

TITULO DOCUMENTO

Índice de contenido

Contenido

1.	Portada	2
2.	Documento Descripción del proyecto	3
2.1.	Contexto del proyecto	4
2.1.1.	Ámbito y entorno	4
2.1.2.	Análisis de la realidad	4
2.1.3.	Solución y justificación de la solución propuesta	4
2.1.4.	Destinatarios	4
2.2.	Objetivo del proyecto	4
2.3.	Objetivo del proyecto en lengua extranjera	5
3.	Documento de Acuerdo del proyecto	6
3.1.	Requisitos funcionales y no funcionales	7
3.2.	Tareas	12
3.3.	Metodología a seguir para la realización del proyecto	13
3.4.	Planificación temporal de tareas	14
3.5.	Presupuesto (gastos, ingresos, beneficios)	15
3.6.	Ánalisis de riesgos	15
3.7.	Contrato/Pliego de condiciones	17
4.	Documento de Análisis y Diseño	21
4.1.	Modelado de Datos. Análisis y diseño de la base de datos	22
4.1.1.	Diagrama E/R	23
4.1.2.	Diagrama Relacional	23
4.2.	Análisis y diseño del sistema funcional	24
4.3.	Análisis y diseño de la interfaz de usuario. Mockups	37
4.4.	Diseño de la arquitectura de la aplicación	48
4.4.1.	Tecnologías/Herramientas usadas y descripción de las mismas	48
4.4.2.	Arquitectura de componentes de la aplicación	56
5.	Documento de Implementación, Pruebas e Instalación del sistema	58
5.1.	Implementación	59
5.2.	Pruebas	60
5.2.1.	Pruebas Unitarias	60
5.2.2.	Pruebas Funcionales	60
5.3.	Instalación/Despliegue y Configuración	62
5.4.	Manual de Usuario	63
6.	Documento de cierre	65
6.1.	Resultados obtenidos y conclusiones	66
6.2.	Diario de bitácora	66
7.	Bibliografía	67
8.	Anexos	69

1. Portada

Randomly

**Curso:**

2º Dam Diurno 10/04/2022

**Datos Del Alumno:**

Víctor Manuel Pena Gascó

victorpenawow@gmail.com**Datos Profesores Tutores:****Centro:**

CPIFP Los Enlaces

C. Jarque de Moncayo, 10, 50012 Zaragoza

[976 30 08 04](tel:976300804)[Zaragoza](#)

2. Documento Descripción del proyecto

Randomly Víctor Manuel Pena Gascó	Pena_Gasco_Victor_Descripcion Página 4 de 69	 ROSE RANDOMLY
---------------------------------------------	-------------------------------------------------	--------------------------------------------------------------------------------------------------

2.1. Contexto del proyecto

2.1.1. Ámbito y entorno

Realizo este proyecto por motivación propia, sin desmerecer que se le puede sacar partido, no así en el tiempo que voy a invertir en primer lugar, ya que un videojuego requiere de mucho más tiempo que 3 meses para sacar partido de ello.

Mis motivaciones principales para elegir hacer este tipo de proyecto son varias:

- Aprender las tecnologías necesarias para este tipo de proyecto.
- Dedicar tiempo a algo que me apasiona.

2.1.2. Análisis de la realidad

El proyecto parte de 0, con herramientas ya disponibles. Algunas de esas herramientas son: Unity, Angular, BootSpring, Oracle, Visual Studio.

Existen aplicaciones similares, en el store se pueden encontrar miles de videojuegos para móviles, con sus respectivas webs de información.

2.1.3. Solución y justificación de la solución propuesta

Realmente no tengo nada nuevo que inventar/solucionar, quizás sea más difícil la implementación en Apple debido a que es un entorno más cerrado, tengo que picar código y plantear la lógica. La parte de encontrar los assets adecuados también puede resultar relativamente compleja.

2.1.4. Destinatarios

El público objetivo es el usuario apasionado de los videojuegos, en concreto de los retro en el ámbito de los plataformas 2D (Estilo de juego).

