Artax: La redención de la llama

Ibai Valtierra Fernandez
Grado de Multimedia
TFG Videojuegos

Ester Arroyo Garriguez
Joan Arnedo Moreno

Domingo, 7 de junio de 2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Artax: La redención de la llama*
Nombre del autor:	*Ibai Valtierra Fernandez*
Nombre del consultor/a:	*Ester Arroyo Garriguez*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega (mm/aaaa):	06/2020
Titulación::	*Grado de Multimedia*
Área del Trabajo Final:	*76.587-TFG-Videojuegos*
Idioma del trabajo:	*Español/Inglés*
Palabras clave	*Unity, Plataformas, android*

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

El objetivo principal de éste TFG es crear un videojuego con el fin adquirir los conocimientos técnicos y de investigación necesarios en todas las fases de desarrollo de un proyecto de estas características, a la vez que se realiza un producto de entretenimiento para el usuario final.

Debido a los plazos y recursos disponibles, se ha creado un juego indie de plataformas 2D tipo arcade. Para su realización, se ha utilizado el motor gráfico Unity junto con el lenguaje de programación C# y una serie de programas para producir el contenido. La aplicación generada tiene como destino cualquier dispositivo que disponga de una pantalla táctil y sistema operativo Android. La metodología seguida para llegar a cabo el proyecto ha sido la planificación y desglose de las fases en tareas a realizar, a las que se les ha asignado un tiempo.

El producto resultante, *Artax: La redención de la llama,* es un juego en el que el personaje realiza un viaje sorteando obstáculos y vence a enemigos para conseguir derrotar al enemigo principal. El jugador debe hacer uso de su habilidad para manejar el personaje a través de los controles táctiles mostrados en la pantalla.

Como conclusión, he llegado a los objetivos marcados y adquirido una buena base para futuros proyectos. Además, he comprobado de primera mano todas las fases en la creación de videojuegos, a la vez de generar un juego de calidad y divertido con un gran potencial.

Abstract (in English, 250 words or less):

The main objective of this TFG is to create a video game to acquire the necessary technical and research knowledge in all phases of development of a project of this nature, while creating an entertainment product for the end user.

An 2D platformer arcade indie game has been created due to the deadlines and available resources. For make it, it has been used the graphics engine Unity together with the C # programming language and a series of programs to produce the content. The generated application is destined for any device that has a touch screen and Android Operating System. The followed methodology to carry out the project has been the planning and breakdown of the phases in tasks to do, and then assign time to them.

The resulting product, Artax: la redención de la llama, is a game in which the character goes on a journey avoiding obstacles and defeating enemies to defeat the main enemy. The player must use his skill to manage the character through the touch controls shown on the screen.

In conclusion, I have reached the goals and acquired a good base for future projects. In addition, I have seen first hand all the phases in the creation of videogames, at the same time as I have generated a quality, fun and with great potential game.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

A lo largo de la historia de los videojuegos, hemos podido comprobar como la industria se vuelve cada vez más grande. Los juegos, cada vez contienen gráficos más realistas, mundos enormes a explorar e infinidad de posibilidades. Son juegos donde el jugador debe invertir infinidad de horas para lograr los objetivos marcados. Ciertamente son juegos impresionantes queridos por los jugadores.

Sin embargo, los jugadores no siempre tienen el tiempo necesario que esos juegos requieren. Además, estos juegos requieren de un hardware con unas características mínimas por lo que la mayoría han de ser ejecutados en consolas u ordenadores de gran potencia.

Por otra parte, la sociedad cada vez se centra más en aplicaciones móviles. Muchas personas juegan a juegos en dispositivos móviles. La mayoría de estos juegos siguen el modelo de negocio “Free to play pay to win”. Este modelo desde un punto económico y comercial muy acertado. No obstante, el hecho de poder pagar para cumplir los objetivos marcados por el juego, creo que puede llegar a ser negativo.

Cuando un jugador juega a un juego lo ve como un reto. Al ofrecer la posibilidad de pasar algo de una manera sencilla, hace que no se valore el esfuerzo.

Artax: La redención de la llama intenta volver a esos orígenes en los que las partidas no son muy largas, pero sí increíblemente divertidas. En las que el jugador pierde los papeles porque siempre se queda atascado en un sitio difícil de sortear, pero siente gran satisfacción cuando lo consigue.

Es un juego con el encanto de los juegos clásicos de plataformas 2D donde no premia la calidad gráfica, pero si la diversión.

1.2 Objetivos del Trabajo

- Creación de un videojuego que contenga dos pantallas.
- La segunda pantalla contendrá un Enemigo final.
- Creación del personaje principal con animación y al menos una mejora de arma o habilidad.
- Creación de al menos dos enemigos básicos con sus animaciones.
- Creación de pantalla de inicio y menú principal.
- Texto con la historia o animática inicial (en caso de tener tiempo).
- Creación de un elemento que el personaje principal pueda romper (vasijas).

- Programación de mecanismo de vidas de los personajes y menús de juego.

1.3 Enfoque y método seguido

Aunque el tiempo para la realización del TFG ha sido escaso, se ha optado por la realización de un producto nuevo. Se ha descartado crear una adaptación porque ésta limitaría aspectos de diseño o de jugabilidad. Además, el crear un producto propio, aunque más simple también implica que pueda ser mejorado posteriormente.

No obstante, aunque el producto es nuevo para agilizar el desarrollo se han utilizado assets en la medida de lo posible en cuanto al apartado gráfico y músicas y sonidos de licencia libre.

El método seguido para su desarrollo ha sido simple. Primero se ha planteado el juego y se ha realizado una lista lo más detallada posible de los elementos necesarios (elementos de las pantallas, personajes, animaciones, músicas, efectos de sonido, etc…).

Una vez que se ha tenido claro lo que se quería hacer se ha tenido que hacer un estudio de los diferentes motores gráficos del mercado para ver cuál podría ser el más adecuado para su ejecución.

A partir de este puto la metodología ha consistido generar un diagrama de Gantt de todas las tareas a realizar e intentar seguir los plazos lo más estrictamente posible.

Después de cada desarrollo importante se han realizado las pruebas pertinentes y en caso positivo se ha hecho una copia de seguridad en el repositorio GitHub.

1.4 Planificación del Trabajo

Tareas a realizar:

- Estudio de Unity y C#
- Diseño de flujo de pantallas.
- Creación de los gráficos de flujo de pantallas (menús/ contadores de inventario).
- Diseño de personajes.
- Animación de personajes.
- Integración de personajes en unity.
- Diseño de pantallas (flujo y posición de elementos, obstáculos, enemigos, objetos).
- Creación de assets y decorados de las pantallas.
- Creación de elementos interactivos (Vasija, piedras preciosas).

- Creación de mejora o habilidad.
- Creación de introducción de la historia del juego texto o animática.
- Programación de interacción de personaje principal (controles).
- Programación de mecanismos de vida de los personajes/ inventario.

Diagrama de Gantt:

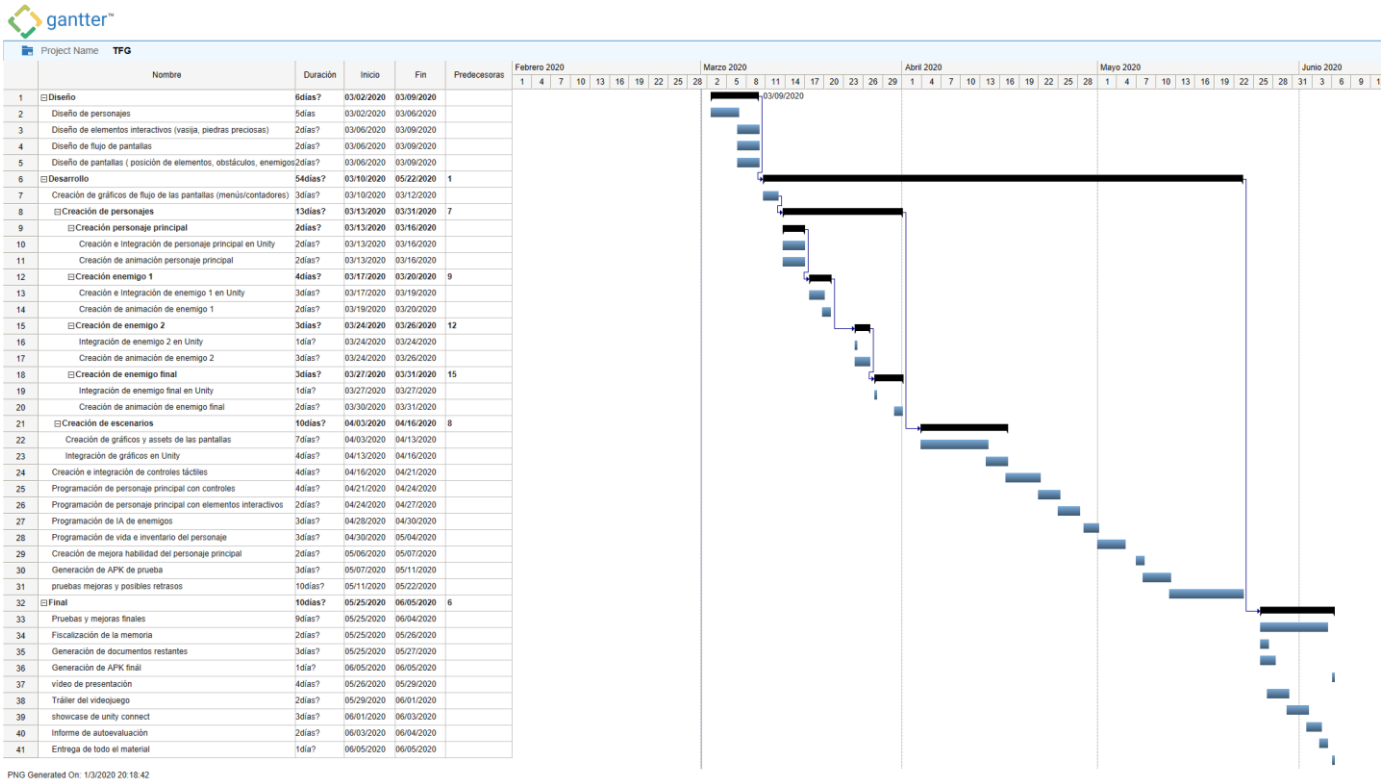


Ilustración 1: Diagrama de Gantt

Recursos necesarios para realizar el Trabajo:

- Unity: Creación del juego.
- Visual Studio: Creación del código.
- Adobe Photoshop, Adobe Illustrator: Generar y modificar componentes del videojuego (personajes, decorados, objetos, etc…)
- Adobe After Effects: Realización de vídeos, animáticas, etc…
- Adobe Media Encoder: Renderizado de vídeos realizados con After Effects.
- Audacity: Gestión de audio.
- Sourcetree: Subir los datos a GitHub.
- Busqueda de diferentes páginas de contenidos (assets, imegenes, etc…).

