior

La parte inferior o sótano, es con diferencia la más desafiante de toda la demo, nada más entrar en ella, nos veremos envueltos de escuridad y deberemos ser muy cautelosos al avanzar, de lo contrario, seremos presa de la enorme cantidad de trampas y enemigos que acechan en las sombras.

Para paliar la dificultad de esta parte, se han añadido una gran cantidad de chekpoints, no obstante, requerirá que el jugador esté concentrado y atento, muchas veces el sonido será la única manera de detectar la distancia a la que nos encontramos de un enemigo.

5.4 Parte final

Esta última parte, tiene un aspecto mucho más similar a un juego de plataformas clásico, donde encontramos gran cantidad de estas flotando en el aire. La dificultad principal en esta zona, será que la cámara no nos permitirá ver casi lo que tenemos alrededor, de forma que el jugador se verá obligado a arriesgarse y saltar al vacio en algunas ocasiones. Encontramos también una zona infestada de mosquitos, que a menos que consigamos una pipeta para ser invulnerables, será imposible atravesar.

Mediante la obligación de uso de este powerup, se añade de forma puntual una mecánica extra, tener que atravesar la zona y volver hasta la barrera final antes de que el tiempo del powerup se agote, forzando así al jugador a ser rápido y preciso en sus movimientos.

Una vez lleguemos al final, encontraremos a Moriarty escondido y nos comunicará el fin de la demo.

Figura 69 Zona final

6. Manual de usuario

6.1 Requisitos mínimos

Dada la sencillez del juego, se considera que los requisitos mínimos para jugarlo son los mismos establecidos por Unity. Pese a que estos ya han sido definidos en el punto 4.2.2, se adjunta breve resumen de los mismos.

- **Sistema operativo:** Windows 7 SP1+ /macOS 10.13+ / Ubuntu 18.04+
- **Procesador:** Compatible con el conjunto de instrucciones SSE2.
- **Memoria:** 1 GB de RAM
- **Gráficos:** DirectX10 (modelo de shader 4.0)
- **Almacenamiento:** 1000 MB de espacio disponible

6.2 Controles

En este apartado se hará una breve explicación de los controles del juego y sus mecánicas principales, no obstante, todo esta información se da al empezar a jugar en una pequeña zona de tutorial.

- **Movimiento:** teclas WASD y flechas de dirección.
- **Salto:** Barra espaciadora.
- **Enemigos:** existen dos tipos de enemigos, los perros que patrullan por el suelo y los mosquitos que lo hacen por el aire. Si el jugador colisiona con ellos volverá al punto de respawn.

- **Barreras de Leishmania:** Al igual que los enemigos, el jugador morirá si colisiona con ellas. Existen cuatro tipos de estas, rojas, verdes, azules y genéricas. Cada una de ellas se disipará si poseemos una llave(pastilla) del mismo color que la barrera, mientras que las genéricas nunca se disiparán.

- **Pastillas:** Actúan como llaves dentro del juego y existen tres tipos de ellas, rojas, azules y verdes. Cada una disipa una barrera que coincida en color y se destruye al hacerlo.

- **PowerUps:** Estos objetos aumentaran nuestras estadísticas de forma temporal, existen tres tipos:

    - Snacks: aumentan lo alto que sultán puede saltar
    - Comida: aumenta la velocidad de desplazamiento de Sultán.
    - Pipeta: vuelve a Sultán invulnerable.

- **Checkpoint:** Si morimos, volveremos al último checkpoint visitado.

- **Moriarty:** Este pequeño gato, nos guiará en la zona de tutorial y nos indicará en que punto hemos finalizado la demo.

7. Conclusiones

7.1 Lecciones aprendidas

Durante el desarrollo de este proyecto, el aprendizaje ha sido una máxima siempre presente, especialmente, debido al carácter multidisciplinar necesario para el desarrollo de un videojuego que se ha citado diversas veces a lo largo de la memoria.

A nivel meramente técnico ha resultado un auténtico reto, ya que me he visto obligado a invertir muchísimas horas para documentarme y dominar con soltura herramientas que hasta el momento eran desconocidas para mí, todo en un lapso temporal muy ajustado y con la presión añadida de la autoexigencia por crear algo que no se viese excesivamente “cutre”.

Por otra parte, considero que el haber empleado tantos de los conocimientos adquiridos durante el grado a la hora de cerrarlo con este proyecto, me ha permitido hacer un juicio con retrospectiva, y valorar más que la propia meta algunas de las partes de este camino que ya termina que yo consideraba innecesarias.

7.2 Reflexión sobre los objetivos.

Pese a que estoy satisfecho con el producto final obtenido, creo que mi idea inicial era mucho más ambiciosa y tanto la inexperiencia como la falta de tiempo han pasado factura, obligándome a acotar mucho más el proyecto y a recortar en muchos aspectos y mecánicas de los inicialmente planteados que me hubiese gustado tener tiempo para desarrollar.

No obstante, el roadmap se ha ido reajustando según evolucionaba el proyecto, adaptándose al ciclo de vida de este de una manera sorprendentemente efectiva pese a la ingente cantidad de horas necesarias para ello.

7.3 Análisis de la metodología

Tal como se ha mencionado en el apartado anterior, la planificación del proyecto se ha tenido que ir modificando en diversas ocasiones a lo largo de su ciclo vital con tal de poderse adaptar a los diferentes problemas que han ido surgiendo durante este.

En un inicio, se pretendía desarrollar los tres actos del juego, consiguiendo un producto con una duración estimada de unas diez horas y cubriendo así toda la historia de Sultán. Por suerte, el consultor avisó desde el principio de la inviabilidad de estas pretensiones y de no haber acotado la idea inicial, el proyecto hubiese sido un desastre.