Hay un gran abanico de clientes en el entorno de los videojuegos móvil, debido a la comodidad que aporta jugar este tipo de juegos desde el móvil.

Respecto a el juego para Windows tiene el mismo público o más que en móviles debido a la existencia de plataformas como Steam y al trato que le da a este tipo de aplicaciones.

2.2. Objetivo del proyecto

Entrando en más detalles el proyecto consiste en desarrollar un videojuego 2D de tipo plataformas 2D, modelado con Unity con una plantilla destinada para entornos móvil y Windows planteando la lógica de este mediante C#.

Se planteará la aplicación web enlazada al videojuego con su respectiva información. Login de usuarios. En la aplicación web se utilizará Angular para el frontend mediante typescript, javascript, html, css, tailwind y SpringBoot para el backend mediante java y json.

La información del videojuego se persistirá en ficheros en local haciendo uso del patrón de entities y models. De apoyo se aportará una BD a la web, realizada en H2 de spring boot, generada automáticamente gracias a los entitys y los models..

Una vez realizado esto, se integrará en Android Studio y IOS.

Por lo tanto, se hará uso de lo aprendido en DAM, mediante Java en el backend, H2, HTML, CSS, Json, C#

Randomly Víctor Manuel Pena Gascó	Pena_Gasco_Victor_Descripcion Página 5 de 69	
---------------------------------------------	-------------------------------------------------	------------------------------------------------------------------------------------

y se experimentará con Unity, Angular, TS, JS. Se llevará al extremo el trabajar con objetos ya que Unity vive de ello, tanto en el Ide como en las clases en C#, literalmente lleva al extremo la programación con objetos.

2.3. *Objetivo del proyecto en lengua extranjera*

Going into more details, the project consists of developing a 2D platform-type 2D video game, modeled with Unity with a template intended for mobile and Windows environments, proposing its logic using C#.

The web application linked to the video game will be presented with its respective information. User login. In the web application, Angular will be used for the frontend through typescript, javascript, html, css, tailwind and SpringBoot for the backend through java and json.

The video game information will be persisted in local files using the pattern of entities and models. As support, a database will be provided to the web, made in Oracle.

Once this is done, it will be integrated into Android Studio and IOS.

Therefore, what has been learned in DAM will be used, through Java in the backend, Oracle, HTML, CSS, Json, C# and will be experimented with Unity, Angular, TS, JS. Working with objects will be taken to the extreme since Unity lives on it, both in the Ide and in the C# classes, it literally takes programming with objects to the extreme.

3. Documento de Acuerdo del proyecto

3.1. Requisitos funcionales y no funcionales*Requisitos funcionales***Requisito: RF1 Assets**

Descripción	El proyecto debe de tener Assets adecuados al entorno del videojuego.		
Tipo	Funcional	Prioridad	Baja

Requisito: RF2 Crear Timelaps

Descripción	Se debe de crear los timelaps, los cuales deben limitar la escena por la que se mueve el héroe.		
Tipo	Funcional	Prioridad	Media

Requisito: RF3 Movimiento Lateral Héroe

Descripción	Nota: Todo se hará mediante programación en C#. El héroe debe de poder moverse por la escena de forma lateral		
Tipo	Funcional	Prioridad	Alta

Requisito: RF4 Salto Héroe

Descripción	Debe de poder realizar un salto y un doble salto si aun no ha tocado tierra. Por lo tanto, da paso al siguiente requisito funcional.		
Tipo	Funcional	Prioridad	Alta

Requisito: RF5 Detectar Colisiones Entorno Héroe

Descripción	El personaje héroe tiene que detectar las colisiones que ocurren a su alrededor, de esta forma sabrá cuando esta en tierra y cuando en el aire mediante unos colisionadores en el pie. Habilitando ciertos comportamientos dependiendo de su situación.		
Tipo	Funcional	Prioridad	Alta