1.5 Breve sumario de productos obtenidos

- Archivo ejecutable APK.
- Memoria final del proyecto.
- Video tráiler del proyecto.
- Video de defensa del proyecto.
- Repositorio GitHub con el proyecto
- Showcase de Unity Connect

1.6 Breve descripción de los otros capítulos de la memoria

- **2. Idea del juego:** Muestra una breve descripción del juego, un estudio de la historia y referencias a juegos similares, el tipo de interacción entre el juego y el jugador y la plataforma de destino.

- **3. Hardware y engines:** Muestra un listado del hardware utilizado para el desarrollo del proyecto, un estudio de los diferentes motores gráficos, cual se selecciona y los motivos.

- **4. Conceptualización:** Muestra la historia, los personajes, un listado de elementos para la creación del juego, objetivos planteados al jugador, el estilo buscado y un listado de animaciones de los personajes.

- **5. Diseño de flujo de pantallas:** Muestra un esquema de relaciones entre las diferentes pantallas, el diseño en modo de concepto de cada pantalla y se explica su contenido.

- **6. Diseño de elementos:** Muestra los diseños finales y una pequeña explicación de su creación gráfica de los elementos del juego como personajes, objetos, chechkpoints, vasijas, armas, pantallas, etc…

- **7. Desarrollo:** Muestra un listado de todos los programas utilizados y su finalidad y un apartado explicando los desarrollos técnicos más destacados.

- **8. Manual de usuario:** Muestra los requerimientos mínimos necesarios para poder utilizar el juego y las instrucciones de manejo del mismo.

- **9. Estudio de mercado:** Muestra un pequeño estudio con la posibilidad de un negocio futuro.

2. Idea del juego

2.1 Breve descripción del juego

El juego consiste en la creación de un juego indie de plataformas tipo arcade, pero con un estilo más actual. Es decir, el personaje principal irá recorriendo una serie de obstáculos y eliminando enemigos para conseguir su objetivo.

2.2 Subgénero y referencias a videojuegos existentes

Este tipo de juegos se caracterizan por tener una cámara centrada en el jugador de modo que esta se mueve dependiendo de la posición de este, de forma horizontal y vertical. El personaje parte de una serie de acciones como andar, correr, saltar y atacar.

En cuanto a las referencias a otros juegos se pueden encontrar infinidad de ellas. Los pioneros, aunque en este caso la capara era fija son Space Panic (1980) y Donkey Kong (1981).

Ilustración 2:Space panic 1980

Ilustración 3:Donkey Kong 1981

Más adelante aparecieron los primeros juegos con movimiento de cámara como “pitfall!”. No obstante, todo cambió con la llegada de las consolas de 8 bits ya que aparecieron los primeros juegos que se convertirían en grandes clásicos como el “Super Mario Bros”, “Mega Man” o el “ghost and goblins”.

Ilustración 4:Pitfall! (1982)

Ilustración 5:Super Mario Bros (1985)

Ilustración 6:Mega Man (1987)

Ilustración 7:Ghost and goblins (1985)

2.3 Tipo de interacción juego-jugador

La función principal del juego es entretener al jugador, poniendo a prueba habilidades como los reflejos y la coordinación. El jugador tendrá que demostrar estas habilidades mediante la presión de los diferentes botones de acción del juego.

2.4 Plataforma destino

El juego será desarrollado para una plataforma móvil, aunque en un futuro podría ser difundida en otras plataformas. No obstante, en este caso se crearán unos botones táctiles como controles, elemento que tendría que ser eliminado para las plataformas restantes. En el caso de disponer de tiempo y dependiendo de las condiciones podría ser interesante subirlo a alguna plataforma online como Play Store de google o Apple Store de Apple.

3. Hardware y engines

3.1 Hardware

Para la realización de este juego se cuenta con un PC para el desarrollo y un móvil Motorola One Action para el testeo.

- Características del PC:
  - Procesador: AMD FX 8150 3.60GHz Oct acore
  - RAM: 16.0 GB
  - GPU: AMD Radeon HD 6970 2GB GDDR5
  - SO: Windows 10

- Características del móvil:
  - Procesador Samsung Exynos 9609 con CPU Octa-core de 2,2 GHz
  - RAM:4 GB
  - GPU MP3 Mali G72

3.2 Evaluación de engines

Antes de comenzar se ha realizado un estudio de los diferentes engines o motores gráficos y sus características con el fin de elegir el más adecuado para el desarrollo del juego.

Unreal engine:
Es uno de los motores más potentes y utilizados, desarrollado por Epic Games. Sobre todo, se utiliza para juegos AAA entre los que se pueden encontrar *Gears of war 4 o Final Fantasy VII remake*. Es un motor gratuito, pero en el caso de que el producto generado recaude más de 3.000 dólares al mes piden un 5% de las ganancias. Esto facilita mucho la entrada de nuevos desarrolladores, ya que si el producto generado no obtiene un mínimo de resultados económicos no hace falta pagar. Dispone de infinidad de assets y tutoriales para poder aprender y el lenguaje de programación que utiliza es el C++. Además, ofrece la posibilidad de exportar los productos a plataformas como PS4, Xbox One, Nintendo Switch, Steam, Apple store, Play store, Windows, Linux, etc…

Game Maker:
Game Maker es un motor creado por YOYO games. Este motor está diseñado para la creación de videojuegos 2D. Contiene una versión de prueba gratuita, pero en el caso de que se quiera utilizar para una plataforma se tiene que adquirir una licencia que puede variar de 39 a 799 dolares dependiendo de la plataforma. Entre los juegos generados con Game Maker se pueden encontrar juegos como “Hotline Miami”.

Cocos 2D:
En este caso hablamos de un framework escrito en Python y está destinado a la creación de juegos 2D. Este framework está para varias plataformas de desarrollo como Phyton, C++, JavaScript, C#, etc…

Este framework es de código abierto y totalmente gratuito por lo que se ha convertido en uno de los más usados.

Unity:
Es uno de los motores 2D y 3D más utilizados en la actualidad. Al igual que Unreal dispone de una versión gratuita siempre que se cumplan algunos requisitos como no ganar más de 100.000 dolares al año. Entre sus características, cabe destacar que de la lista es el que permite exportar a más plataformas (más de 25). La programación que usa es C#.

3.3 Reflexión de selección de engine

Tras el estudio realizado y habiendo tenido en cuenta otros motores como Construct 2, Godot, etc…, finalmente me he decantado por el Unity para la realización del videojuego. Los motivos son los siguientes.

El game Maker no tiene una versión gratuita y el Cocos 2D por lo que he podido leer tiene muy poca asistencia y documentación para poder empezar por lo que has sido descartados rápidamente.

Por otra parte, por lo que he podido ver el Unreal me parecía muy buena opción y creo que es un motor que me puede gustar mucho. No obstante, por lo que he podido investigar, parece que la curva de aprendizaje de Unreal es la más alta y teniendo en cuenta el tiempo disponible lo he tenido que descartar.

Finalmente, tengo algo de experiencia con Java y por lo que he visto se asemeja mucho a C#, elemento que hace que ahorre tiempo de aprendizaje del lenguaje de programación.

4. Conceptualización

4.1 Historia, ambientación y trama

La trama de la historia se ambienta en historia fantástica, con personajes ficticios, en la que se puede considerar una ambientación medieval - gótico - fantástica. Una vez introducida la historia, el juego se basa en la venganza de Artax sobre Jar y conseguir la libertad para su pueblo. Para ello Artax viajará en busca de Jar eliminando enemigos.

El siguiente fragmento es una introducción a la historia del juego en forma de animática.

“En la gran batalla de la era primera cuando ya no había esperanza y la vistoria se había tornado a favor del ejercito demonio. Entre sus filas había un general que ansiaba el poder del ejército. Cegado por la codicia y el poder, este general, conocido como Jar, decidió traicionar a Pazuzu, el rey de los demonios. Se presentó en medio del campo de batalla e hizo un pacto con Artax el único superviviente de la legión de la llama. Jar le prestó un poder inimaginable a Artax para matar a Pazuzu a cambio de devolver a sus amigos a la vida y retirarse cuando alcanzase el poder. Artax accedió y mató a Pazuzu. Jar se hizo con el poder, pero su maldad no tenía fin y este continuo con la guerra hasta alzarse con la victoria. Artax salió muy malherido y se retiró al exilio con una única idea en la cabeza. Recobrar la libertad de su pueblo y hacer pagar su traición a Jar.”

4.2 Definición de los personajes y elementos

Artax: Personaje principal jugable. Su arma es una espada que podrá intercambiar al obtener mejoras. Su apariencia es un caballero medieval, pero con un toque fantástico.

Jar: Enemigo principal y monstruo final de la demo. Actual jefe de la oscuridad y de los demonios. Sus habilidades son ataque con la espada y lanzamiento de hechizo.

Pazuzu: Antiguo jefe de la oscuridad y de los demonios. Aunque es un personaje que forma parte de la trama solo sale en la animática principal del juego y no será jugable y posiblemente no se vea.

Enemigo 1: Monstruo con garrote. Este tipo de monstruo será el más sencillo de eliminar. Su ataque, el garrotazo, es lento además tiene poca vida.

Enemigo 2: Monstruo con martillo. Este tipo de monstruo será el más complicado de eliminar. Su ataque se basa en lanzar el martillo, por lo que realiza ataques a distancia. Además, este personaje tiene más vida que el enemigo 1.

En diseño de los enemigos secundarios será básicamente el mismo, cambiando de arma y tonalidad de la textura.

Elementos:

- **Fondos (plataformas, elementos de escenarios)**
- **Rocas/piedras**
- **Espadas**
- **Jarras/Vasijas**
- **Joyas (roja, verde, azul)**

4.3 Objetivos planteados al jugador

El objetivo es que el jugador vaya superando las pantallas eliminando a los enemigos y obstáculos que encuentre por el camino con la finalidad de enfrentarse al enemigo final. Por el camino deberá obtener las mejoras para su arma y recolectar piedras preciosas que se convertirán en una vida extra al llegar a 100 unidades (las verdes tendrán valor 1, las rojas 5 y las azules 10) con la finalidad de llegar lo mejor preparado a su enfrentamiento final.

4.4 Concept Art

El tipo de diseño que se busca es una estética 2D de tipo cartoon. Con un dibujo no muy complicado (en tal caso se mejoraría en una segunda versión).