Pese a que en un inicio se percibió como algo negativo, esta reducción ha acabado siendo muy positiva, permitiendo centrarse más en tener una menor cantidad de assets pero más pulidos, que no un proyecto de grandes dimensiones que hace aguas por todas partes.

7.4 Futuro del proyecto

Como se ha venido comentando a lo largo de todo el documento, el proyecto no termina aquí y se tiene intención de desarrollar la historia completa compuesta por los tres actos que narran la vida de Sultán.

Posteriormente, una vez finalizado, me gustaría adaptar el juego para poder ser lanzado en consolas, ya que creo que, al tener unos controles muy sencillos, no sería nada difícil.

Aun así, ambas cosas son proyecciones a largo termino, mientras que las líneas de trabajo con una perspectiva más inmediata serian:

- Revisión integral del core del juego con tal de optimizarlo y eliminar los bugs comentados en el punto 4.4.5.

- Pulir aún más todo el aspecto visual del juego, haciendo especial hincapié en algunas de las animaciones como la de las barreras.
- Añadir más opciones al menú de opciones.
- Añadir movimientos de cámara y efectos de Zoom (tanto in como out).
- Implementar una animación para cuando morimos, la ida seria acompañarla de un fundido a negro mientras esta se ejecuta, algo similar a la de “Zelda- The Ocarina Of Time”
- Usar paralax para el background, desgraciadamente se descubrió esta opción cuando ya era tarde para implementarlo, pero permitiría obtener un producto mucho más limpio.
- Implementar un sistema de vida para el primer acto. La idea es que cada vez que colisionemos con un hermano de Sultán este nos empiece a seguir (con un límite de 5), cuando algún elemento nos dañe, si nos sigue algún hermano, en lugar de morir, el tiempo se ralentizará y aparecerá una mano desde cualquier extremo de la pantalla que se llevará al hermano.
- Es decir, en esencia los hermanos actuarán como puntos de vida.
- Implementar un sistema de diálogos real en lugar de usar imágenes que activamos y desactivamos.

8. Glosario

2D: Acrónimo utilizado para referirse a entornos de 2 dimensiones.

3D: Acrónimo utilizado para referirse a entornos de 3 dimensiones.

Asset: Elemento que puede ser usado en un proyecto.

Bug: error de programación que causa problemas en el correcto funcionamiento de la aplicación

Checkpoint: Punto de control, almacena una posición para spawnear en ella posteriormente.

Colliders: Componente colisionador que define la forma de un objeto a efectos de colisiones físicas.

Engine: En español motor, hace referencia al game engine, entorno usado para desarrollar un videojuego.

Frame: fotograma del juego.

IA: Acrónimo utilizado para referirse a la inteligencia artificial.

Idle: Animación usada para periodos de no actividad o en que el elemento no se desplaza.

Indie: proveniente del término independiente, en el caso de los videojuegos se usa para referirse a aquellos desarrollados individualmente o por equipos pequeños

Raycast: Elemento de Unity que simula el lanzamiento de un rayo en línea recta desde el punto de partida definido y nos da la información del punto de impacto.

Rigidbody: Componente de Unity que permite aplicar fuerzas y torsiones al objeto al que se asigna, se trata de un elemento indispensable en las físicas

Script: Fragmento de código usado para implementar funcionalidades en los objetos.

Spawn: Usado para referirse a cuando el personaje es generado al inicio de la escena.

Sprite: Imagen de mapa de bits presente en el proyecto, usado para referirse a la mayoría **Spritesheet:** Concatenación de sprites mediante los que se consigue una animación.

Trigger: Disparador del proyecto, nos sirve para definir los tempos de los eventos y obtener información de los mismos.

PowerUp: Objeto que potencia las estadísticas del personaje

Prefabs: instancias de un mismo objeto que comparten todas sus propiedades.

UI: Acrónimo utilizado para referenciar a la interfaz de usuario, del inglés “User Interface”.

de los archivos de imagen.

9. Enlaces

9.1 Ejecutable

https://drive.google.com/drive/folders/1gK3zGx8cUNxUwJXA85CSbhStMUnUPUwZ?usp=sharing

9.2 Repositorio de github

https://github.com/Bardinator93/TFG

10.Bibliografía

Brackeys. (10 de octubre de 2021). SCRIPTABLE OBJECTS in Unity. [en línea]: https://www.youtube.com/watch?v=aPXvoWVabPY

Unity. (21 de octubre de 2021). Live Training 5th September 2016 - Creating a Main Menu. [en línea]:

https://www.youtube.com/watch?v=OWtQnZsSdEU

Unity Documentation. Time.deltaTime (22 de octubre de 2021). [en línea] http://docs.unity3d.com/ScriptReference/Time-deltaTime.html

Unity Documentation. Update and FixedUpdate (25 de octubre de 2021). [en línea] https://unity3d.com/es/learn/tutorials/modules/beginner/scripting/update-and-fixedupdate. [Fecha de consulta: 06/03/2016]

Brackeys. (29 de octubre de 2021). HDRP vs. URP - Which Unity Template should you choose? [en línea]:

https://www.youtube.com/watch?v=5MuA92xUJCA&ab_channel=Brackeys

Brackeys. (12 de noviembre de 2021). Introduction to Audio in Unity. [en línea]: https://www.youtube.com/watch?v=6OT43pvUyfY&ab_channel=Brackeys

Multiples Autores. (23 de diciembre de 2021). List of Unity games. [en línea]: https://en.wikipedia.org/w/index.php?title=List_of_Unity_games

Gibson, Jeremy (2015) Introducction to Game Design Prototyping and Development. New Jersey. Addison-Wesley.