Requisito: RF6 Detectar Colisiones Daño Héroe

Descripción	De igual forma que detecta el entorno debe de tener otro colisionador en forma de su silueta con el cual detectara el daño entrante.		
Tipo	Funcional	Prioridad	Alta

Requisito: RF7 Detectar Colisiones Arma Héroe

Descripción	El héroe deberá portar un arma con un colisionador con el cual permita hacer daño a otros objetos con colisionadores al entrar en contacto, todo de forma programática.		
Tipo	Funcional	Prioridad	Alta

Requisito: RF8 FeedBack Daño Héroe

Descripción	Al recibir daño en el colisionador deberá realizar un impulso hacia la dirección contraria a la que ha recibido el daño y el recibir daño desencadenara una animación y sonido pertinente.		
Tipo	Funcional	Prioridad	Alta

Requisito: RF9 Animaciones Héroe

Descripción	Se deberá de programar distintas animaciones en función de lo que se requiera en el momento, para correr, estar quieto, saltar, recibir daño y atacar.		
Tipo	Funcional	Prioridad	Media

Requisito: RF10 Sonido Héroe

Descripción	Se programa de igual manera a las animaciones su respectivo efecto de sonido.		
Tipo	Funcional	Prioridad	Baja

Requisito: RF11 Parámetros Héroe

Descripción	El héroe deberá de tener vida, la cual se ira restando o sumando en función de x condiciones, magias para lanzar con sus respectivas cargas y dinero		
Tipo	Funcional	Prioridad	Alta

Requisito: RF12 Girar Héroe

Descripción	El asset tiene que apuntar en la dirección a la que se mueva.		
Tipo	Funcional	Prioridad	Media

Requisito: RF13 Persistir Héroe

Descripción	El héroe deberá de mantener sus parámetros al cambiar de escena. De normal los objetos al cambiar de escena se destruyen y vuelven a crear, reseteando sus parámetros, no así el caso del Héroe, por lo cual se deberá de crear un patrón singleton para no pisar las instancias del héroe y así persistir sus datos.		
Tipo	Funcional	Prioridad	Alta

Requisito: RF14 Programar Enemigos General

Descripción	De misma forma que con el héroe se deberá programar a los distintos enemigos, con la diferencia de que no se podrán manejar, si no que tendrán que tener una IA, así como sus animaciones, efectos de sonido, colisionadores...		
Tipo	Funcional	Prioridad	Media

Requisito: RF15 Enemigo Fantasma

Descripción	Este deberá de seguir al héroe cuando este a la vista dentro de un rango y cuando no esté a rango patrullará la zona.		
Tipo	Funcional	Prioridad	Media

Requisito: RF16 Enemigo Esqueleto

Descripción	Los enemigos tipo esqueleto aparecerán del suelo cuando el héroe este lo suficientemente cerca y deberán andar en línea recta sin seguir al héroe siguiendo una ruta.		
Tipo	Funcional	Prioridad	Baja

Requisito: RF17 Enemigo Bestia

Descripción	Este tipo de enemigo deberá de tener más velocidad que los esqueletos y seguirá al héroe allí a donde vaya, comentar que esto se realizará con rangos de manera programática y siguiendo la cámara del héroe.		
Tipo	Funcional	Prioridad	Media

Requisito: RF18 Comunicación Escenas

Descripción	Las escenas contendrán mapas por los cuales se moverá el héroe y deberán de estar todos interconectados, así como delimitar el entorno. El héroe debe persistir entre las escenas, así como teletransportarse. Esta comunicación se realizará mediante portales invisibles que al detectar la colisión con el héroe cargarán otra escena posicionando al héroe.		
Tipo	Funcional	Prioridad	Baja

Requisito: RF19 Suelo Dañino

Descripción	Se programará un tipo de suelo que matará instantáneamente al héroe al entrar en contacto.		
Tipo	Funcional	Prioridad	Baja

Requisito: RF20 Plataformas Móviles

Descripción	Las escenas tendrán plataformas móviles que seguirán una ruta concreta con IA.		
Tipo	Funcional	Prioridad	Media