- **Personajes:**

Ilustración 8:referencias estilo personajes

- **Pantallas:** El juego transcurrirá en dos pantallas inicialmente. La primera comienza desde la cabaña del caballero que se adentra en el bosque y que cada vez se hace más sombrío hasta que llega a la guarida del enemigo final. La segunda transcurrirá dentro de la guarida del enemigo final.
- **Animaciones:**
  - **Artax:**
    - Correr
    - Saltar
    - Atacar
    - Recibir daño
    - Morir
  - **Jar:**
    - Caminar
    - Atacar 1 (espada)
    - Atacar 2 (hechizo)
    - Recibir daño
    - Morir
  - **Enemigo 1:**
    - Andar
    - Atacar
    - Recibir daño
    - Mori
  - **Enemigo 2:**
    - Andar
    - Atacar
    - Recibir daño
    - Morir

5. Diseño de flujo de pantallas

5.1 Definición de flujo básico de las pantallas del juego

El juego comienza con una splash screen en la que se ve el logotipo creado y una pantalla de “made with Unity” de aparición obligatoria por tratarse de la versión gratuita. A continuación, aparece la pantalla de menú principal en la que se cargan 3 botones “Jugar”, “Configuración” y “Salir”.

Pulsar el botón “Jugar” comenzará el juego: en primer lugar, se verá la introducción del juego que se podrá saltar pulsando un botón “saltar”. Desde la introducción del juego se accederá a la primera pantalla en la que pueden suceder 3 opciones.

a) El jugador muere en la pantalla sin haber consumido las vidas restantes: Vuelve al inicio de la pantalla o al último checkpoint activado.

b) El jugador muere en la pantalla habiendo consumido las vidas restantes: El juego muestra una pantalla con la frase “Fin de partida” y un botón con el texto “volver”. Tanto si el usuario pulsa el botón volver como si no realiza ninguna acción (pasando un periodo de tiempo) el juego volverá a la pantalla a la pantalla del menú principal.

c) El jugador finaliza la pantalla: Accede a la pantalla 2

Pulsar el botón “Configuración”: El usuario accederá a las opciones del juego en el que podrá nivelar el volumen de la música y de los efectos de las opciones de juego.

Pulsar el botón “Salir”: La aplicación se cerrará.

A lo largo de la partida el usuario podrá pulsar sobre un botón en forma de engranaje el cual pausará el juego y mostrará las opciones de sonido las mismas que en la opción configuración de la pantalla del menú principal, junto con botones de reanudar, “Seguir” para continuar con la partida y “Menú” para volver a la pantalla del menú principal (esta última opción hace que se pierdan los avances del juego).

Cuando llegue al final de la pantalla accederá a la siguiente pantalla.
En la pantalla 2 sucederá lo mismo que en la pantalla 1 solo que al finalizar la pantalla en vez de acceder a una tercera pantalla jugable se le mostrará una pantalla con el texto “Enhorabuena! El mal ha sido destruido”. Al de un tiempo volverá al menú principal.

5.2 Diagrama de flujos del juego

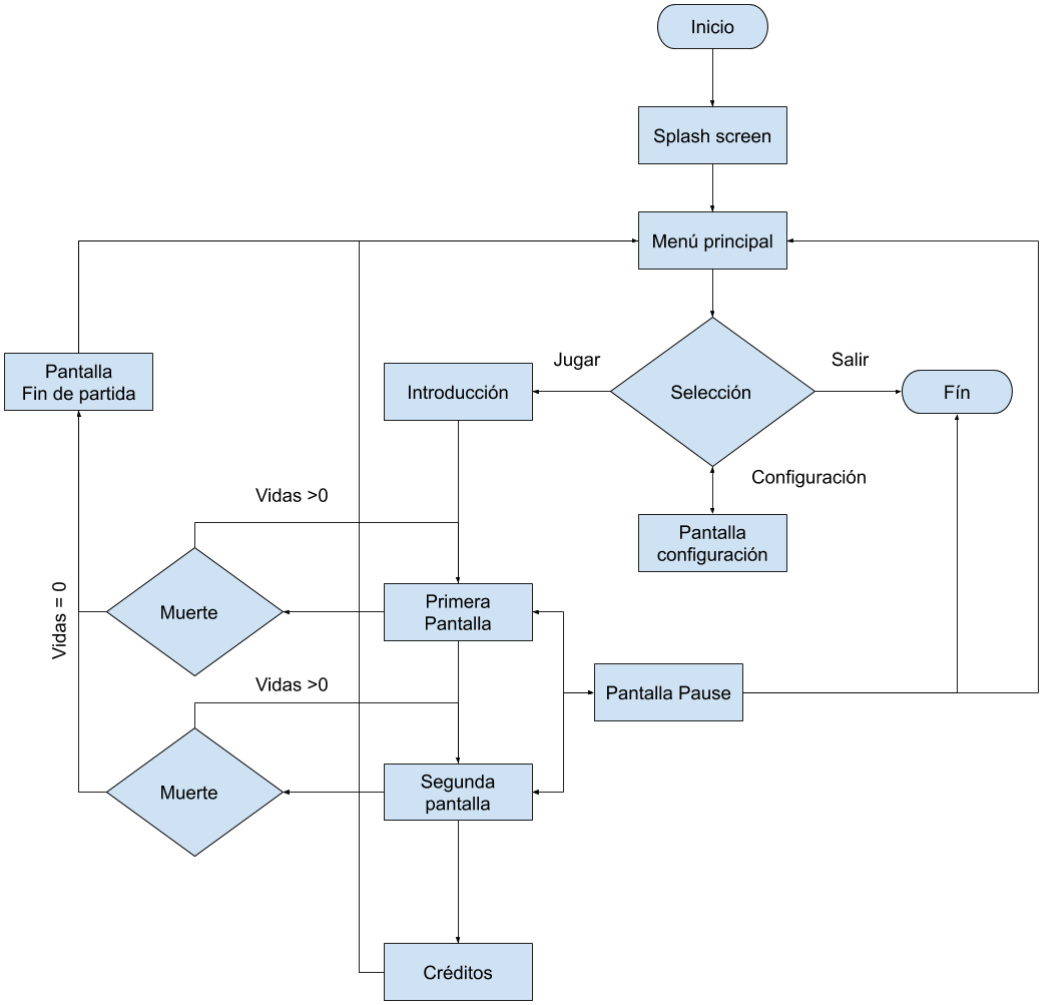


Ilustración 9: Diagrama de flujos del juego

5.3 Diseño de pantallas de flujo básico

Splash screen: Dado que el proyecto se está haciendo con la versión gratuita de Unity, que aparezca su logo es inevitable ya que no se puede eliminar en esta versión. Por lo que en el splash screen aparecerá el logo con una animación (la que facilita Unity) y otra generada propia la que saldrá una pequeña animación con el logo que se creará.

Animación logo Unity

Ilustración 10: splash screen Unity

Animación logo

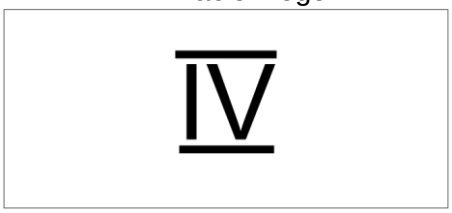


Ilustración 11: Splash screen creada

Menú principal: El menú principal mostrará una composición con los dos personajes principales Artax y Jar, junto con el título del juego “Artax la redención de la llama” y los botones de acción.

Ilustración 12: Diseño pantalla menú principal

Pantalla configuración: La apariencia de la pantalla de configuración será igual que la del menú principal cambiando la zona de botones y el título del juego por la palabra “Configuración”.

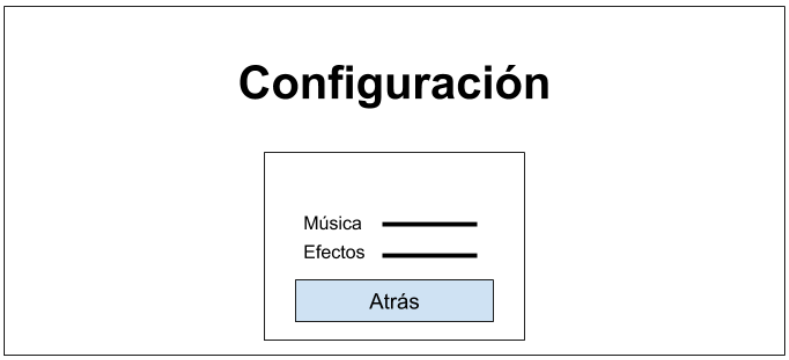


Ilustración 13: Diseño opción configuración de menú principal

Introducción: La pantalla de introducción será un video sencillo en el que se cuente la historia de la trama del juego. En esta pantalla aparecerá un botón “Saltar”, con el que el usuario se podrá saltar la introducción e ir directo a la primera pantalla del juego.

Ilustración 14: Diseño pantalla introducción

Pantalla juego (primera y segunda): En las pantallas del juego se mostrarán, los botones de acción del personaje en la parte inferior y las vidas restantes y las opciones (engranaje) en la parte superior. Al pulsar sobre este último botón, el usuario accederá a la pantalla pause.

Ilustración 15: Diseño pantallas de juego

Pantalla pause (engranaje): Esta pantalla ofrece al usuario ajustar las opciones de sonido, seguir jugando, volver al menú principal. Mientras el usuario esté en esta pantalla el juego se pausará. En el caso de que vuelva al "menú" el usuario perderá todos los avances.

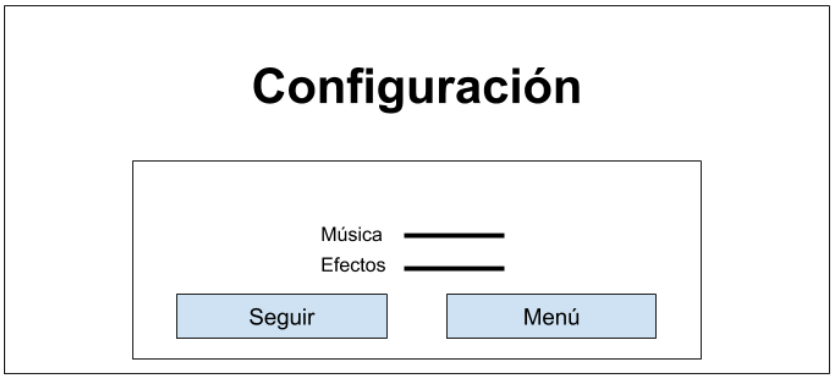


Ilustración 16: Diseño pantalla configuración/pause

Pantalla fin de partida: Esta pantalla aparece cuando el personaje pierde todas las vidas. Después, volverá automáticamente a la pantalla del menú principal.

Ilustración 17:Diseño pantalla fin de partida

Pantalla de créditos: Esta pantalla aparecerá una imagen con la frase “Enhorabuena! El mal ha sido destruido”. Pasados unos segundos la aplicación mostrará de nuevo la pantalla del menú principal.

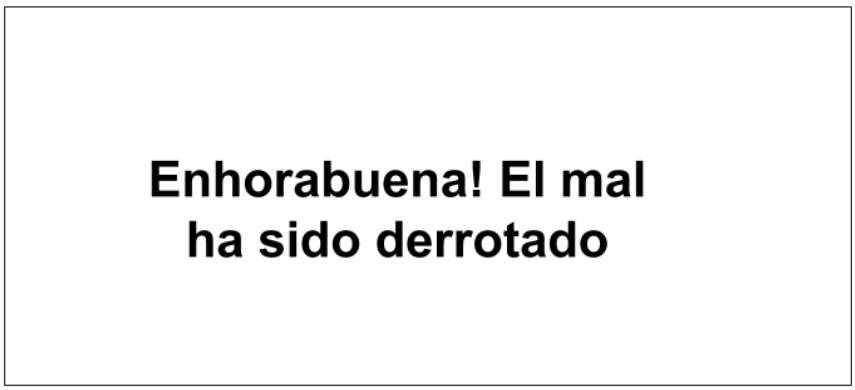


Ilustración 18:Diseño pantalla fin de juego

6. Diseño de elementos

6.1 Personajes:

Artax:

Para la realización del personaje principal se ha seleccionado un asset ya creado al que se le han realizado modificaciones para conseguir el resultado deseado.

Partiendo del asset libre de derechos que se puede encontrar en el siguiente link https://craftpix.net/freebies/2d-fantasy-knight-free-sprite-sheets/

Ilustración 19:Asset referencia Artax

He creado el personaje principal mediante modificaciónes primero con Abobe Ilustrator y posteriormente con Adobe PhotoShop hasta conseguir el el resultado final.

Ilustración 20:Artax diseño final

Aunque el Asset venía con animaciones al generar un nuevo personaje a raíz de estos, las animaciones quedaron inserbibles así que le he generado un sistema de huesos y le he realizado las animaciones creando clips de animación con la opción “Animation” del Unity.

Ilustración 21:Artax rigging

Jar:

El diseño de Jar parte del mismo asset que Artax, por lo que se ha tenido que tratar de la misma manera creando el esqueleto y las animaciones con la opción “Animation” del Unity. Este enemigo aparte de golpear con su gran espada también lanza bolas. Estas bolas han sido creadas con Photoshop.

Ilustración 22: Jar diseño final

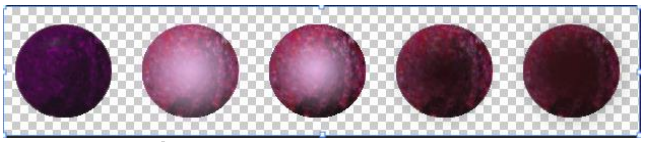


Ilustración 23: Srpite generado de bolas de ataque

Enemigo 1 y 2

Estos personajes se han obtenido del asset del siguiente link https://craftpix.net/freebies/2d-fantasy-trolls-free-sprite-sheets/ .

Ilustración 24: Asset enemigos

Al igual que el asset de Atax y Jar este asset es de licencia libre. En este caso el paquete ya venía con animaciones por lo que prácticamente se han usado sin modificaciones. La única modificación destacable es que el enemigo 2, el del martillo venía unido por lo que modifique los sprites de ataque para que lo lanzara.

Ilustración 25: Ataque enemigo 2

6.2 Objetos:

Vasijas:

A lo largo de las pantallas el personaje se encuentra con vasijas. Si las golpea con el arma, estas se rompen y muestran su contenido. El personaje tiene que coger el contenido, de lo contrario pasados unos segundos este desaparece. Si el personaje se muere al cargar la escena las vasijas que haya destruido no se restauran. En su interior se pueden encontrar los siguientes elementos:

- Corazón: restaura un corazón de vida.
- Joya azul: suma 10 unidades al contador de joyas.
- Joya roja: suma 5 unidades al contador de joyas.
- Joya verde: suma 1 unidad al contador de joyas.

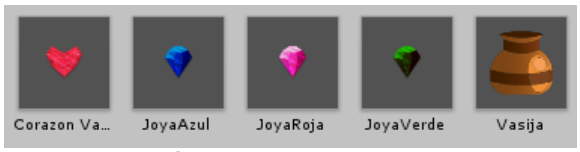


Ilustración 26: Vasijas y contenidos

Ilustración 27: Vasija rota

* La vasija es de creación propia, los elementos del interior han sido obtenidos del asset del siguiente link https://craftpix.net/freebies/free-crystals-2d-game-items/ . Este asset también se utiliza para los contadores de vida y joyas de los niveles

Ilustración 28: Joyas y corazon

Checkpoint:

El checkpoint es una piedra con una luz rondando alrededor. Cuando el personaje pasa sobre este la luz acelera su giro, se coloca en el centro y aparece la marca. Cuando el personaje pasa por un checkpoint en el caso de que muera, en vez de comenzar desde el punto inicial de la pantalla, aparecerá en el punto del checkpoint.

Ilustración 29: checkpoint

* Para su creación, la luz que vuela al rededor es de creación propia, para el resto se ha utilizado el asset gratuito creado por NEMO-MK2 del siguiente link https://assetstore.unity.com/packages/2d/environments/too-cube-forest-the-free-2d-platformer-game-tile-set-117493#content de la asset Store de unity.Este asset se ha usado también para el diseño de los niveles.


Ilustración 30: Asset checkpoint

Mejora de arma:

Al coger este objeto el arma del personaje cambiará. Esta arma inflige el doble de daño a los enemigos.


Ilustración 31: Mejora de arma

* Para crear el arma se ha tomado como referencia el arma original de Artax y se ha modificado con Adobe Ilustrator.

6.3 Niveles:

Pantalla 1:

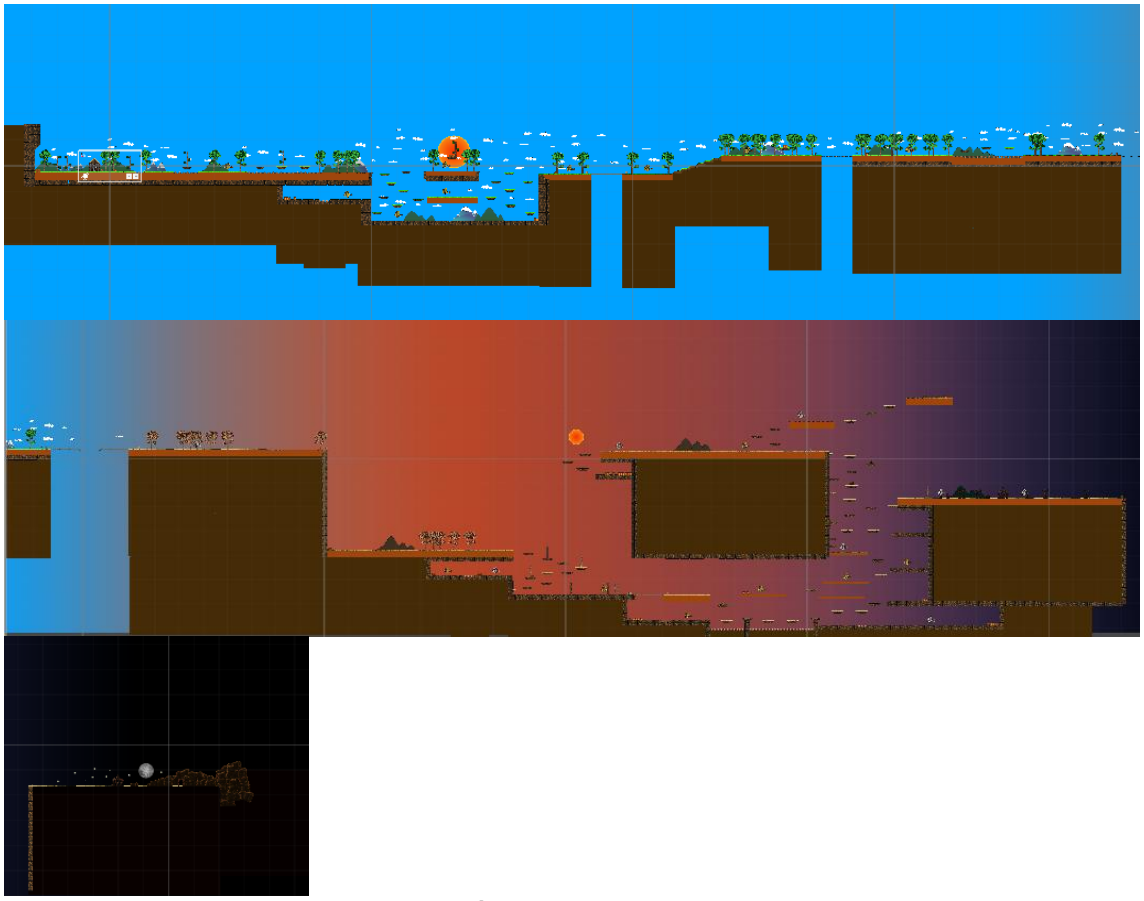


Ilustración 32: Diseño pantalla 1

La primera pantalla está dividida en varias franjas para ofrecer al jugador una sensación de viaje. Al comienzo el personaje sale de su casa en un día soleado. En esta franja solo se encuentra un tipo de enemigo (el más sencillo de vencer). En la segunda franja (el atardecer) el personaje se encuentra con enemigos sencillos, pero también la variante que arroja martillos. Además, los obstáculos que se encuentra son más complejos de sortear al tener más plataformas y zonas de pinchos.

En la última franja solo se encuentra enemigos del segundo tipo, cuando el personaje toca la puerta de la cueva la pantalla finaliza y pasa a la siguiente.

Para la creación de esta pantalla se ha utilizado:

- El asset gratuito creado por NEMO-MK2 del siguiente link https://assetstore.unity.com/packages/2d/environments/too-cube-forest-the-free-2d-platformer-game-tile-set-117493#content de la asset Store de unity (el mismo que para el checkpoint).

- El asset gratuito creado por Szadi Art. del siguiente link https://assetstore.unity.com/packages/2d/environments/pixel-dark-forest-136825

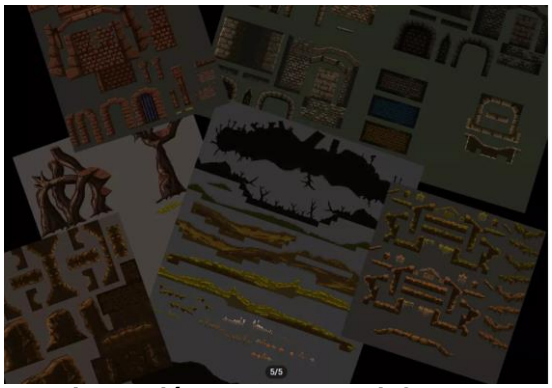


Ilustración 33: Asset Dark forest

- El resto de componentes son de creación propia como: El sol, la luna, las montañas y las nubes y el fondo.