Requisito: RF21 Magias

Descripción	Se diseñarán distintos tipos de magias para los enemigos y el héroe, con sus propias colisiones, efectos y daño, así como paráolas distintas.		
Tipo	Funcional	Prioridad	Baja

Requisito: RF22 Coleccionables

Descripción	Las escenas tendrán que contener objetos colecciónables dispersos por el mapa, que servirán de apoyo al héroe, monedas, corazones, antorchas (las cuales desencadenarán otro objeto de manera aleatoria).		
Tipo	Funcional	Prioridad	Media

Requisito: RF23 Boss Final

Descripción	El juego tiene que tener un boss final con sus efectos y animaciones, así como magias propias.		
Tipo	Funcional	Prioridad	Alta

Requisito: RF24 Estados Boss Final

Descripción	El boss final deberá contener distintos patrones los cuales ejecutará de forma aleatoria según la situación.		
Tipo	Funcional	Prioridad	Alta

Requisito: RF25 Presentación Boss Final

Descripción	Habrá una pequeña cinemática al aparecer el boss final, reestructurando programáticamente el escenario, así como la música de fondo.		
Tipo	Funcional	Prioridad	Media

Requisito: RF26 Teletransporte Boss Final

Descripción	El boss se teletransportará por el escenario siguiendo un patrón aleatorio en distintos puntos.		
Tipo	Funcional	Prioridad	Media

Requisito: RF27 Movimiento Boss Final

Descripción	El boss se moverá hacia una dirección aleatoria arrollando a su paso.		
Tipo	Funcional	Prioridad	Media

Requisito: RF28 Lanzamiento Proyectiles Boss Final

Descripción	Deberá lanzar magias en dirección al héroe.		
Tipo	Funcional	Prioridad	Media

Requisito: RF29 Fase Final Boss Final

Descripción	Al alcanzar menos del 50% tendrá una fase final con sus respectivas mecánicas.		
Tipo	Funcional	Prioridad	Media

Tipo	Funcional	Prioridad	Media
-------------	-----------	------------------	-------

Requisito: RF30 Muerte Jugador

Descripción	El jugador debe poder morir y cuando esto ocurra inhabilitar el personaje.
Tipo	Funcional

Requisito: RF31 Pantalla de Inicio

Descripción	El juego debe de tener una pantalla de inicio con unas opciones que se desbloquearan al darle a intro, las cuales son: Nueva partida, Continuar y Salir.
Tipo	Funcional

Requisito: RF32 Pantalla de GameOver

Descripción	Se diseñará otra pantalla con sus respectivos botones y haciendo uso de canvas para cuando se muera, permitiendo reiniciar la escena o ir al menú principal.
Tipo	Funcional

Requisito: RF33 Pantalla Guardar Partida

Descripción	Tiene que haber una pantalla de guardar partida a la cual se podrá acceder al entrar a salas seguras.
Tipo	Funcional

Requisito: RF34 Persistencia de los Datos

Descripción	Se deberá persistir los datos de forma local en ficheros, con el uso de entities y modelos, serializando los datos al guardar partida y deserializandolos al cargar partida. Se guardará información del héroe y de los niveles en general.
Tipo	Funcional

Requisito: RF35 Programar UI

Descripción	El juego tendrá que tener una UI de fondo, con la vida actual, monedas y magias disponibles mientras se juega.
Tipo	Funcional

Requisito: RF36 Cámara Móvil

Descripción	La cámara debe seguir al Héroe
Tipo	Funcional

Requisito: RF37 BackEnd Web

Descripción	Debe devolver datos del personaje mediante una api que se consumirá en angular
Tipo	Funcional

Requisito: RF38 Angular FrontEnd

Descripción	Se debe de realizar el front con tailwind y enlazarlo a la api para que rellene los datos del personaje.
Tipo	Funcional