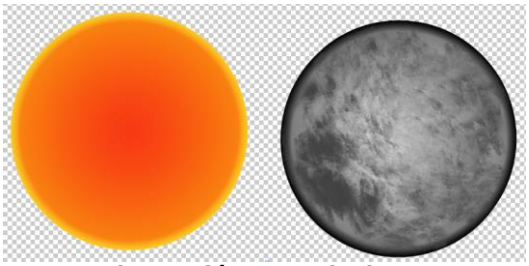


Ilustración 34: sol y luna

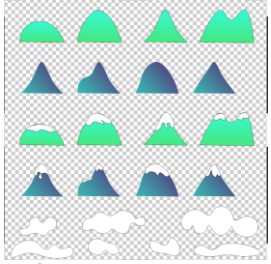
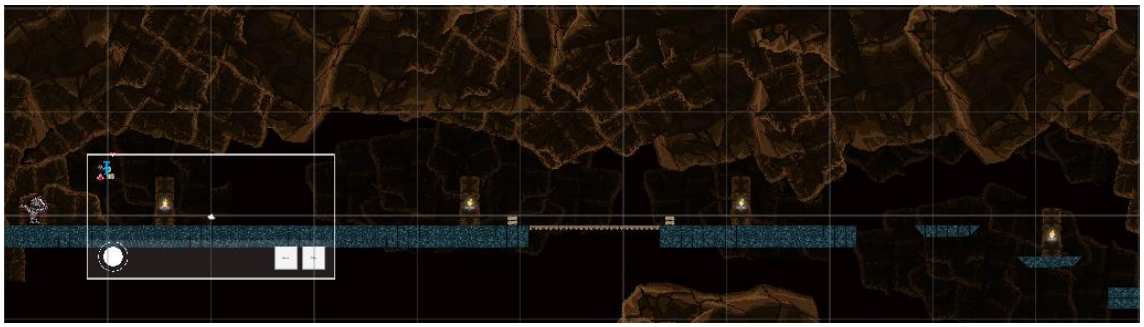


Ilustración 35: Montañas y nubes

Ilustración 36: Degradado fondo

Pantalla 2:



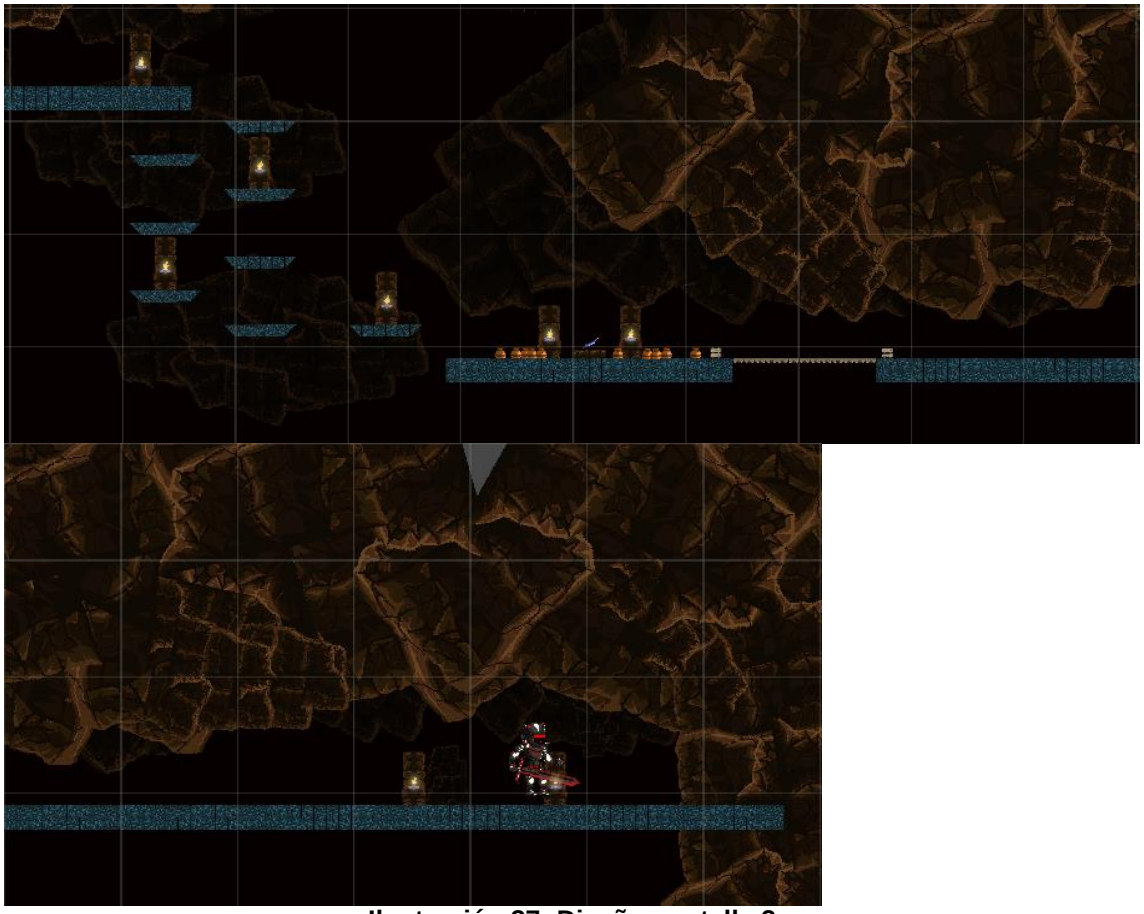


Ilustración 37: Diseño pantalla 2

La segunda pantalla es la guarida del enemigo final. En cuanto a complejidad es sencilla ya que no dispone de enemigos y los obstáculos a sortear no son complicados. En mitad de la pantalla el personaje podrá coger una mejora de arma. Al final de la pantalla el personaje se encuentra con el enemigo final, si le vence se mostrará la escena de fin de juego, de lo contrario se le mostrará la pantalla fin de partida.

Para la creación de esta pantalla se han utilizado los mismos assets que para la pantalla 1.

Cabe destacar el candelabro y la llama de creación propia. El brillo es un degradado al que se le ha animado la escala.

Ilustración 38: Candelabro y llamas

6.4 Pantallas finales:

Ilustración 39:Splash final

Ilustración 40:Menú final

Ilustración 41:Configuración final

Ilustración 42:Introducción final

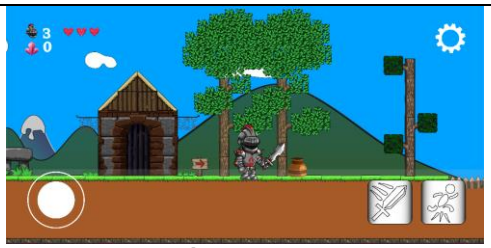


Ilustración 43:Juego Final

Ilustración 44: Pause final

Ilustración 45:Fin de partida final

Ilustración 46:Fin de juego final

7. Desarrollo

7.1 Programas utilizados:

- **Unity:** Motor utilizado para la creación del videojuego.
- **Visual Studio:** Entorno de desarrollo para la creación del código en lenguaje C#.
- **Adobe Ilustrator:** Editor de gráficos vectoriales para la creación y modificación de los elementos gráficos.

- **Adobe Photoshop:** Editor de imágenes para modificar y exportar los elementos gráficos en la resolución y formato deseados.
- **Adobe After Effects:** Programa de composición para la creación de los vídeos del juego y entregables.
- **Adobe Media Encoder:** Motor de codificación para exportar los vídeos creados a los formatos deseados.
- **Audacity:** Programa de grabación y edición de audio para la gestión de los audios.
- **SourceTree:** Cliente GUI para el manejo del repositorio GitHub.

7.2 Desarrollos destacados:

Menú principal:

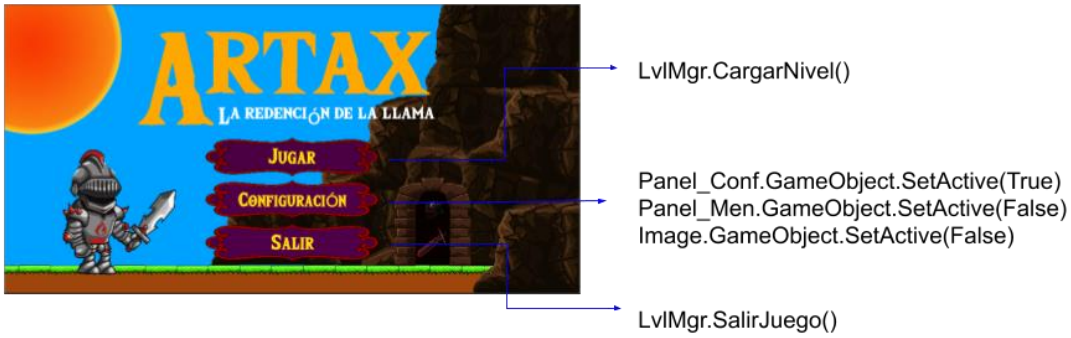


Ilustración 47:Opciones menú

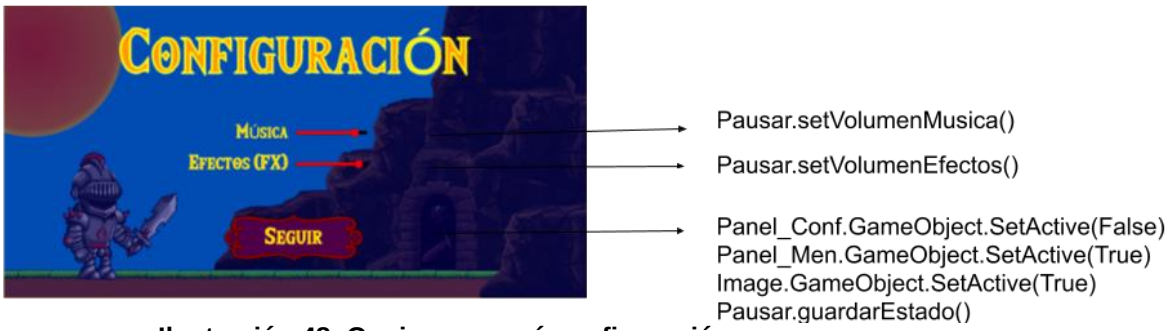


Ilustración 48: Opciones menú configuración

El menú principal consta de 3 botones. El botón “Jugar” y “salir” hacen referencia a la función “LvlMnr()” que sirve para gestionar los niveles. El botón configuración realiza las acciones ocultar el panel de botones principal, ocultar la imagen del título del juego y mostrar el panel de configuración en el método “onClick”.

El panel de configuración muestra 2 sliders que cambian el valor del Del Mixer al que apuntan. El slider de “efectos(FX) apunta al Mixer “Efectos” y el slider “Música” apunta al Mixer “Musica”.