*Requisitos No Funcionales***Requisito: RNF1 Interfaz UI**

Descripción	El juego debe de tener una interfaz de las pantallas de la UI intuitivas, bonitas y con utilidad, así como un diseño propio al videojuego.
--------------------	--------------------------------------------------------------------------------------------------------------------------------------------

Tipo	No Funcional	Prioridad	Alta
-------------	--------------	------------------	------

Requisito: RNF2 Datos

Descripción	Se debe comprobar la persistencia de los datos mediante pruebas y utilizando patrones de diseños ya establecidos, en mi caso el DTO, MODELS, ENTITIES que aseguran un correcto funcionamiento si se aplica bien.		
Tipo	No Funcional	Prioridad	Alta

Requisito: RNF3 Dificultad

Descripción	En este caso el juego debe de tener una dificultad escalable, de tal manera que permita ir aprendiendo al usuario mientras se avanza, permitiendo mejora y no frustando la experiencia de usuario.		
Tipo	No Funcional	Prioridad	Media

Requisito: RNF4 Compatibilidad

Descripción	El juego debe de estar realizado con la última versión de Unity para que no haya problemas de compatibilidad con plataformas actuales.		
Tipo	No Funcional	Prioridad	Media

Requisito: RNF4 Compatibilidad Móvil

Descripción	Se debe de instalar el módulo de Android para que pueda ser ejecutado en móviles.		
Tipo	No Funcional	Prioridad	Media

Requisito: RNF5 Responsive

Descripción	Las UI deben de adaptarse en función de las pantallas.		
Tipo	No Funcional	Prioridad	Media

Requisito: RNF6 Diseño de Mapas

Descripción	El diseño de los distintos mapas debe de tener cierta coherencia y evolución para no entorpecer la experiencia de usuario.		
Tipo	No Funcional	Prioridad	Alta

Requisito: RNF7 BackUps

Descripción	El juego deberá de tener copias de seguridad en Git y deberá de estar subido en drive para facilitar su descarga.		
Tipo	No Funcional	Prioridad	Media

Requisito: RNF8 Control de Bugs

Descripción	Se deberá de Fixear los distintos bugs que puedan surgir y realizar distintas pruebas exponiendo al personaje a distintas condiciones.		
Tipo	No Funcional	Prioridad	Baja

Requisito: RNF9 Rendimiento

Descripción	El juego debe de tener un rendimiento estable, que permita disfrutar de la experiencia al jugador.		
Tipo	No Funcional	Prioridad	Alta

Requisito: RNF10 Seguridad

Descripción	El juego no debe de contener ningún tipo de código malicioso ni inyecciones de datos.
--------------------	---------------------------------------------------------------------------------------

Tipo	No Funcional	Prioridad	Alta
-------------	--------------	------------------	------

Requisito: RNF11 Estilo

Descripción	Se deberá determinar un estilo visual de juego y seguir ese patrón, de esta manera mejorará la inmersión del jugador.		
Tipo	No Funcional	Prioridad	Alta

Requisito: RNF11 Animaciones, Sonido

Descripción	Las animaciones deben de tener sentido y coherencia con la situación en la que se invocan, así como la música ambiente y efectos de sonido.		
Tipo	No Funcional	Prioridad	Media

Requisito: RNF11 Mecánicas de Juego

Descripción	El movimiento debe de ser fluido y las mecánicas de los bosses fluidas para no interferir en la experiencia.		
Tipo	No Funcional	Prioridad	Media