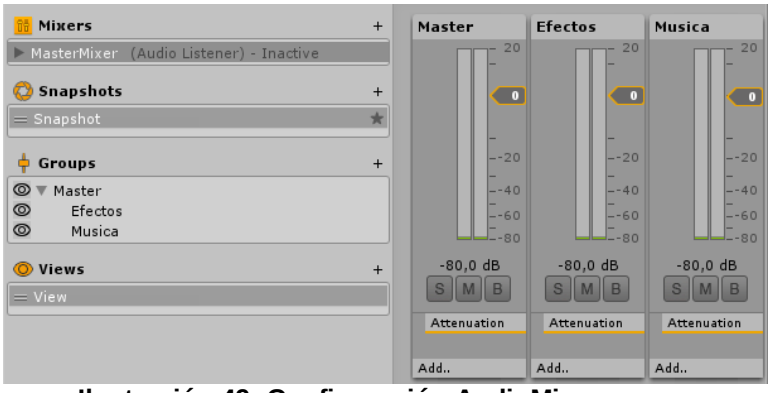


Ilustración 49: Configuración AudioMixer

El audioMixer està compuesto por dos subgrupos “Efectos” y “Musica” a los que se les ha asignado todos los elementos sonoros del juego de tal manera que modificando estos valores se puede controlar el volumen de la música y de los efectos por separado.

Finalmente, el panel de configuración cuenta con botón “Seguir” que al pulsarlo oculta el panel de configuración, muestra el panel de menú, muestra la imagen del título del juego y guarda los valores asignados a los sliders.

Estos valores se guardan en un Script de tipo static con variables también de tipo static para poder acceder a ellas en todo momento. De este modo cada vez que se carga una pantalla carga los valores guardados en la variable estática al valor de los sliders y del los Mixers.

El funcionamiento del panel de configuración de las pantallas de juego cuando se clica el botón con forma de engranaje realiza las mismas acciones salvo que en este caso también dispone de un botón “Menú” que vuelve a la pantalla del menú principal y guarda los datos de los sliders.

Cámara:

La cámara principal de la escena contiene un script “camaraFollow” que se encarga de seguir al personaje. Para ello obtiene la posición del personaje y le asigna el valor con un retardo para que el efecto no sea tan brusco.

Zonas de Muerte:

Ilustración 50: Muerte trampa

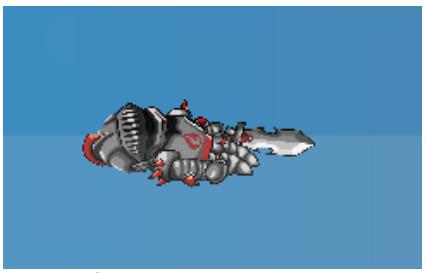


Ilustración 51: Muerte caida al vacio

A lo largo de la partida el personaje puede car en una trampa de pinchos o caer al vacio. En estos casos el personaje perderá una vida. Estos elementos contienen un colider que al detectar la colisión con el personaje llaman a la función muere del personaje.

Checkpoints:

Ilustración 52: Checkpoin

Los checkPoints están generados por varios elementos. En primer lugar, aparece una imagen de una piedra con una luz volando alrededor al que se le ha asignado la etiqueta “checkpoint”. Esa luz tiene un collider de tipo trigger que al contactar con el personaje llama a un script. Este script crea una instancia de la imagen de la piedra grabada y reproduce el sonido asignado.

Por otra parte, el script del personaje principal también detecta la colisión con el checkpoint que guarda la posición en el script “EstadoJuego”.

Vasijas:

Ilustración 53: Vasija desarrollo

Las vasijas en realizad es un objeto al que se le han añadido varios objetos. Cuando el personaje golpea con la espada a una vasija esta llama al script “Basija”.

Este script crea dos instancias. Por una parte, crea la instancia de “basijaRota” que carga un sprite con la vasija en frafmentos y a cada fragmento le aplica una fuerza. Estos fragmentos desaparecen pasados dos segundos para no cargar la escena. Por la otra carga la instancia del objeto (joya roja, joya verde, joya azul, corazón) que le llegue con la variable tipo mediante un switch. El objeto comienza una cuenta atrás de 5 segundos y si el personaje no lo coge en ese intervalo de tiempo el objeto desaparece. Estos objetos tienen un collider y dependiendo de si es una joya o es un corazón llaman a las funciones “sumaJoya” o “sumaCorazon” del script del personaje.

- **SumaJoya():** guarda el número de joyas en la variable “nJoyas” del scrip “EstadoJuego” y actualiza el contador. Si el número de joyas supera 100 se le resta 100 al valor para que nunca pase de 100 y añade una vida actualizando la variable “nVidas” del script “EstadoJuego” y del contador de pantalla. Finalmente destruye el objeto joya con el que ha colisionado.
- **sumaCorazon:** Si el valor de la variable “vida” es inferior a 3 se añade una unidad y actualizará los corazones de la pantalla.

Plataforma móvil:

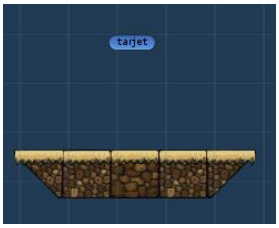


Ilustración 54: Desarrollo plataforma móvil

Las plataformas móviles mueven su posición hasta un objeto llamado “tarjet”. Cuando el valor de la posición de la plataforma es igual a la de la posición del objeto “tarjet”, al objeto se le cambia la posición a la posición inicial de la plataforma.

Cuando el personaje accede a la plataforma se emparenta a la plataforma para que esta le lleve y cuando salta fuera de ella se vuelve a desemparentar.

Pantalla finalizada:

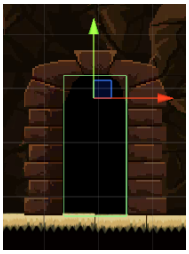


Ilustración 55: pantalla finalizada trigger

Cuando el personaje toca el trigger, se guardan las variables de numero de joyas y vidas en el script “EstadoJuego” (dependiendo de la escena que se vaya a cargar se inicializan las variables) y se carga la escena siguiente.

Mejora de arma:

Ilustración 56: Mejora de arma desarrollo

Este objeto contiene un collider y una etiqueta “Arma”. Cuando el personaje colisiona con el objeto este se destruye, y el personaje esconde el arma anterior y muestra ésta (la lleva oculta y realiza un cambio). También oculta las partículas y muestra las partículas del arma nuevo y se cambia la variable “danioArmaActual”.

* Actualmente solo tiene el arma básica y esta es la mejora por lo que con un cambio es suficiente. En el caso de que se le asignarán más armas se podría añadir una variable “tipo” y dependiendo del tipo cargar el arma deseada.

Gestión de espacio Jar:

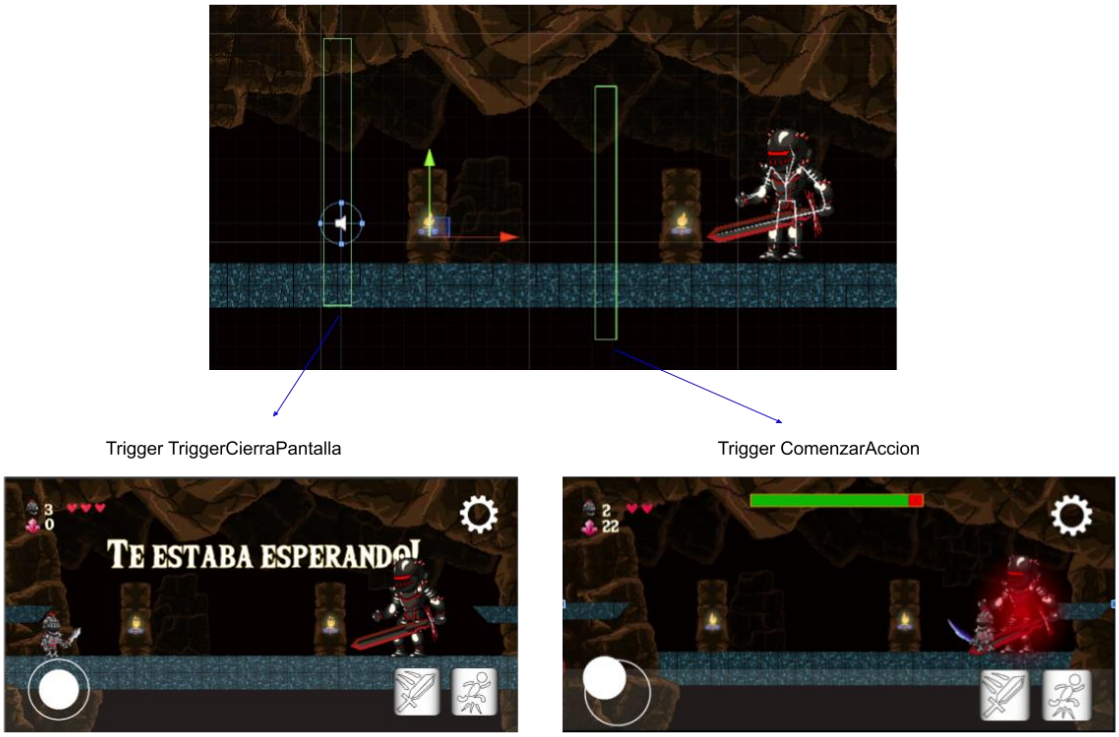


Ilustración 57: Gestión espacio Jar

Cuando el personaje llega al lugar donde está el enemigo final, para llegar hasta él tiene que pasar por dos triggers. El primero bloquea la pantalla mostrando una pared que contiene un collider para que el personaje no pueda retroceder y muestra un texto “te estaba esperando!”. Además, desvincula el movimiento de la cámara del personaje para que se quede estática. El segundo hace comenzar la acción. Oculta el texto, cambia el enemigo (cambia el personaje con solo idle por el personaje con las dinámicas de ataque) y muestra la barra de vida.

Artax:

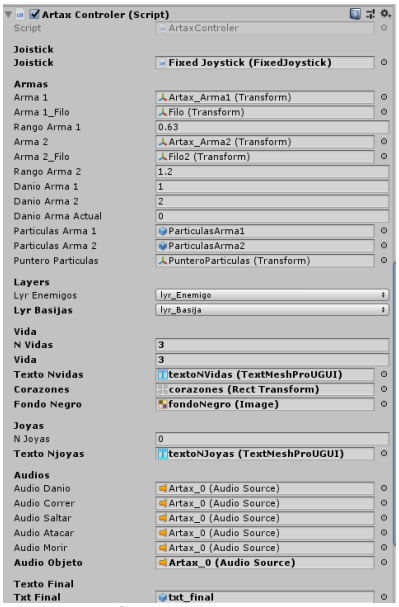


Ilustración 58: Variables Artax

El personaje principal interactúa con todos los elementos de la escena desde la cámara hasta el suelo que pisa. Por ese motivo tiene el script de mayor tamaño. A continuación, se explicarán algunas de sus funciones:

- **Movimiento:** El movimiento del personaje se consigue a través de 3 elementos. El joistick para la dirección (izquierda y derecha) y dos botones (atacar y saltar). Cada uno de ellos está vinculado a una acción.
    - Joistick: A través de un script descargado de un asset (script jpistick) se consigue la interacción entre el usuario y la animación del control. Cuando el control es desplazado hacia cualquier dirección el script del personaje recibe a través de la variable “Joistic.Horizontal” su desplazamiento que lo traduce en desplazar al personaje a la dirección a la que se ha desplazado el joistick. Esta acción también activa la animación corren a no ser que el personaje este saltando.

    - Botón atacar: Cuando el usuario pulsa este botón, el script del personaje detecta si el arma ha colisionado con un enemigo o una vasija y en caso positivo realiza las acciones pertinentes. Si es enemigo llama a la función “recibirDanio” del script del enemigo enviándole el valor del daño que le ocasiona el arma. Si es una vasija, llama a la función “destruir” del script “basija”.

    - Botón saltar: Cuando el usuario pulsa este botón, el rigid body del personaje recibe una fuerza lo que hace que suba y después baje por la fuerza de la gravedad. Cuando el

personaje está subiendo el script activa la animación “esSalto” y cuando está descendiendo la animación “esCaida”.

- **CargarEstadoJuego y GuardarEstadoJuego:** Estas dos funciones sirven para cardar el número de vidas, la posición del personaje, el número de joyas, los enemigos restantes, las vasijas restantes y los checkpoints. Son la unión con la clase “EstadoJuego” y se usan cuando se carga la escena o cuando el personaje muere o se pasa la pantalla.
- **Colliders:** Los collider detectan si el personaje ha cogido un arma, el objeto de una vasija, si está sobre una plataforma móvil o si ha finalizado la pantalla o el juego.

- **Gestión de pantalla:** El script del personaje también se encarga de cargar los valores a los contadores de la pantalla como el número de joyas o el número de vidas.

Enemigos:

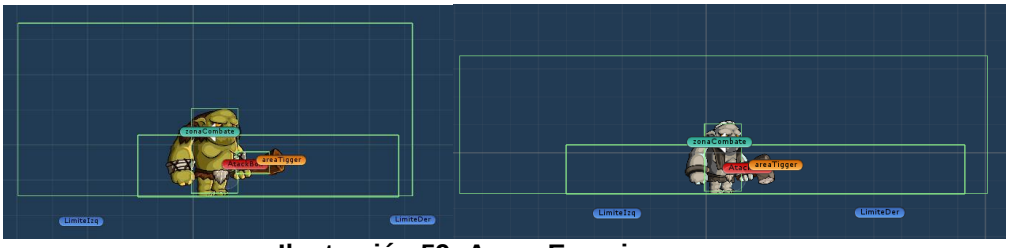


Ilustración 59: Areas Enemigos

Los enemigos disponen de la misma dinámica. Lo único que cambia es su rango, la vida y la velocidad de desplazamiento y de ataque.

Su funcionamiento es el siguiente. Si el personaje principal no está en rango el enemigo se limita a desplazarse entre los limites. Cuando el personaje es detectado por el trigger “areaTriger” desactiva este y activa el trigger “zonaCombate”. Cuando el personaje está en el rango de la zona de combate el enemigo se desplaza hasta él y cuando llega al rango de ataque ataca al personaje. Si el personaje se desplaza fuera de los límites y sigue en rango de ataque el enemigo le sigue sin hacer cado de estos. Cuando el personaje sale del rango de ataque, el trigger “zonaCombate” se desactiva, se activa de nuevo el trigger “areaTrigger” y el enemigo vuelve a desplazarse entre los limites.

Mientras que el enemigo está atacando no recibe daño.

Jar:

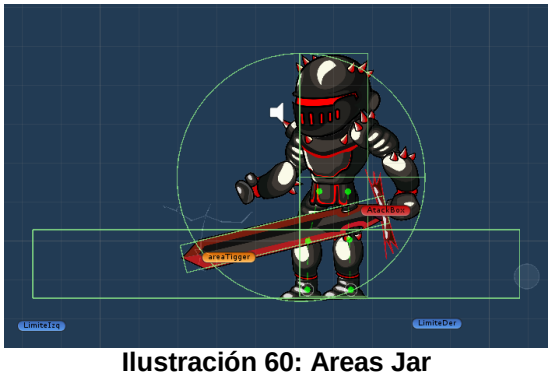


Ilustración 60: Areas Jar

El funcionamiento de Jar es similar al de los enemigos solo que en este caso él siempre se desplaza hasta un límite (el más lejano al personaje) y cuando llega a este límite en vez de pasear lanza el hechizo de bolas. Se mantendrá de esta forma hasta que el personaje entre en el rango del trigger “areaTriger”. En ese puto el funcionamiento será igual que el de los otros enemigos y cuando el personaje esté en el rango de ataque, lanzará un ataque de espada. Otra variante es que Jar siempre está mirando al personaje de tal forma que cuando el personaje sale del rango de ataque de espada y vuelve a la posición del límite para atacar con el hechizo de bolas, no dará la espalda al personaje.

Cuando Jar recibe daño este se ilumina en rojo y durante este intervalo no podrá ser dañado.

EstadoJuego:

EstadoJuego es una clase que contiene la clase estática “EstadoJuego” y una serie de variables que guardan los datos en caso de que el personaje muera, se resten las vidas o no se restauren ni las vasijas ni los enemigos eliminados.

Este script también se encarga de que cuando el personaje cambia de pantalla mantenga las vidas y el número de joyas conseguidos en la pantalla anterior.

8. Manual de usuario

8.1 Requerimientos mínimos técnicos del sistema:

- **Sistema Operativo:** Android 8.1
- **Procesador:** Quad-Core 1,3GHz
- **RAM:** 2Gb
- **Almacenamiento:** 70mb
- **Dispositivo con pantalla multitáctil**

8.2 Instrucciones del juego:

- **Menú**

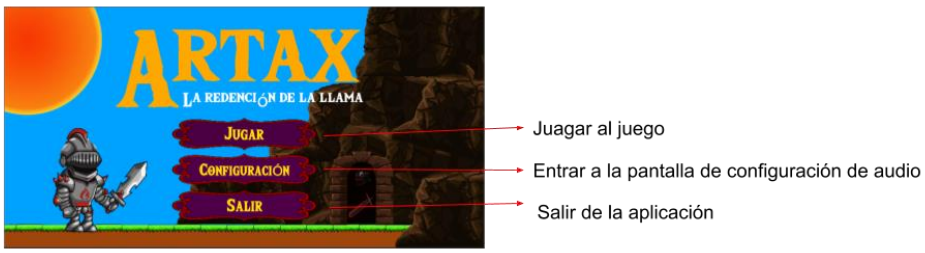


Ilustración 61: instrucciones menú

- **Pantalla de configuración**

Ilustración 62: Instrucciones Configuración menú

- **Pantalla de juego**

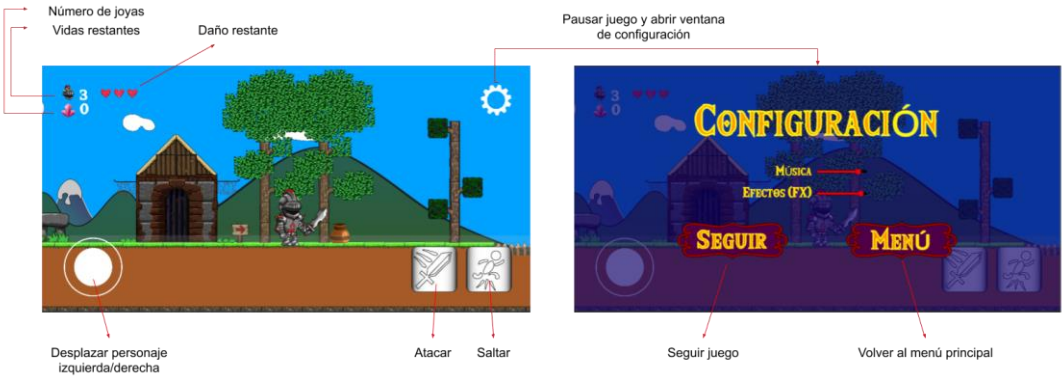


Ilustración 63: instrucciones pantalla juego

9. Estudio de mercado

Con las mejoras pertinentes como la creación de más enemigos, diferentes tipos de mejoras de armas y mayor número de pantallas (tal vez que vayan ampliando la historia), el juego podría ponerse a la venta.

La idea de negocio sería la obtención del juago a través de un pago único. Además, dado que se ha desarrollado en Unity con pequeños cambios en los controles del personaje el juego puede hacerse multiplataforma con el fin de no limitar el juego a usuarios de Android sino expandirlo a otras plataformas móviles, ordenador, web y consolas.

10. Conclusiones

El objetivo de este TFG desde un principio era la obtención de un juego terminado que contara con todos los desarrollos básicos y dejando una posible ampliación en el futuro.

Teniendo en cuenta que era mi primer juego y nunca había tocado Unity ni programado en C# intenté desde un principio no poner los objetivos muy altos. Por ese motivo decidí hacer un juego plataformas 2D. Si hubiera dispuesto de más tiempo estoy seguro de que habría intentado hacer un juego 2.5 D, es decir crear los personajes y decorados en 3D. Sin embargo, era consciente del poco tiempo disponible y lo que suponía la realización de personajes en 3D con sus texturas, rigging y animaciones.

En cuanto al desarrollo se han llegado a todos los objetivos propuestos incluso se han generado más de los propuestos inicialmente. Desde luego ha sido gracias a la utilización de assets, de lo contrario estoy seguro de que no habría llegado a los objetivos o la calidad gráfica sería menor. Aunque en la medida de lo posible he intentado no usarlos, me hubiera gustado prescindir del uso de assets y que todos los elementos gráficos del juego fuesen de creación propia.

En cuanto a la planificación y metodología he intentado seguirlas al máximo. Utilizaba las fechas clave que coincidían con las entregas parciales para saber si el proyecto iba bien o no. Además, gracias al diagrama de Gantt me podía hacer una idea de lo que me faltaba. Es cierto que han surgido retrasos por diversos motivos como desconocimiento o inexperiencia, pero también alguno de mala planificación o mejor dicho de asignar tiempos menos realistas a las tareas. Un ejemplo es la creación de los personajes. En un principio se pretendía que fueran sprites obtenidos de assets pero tras investigar no encontraba nada de mi agrado por lo que tuve que cambiar el sistema de creación. Tras investigar di con el sistema de creación de huesos y animaciones con el animation. Sin embargo, el tiempo que se tarda en tratar un sprite con animaciones ya creadas o realizar un esqueleto y crear las animaciones es diferente.