3.2. Tareas

Descripción	Duración Horas	Dificultad
<u>Unity</u>	249.5	Elevada
IDE UNITY	40.5	Media
Interconexión del diseño de los niveles	1	Baja
Escoger assets adecuados	2	Media
Crear estilo visual de las escenas	8	Elevada
Colisiones de las escenas	4	Media
Implementar Objetos en las escenas (Héroe, Enemigos, Coleccionables, Boss, Trampas...)	7	Media
Animaciones de los elementos	3	Baja
Establecer lógica del fondo	0.5	Media
Pantalla UI Save Game (Botones, Diseño, Paneles, Responsive...)	3	Media
Pantalla UI Tittle Screen	3	Media/Alta
Pantalla UI Game Over	3	Baja
UI In Game Parámetros del Jugador (Monedas, Corazones, Magias, Imagen)	6	Elevada
PROGRAMACIÓN UNITY	209	Elevada
Programar Movimiento Lateral Héroe	3	Baja
Programar Giro hacia la dirección que apunta Héroe	2	Baja
Programar Salto Héroe	1	Baja
Programar Colisionadores Escenario del Héroe	1	Baja
Programar Colisionadores Suelo del Héroe	2	Media
Programar Área Daño Héroe	1	Baja
Programar Doble Salto Héroe	4	Media
Programar Autómata animaciones Héroe	3	Media
Programar Parámetros Héroe	15	Alta
Programar Efectos Sonido Héroe	1	Baja
Programar Arma Héroe (Hacer daño)	10	Alta
Programar Muerte Héroe	3	Media
Programar IA Esqueleto	3	Media

Programar Animaciones Sonido Esqueleto	1	Baja
Programar IA Fantasma	10	Elevada
Programar Animaciones Sonido Fantasma	1	Baja
Programar IA Bestia	5	Elevada
Programar Animaciones Sonido Bestia	1	Elevada
Programar Plataformas Móviles	5	Elevada
Programar Suelo Dañino	5	Elevada
Programar Coleccionables	5	Elevada
Programar Antorchas (Sueltan Coleccionables al romperse)	7	Elevada
Programar Portales (Para trasladar entre escenas)	3	Elevada
Programar Persistencia de algunos objetos	8	Elevada
Programar Boss Final	25	Elevadísima
Programar Sala y Ambiente Boss Final	5	Elevada
Programar Models para la persistencia de los datos	5	Elevada
Programar Entities para la persistencia de los datos	5	Elevada
Programar Clase Controladora Animaciones	3	Elevada
Programar Clase Controladora Cámara	3	Elevada
Programar Clase Controladora GameManager	8	Elevada
Programar Clase Destrucción Objetos	2	Media
Programar Clase Controladora Audio	2	Elevada
Programar Clase Controladora Niveles	4	Elevada
Programar Físicas	6	Elevada
Programar Magias	6	Elevada
Programar Clase Controladora Save para persistir datos y hacer uso de los controladores de los entities y models guardando así la información en archivos locales serializándolos.	10	Elevada
Programar Clase Controladora de Escenas	4	Elevada
Programar lógica de las UI	8	Elevada
Programar Efectos Visuales	6	Elevada
Programar Clase WayPoints, para que los enemigos u objetos sigan una ruta.	3	Media
Programar Clase Controladora de Colisiones	4	Elevada

3.3. Metodología a seguir para la realización del proyecto

Las metodologías son un patrón de trabajo para distribuir el tiempo y enfocar el esfuerzo de manera eficiente en el tiempo.

Para este proyecto he elegido una metodología Ágil, en concreto KanBan. Realmente me gusta más trabajar con Scrum, pero en el caso de Scrum esta más enfocado al trabajo en equipo y las entregas al cliente, como no voy a tener ninguna de ellas, me he decantado por otra de las más importantes y efectivas, KanBan.

KanBan ayuda al freelancer a conocer su capacidad de desarrollo, medir mejor el esfuerzo para realizar una tarea y los tiempos de previsión de entregas gracias a las herramientas que aporta.

Kanban es un método visual para gestionar el trabajo gracias al feedback visual que aporta, maximizando así la eficiencia.

Gracias a que es visual se puede evitar la acumulación de trabajo pendiente, mejorando el flujo de trabajo. Funciona mediante tarjetas gracias a las cuales se puede ver de forma visual que queda pendiente y que no,

que tienes que hacer hoy o en otras fechas, aportando feedback instantáneo.