En cuanto a las líneas de trabajo futuro, como ya he comentado en apartados anteriores, las bases para el juego están finalizadas. A partir de aquí las mejoras posibles serían el intercambio de todos los assets por gráficos de creación propia, mejorar la estética para que quede más homogénea, crear más mejoras de armas, enemigos y pantallas, mejorar los videos de las animáticas y la pantalla de créditos.

Otra línea de futuro importante sería con las mejoras realizadas sacar el juego a mercado para obtener beneficios económicos y dependiendo de los ingresos obtenidos crear otros juegos o una pequeña empresa.

Como conclusión, partiendo de la base que tenía total desconocimiento de la utilización de Unity y C# y que la planificación del juego se realizó según criterios propios, creo que la experiencia ha sido más que positiva. Aunque, he tenido los problemas típicos por desconocimiento creo que el planteamiento fue bastante acertado. Además, he cumplido con los objetivos marcados, el juego es una representación fiel a lo que se tenía en mente en el planteamiento y que es una base con un gran potencial.

11. Glosario

- **TFG:** siglas de trabajo de fin de grado

- **Hardware:** Conjunto de elementos físicos que constituyen un dispositivo electrónico.

- **Free to play pay to win:** Denominación a juegos gratuitos que tienen sistemas de pagos opcionales para avanzar más rápido en el transcurso del juego.

- **Asset:** Recursos que utiliza un videojuego que pueden ser imágenes, sonidos, etc…

- **APK:** Android Application Package. Es un archive ejecutable de aplicaciones para el sistema operativo Android.

- **Indie:** Termino inglés que significa independiente.

- **Arcade:** Juego diseñado siguiendo los principios básicos de los antiguos juegos de máquinas Arcade.

- **Juegos AAA:** Comunmente llamados juegos triple A son juegos desarrollados con un alto presupuesto.

- **Framework:** Es un entorno de trabajo con un conjunto estandarizado de conceptos, prácticas y criterios.

- **Splash screen:** Se denominan pantalla 0 y se presentan al usuario antes de comenzar la aplicación.

- **Rigging:** Proceso de crear un Sistema de huesos y controles a un objeto para que pueda ser animado de forma sencilla.

- **Script:** fragmento de código o comandos para designar a un programa simple.

- **Collider:** Elemento de unity que detecta colisiones entre objetos.

- **Trigger:** Elemento de unity que detecta si un objeto ha entrado dentro de su rango.

- **Checkpoint:** Nombre atribuido a un punto de guardado.

5. Bibliografía

Interés:

- Videojuego de plataformas. (2019, 10 de diciembre). Wikipedia, La enciclopedia libre. Fecha de consulta: 23:59, junio 7, 2020 desde *https://es.wikipedia.org/w/index.php?title=Videojuego_de_plataformas&oldid=121915769.*

- Demonio. (2020, 8 de febrero). Wikipedia, La enciclopedia libre. Fecha de consulta: 23:59, junio 7, 2020 desde *https://es.wikipedia.org/w/index.php?title=Demonio&oldid=123392945.*

- Pazuzu. (2020, 12 de enero). Wikipedia, La enciclopedia libre. Fecha de consulta: 23:59, junio 7, 2020 desde *https://es.wikipedia.org/w/index.php?title=Pazuzu&oldid=122696192.*

Aprendizaje:

- Unity user manual. (2020, 05 de junio). Unity user manual. Fecha de consulta: 23:59, junio 7, 2020 desde *https://docs.unity3d.com/Manual/index.html*

- DGonzalez. (2019, 29 de abril). Tutorial zombies 2D en Unity. Fecha de consulta: 23:59, junio 7, 2020 desde *https://www.youtube.com/watch?v=fo82NGjP_SM&list=PLBxM7h9DJBp5Bfg15JGf-kkXJw-1r6CPY*

- Sykoo. (2018, 01 de octuble). Rig and anímate 2D characters in unity. Fecha de consulta: 23:59, junio 7, 2020 desde *https://www.youtube.com/watch?v=vPgS6RsLIjk*

- Academia de videojuegos. (2015, 21 de octubre). Utilizando el audio Mixer. Fecha de consulta: 23:59, junio 7, 2020 desde *https://www.youtube.com/watch?v=DyTP0nUMHDQ*

- Academia de videojuegos. (2014, 11 de abril). Guardar y cargar. Fecha de consulta: 23:59, junio 7, 2020 desde *https://www.youtube.com/watch?v=zjSfbKq-dKE*

- Sid Makes Games. (2020, 16 de febrero). 2D enemy melee combat in Unity. Fecha de consulta: 23:59, junio 7, 2020 desde *https://www.youtube.com/watch?v=cruE--5ML_Q*

Items y assets:

- freesvg. (2020, 07 de junio). Simple border. Fecha de consulta: 23:59, junio 7, 2020 desde *https://freesvg.org/simple-border*

- Craftpix. (2017, 17 de julio). 2D fantasy knight free character sprite. Fecha de consulta: 23:59, junio 7, 2020 desde *https://craftpix.net/freebies/2d-fantasy-knight-free-sprite-sheets/*

- Craftpix. (2017, 17 de mayo). 2D fantasy trolls free character sprite. Fecha de consulta: 23:59, junio 7, 2020 desde *https://craftpix.net/freebies/2d-fantasy-trolls-free-sprite-sheets/*

- Craftpix. (2018, 20 de junio). Free crystal 2D game items. Fecha de consulta: 23:59, junio 7, 2020 desde *https://craftpix.net/freebies/free-crystals-2d-game-items/*

- NEMO-MK2. (2018, 12 de noviembre). Too cube forest, the free 2D platformer game tile set. Fecha de consulta: 23:59, junio 7, 2020 desde *https://assetstore.unity.com/packages/2d/environments/too-cube-forest-the-free-2d-platformer-game-tile-set-117493*

- Szadi Art. (2019, 26 de agosto). Pixel dark forest. Fecha de consulta: 23:59, junio 7, 2020 desde *https://assetstore.unity.com/packages/2d/environments/pixel-dark-forest-136825*

- Samuel1983. (2020, 07 de junio). Engranaje mecánica configuración icono transmisión. Fecha de consulta: 23:59, junio 7, 2020 desde *https://pixabay.com/es/vectors/engranaje-mec%C3%A1nica-configuraci%C3%B3n-1119298/*

- Fenerax Studios. (2019, 25 de marzo). Joystick pack. Fecha de consulta: 23:59, junio 7, 2020 desde *https://assetstore.unity.com/packages/tools/input-management/joystick-pack-107631*

Audios:

- Tictac9. (2020, 10 de mayo). Epic string ensemble. Fecha de consulta: 23:59, junio 7, 2020 desde *https://freesound.org/people/tictac9/sounds/517540/*

- ptrflr. (2017, 05 de mayo). Choir_bigChurch. Fecha de consulta: 23:59, junio 7, 2020 desde *https://freesound.org/people/ptrflr/sounds/391485/*

- AlexProst. (2017, 27 de febrero). Epic string. Fecha de consulta: 23:59, junio 7, 2020 desde *https://freesound.org/people/AlexProst/sounds/382250/*

- TheBuilder15. (2018, 18 de julio). Collapse. Fecha de consulta: 23:59, junio 7, 2020 desde *https://freesound.org/people/TheBuilder15/sounds/434897/*

- Free-Rush. (2016, 17 de febrero). coins. Fecha de consulta: 23:59, junio 7, 2020 desde *https://freesound.org/people/Free-Rush/sounds/336934/*

- Deathscyp. (2020, 14 de febrero). Deathscyp Factory. Fecha de consulta: 23:59, junio 7, 2020 desde *https://freesound.org/people/Deathscyp/sounds/506016/*

- Josepharaoh99. (2016, 02 de noviembre). Growl and roar. Fecha de consulta: 23:59, junio 7, 2020 desde *https://freesound.org/people/josepharaoh99/sounds/366837/*

- Osiruswaltz. (2016, 16 de octubre). Demon sounds. Fecha de consulta: 23:59, junio 7, 2020 desde *https://freesound.org/people/Osiruswaltz/sounds/362676/*

- Nextmaking. (2009, 19 de diciembre). sword. Fecha de consulta: 23:59, junio 7, 2020 desde *https://freesound.org/people/nextmaking/sounds/86049/*

- FunWithSound. (2017, 01 de mayo). Breaking glass. Fecha de consulta: 23:59, junio 7, 2020 desde *https://freesound.org/people/FunWithSound/sounds/390705/*

- Fax_Box_Creature. (2019, 11 de mayo). For players. Fecha de consulta: 23:59, junio 7, 2020 desde *https://freesound.org/people/Far_Box_creature/sounds/469567/*

- Michel88. (2009, 06 de agosto). deathh. Fecha de consulta: 23:59, junio 7, 2020 desde *https://freesound.org/people/Michel88/sounds/76960/*

- NoiseCollector. (2009, 15 de febrero). Sheet Metal. Fecha de consulta: 23:59, junio 7, 2020 desde *https://freesound.org/people/NoiseCollector/sounds/67376/*

- ShortRecord. (2020, 19 de abril). Enviroment. Fecha de consulta: 23:59, junio 7, 2020 desde *https://freesound.org/people/ShortRecord/sounds/514162/*

- Artmasterrich. (2016, 08 de mayo). Action SFX vocal Kit. Fecha de consulta: 23:59, junio 7, 2020 desde *https://freesound.org/people/Artmasterrich/sounds/345450/*

- qubodup. (2008, 07 de septiembre). whoosh. Fecha de consulta: 23:59, junio 7, 2020 desde *https://freesound.org/people/qubodup/sounds/60013/*

- felixyadomi. (2019, 13 de enero). Jump hop. Fecha de consulta: 23:59, junio 7, 2020
  desde *https://freesound.org/people/tictac9/sounds/517540/*

- nathanaelsams. (2011, 12 de septiembre). Running on grass with wet feet. Fecha de consulta: 23:59, junio 7, 2020
  desde *https://freesound.org/people/nathanaelsams/sounds/127955/*

- mrickey13. (2020, 28 de abril). PlayerHurt1. Fecha de consulta: 23:59, junio 7, 2020
  desde *https://freesound.org/people/mrickey13/sounds/515623/*

- Baltiyar13. (2020, 27 de abril). My melody. Fecha de consulta: 23:59, junio 7, 2020
  desde *https://freesound.org/people/Baltiyar13/sounds/515206/*

- FunWithDound. (2017, 24 de junio). Success triumph resolution sound effect. Fecha de consulta: 23:59, junio 7, 2020
  desde *https://freesound.org/people/FunWithSound/sounds/396174/*

- Airwolf89. (2016, 28 de mayo). Epic_Battle_Music_1. Fecha de consulta: 23:59, junio 7, 2020
  desde *https://freesound.org/people/Airwolf89/sounds/346455/*

- Frankum. (2017, 15 de marzo). Epic chorus-song(no flangerfx problema in cell phones). Fecha de consulta: 23:59, junio 7, 2020
  desde *https://freesound.org/people/frankum/sounds/384308/*