The screenshot shows a task management interface with the following columns:

- Tareas nuevas**: Contains tasks for today: "Logica del Heroe" (Mañana) and "Intergaz Grafica Game Over" (Viernes).
- Para hacer hoy**: Contains tasks for today: "Programar Colisiones" (Hoy) and "Clase Controladora Save".
- Para hacer más tarde**: Contains tasks for later: "Diseño Del TimeLap" and "WayPoints".
- Pendientes**: Contains tasks pending: "Establecer Limites del Mapa".
- Terminados**: Contains completed tasks: "Fantasma Editar" (28 abr) and "Esqueleto Logica" (29 abr).

Each task card includes a checkbox, a title, a due date, and a small icon. Buttons for adding new tasks are located at the bottom of each column.

Tambien dispone de un calendario con las cosas que has ido haciendo según la fecha, aparte del menu principal donde tienes lo actual y los parametros que tu quieras ponerle:

Calendario		Archivos					Elegir plan
Hoy	< >	Abril 2022					
		MAR	MIÉ	JUE	VIE	SÁB	D
ea	26	27	28	29	30	1	
	Establecer Plataformas Fase Presentacion Boss Control de animaciones del fantasma + Agregar tarea	Controles Heroe Que la camara siga al heroe Fisicas del level 1_1 Portal del level check + Agregar tarea	Fantasma Editar + Agregar tarea	Esqueleto Logica + Agregar tarea			

Estoy utilizando las herramientas de asana, un software de tipo Kanban con tarjetas y considero que es una herramienta muy importante el progreso del proyecto, asi como de facil acceso e intuitiva.

3.4. Planificación temporal de tareas

Para ello he realizado un diagrama de gantt, marcando las tareas con su estado en una linea temporal. Adjunto el diagrama entero en la entrega del proyecto, aquí solo lo que cabe en pantalla.

3.5. Presupuesto (gastos, ingresos, beneficios)

Randomly

sanjuanpeña181
50015 zgz
...
España
158918N
Tel: 637285192
victor_pena_wow@hotmail.com

Presupuesto n° P-2022-0001

A fecha de 4/4/22
Válido por 3 meses

Randomly

Nº	Descripción	Cant	PU sin IVA	IVA	PVP sin IVA
1	Agua	1 u	50,00 €	21 %	50,00 €
2	Desarrollo	250 hr	20,00 €	21 %	5.000,00 €
3	Gas	1 u	60,00 €	21 %	60,00 €
4	Electricidad	1 u	50,00 €	21 %	50,00 €
5	Cafe en Horario	50 u	0,70 €	21 %	35,00 €
6	Amortización Licencia Unity	1 u	150,00 €	21 %	150,00 €
7	Amortización Pc	1 u	40,00 €	21 %	40,00 €

Subtotal sin IVA	5.385,00 €
Precio Amigo	- 538,50 €
Retraso en la Entrega	- 538,50 €
Total sin IVA	4.308,00 €
IVA al 21 %	904,68 €
Total con IVA	5.212,68 €

Pago en efectivo o con cheque bancario.

Este documento es un ejercicio de aproximación al valor real de la obra a ejecutar, por lo tanto es susceptible a modificaciones posteriores en función de los materiales y de sus precios.

Se exigirá un anticipo del 30% antes de empezar la obra, el resto se abonará una vez finalizados los trabajos.

Cliente
Leído y aprobado


Víctor Pena


address — 50015 zgz, ..., España
email: victorpenawow@gmail.com

3.6. Análisis de riesgos

Pese a no ser una actividad laboral que conlleve excesivos riesgos laborales, por no ser un trabajo dinámico ni con especial trabajo físico, no está exento de riesgos, peligros ni patologías médicas.

Principales riesgos laborales

- Fatiga Visual.
- Fatiga Muscular.

- Golpes o caídas.
- Contacto Eléctrico.
- Carga Mental.
- Distintos factores ambientales.



Fatiga Visual

Se conoce como síndrome visual informático “SVI” la cual es una afección temporal resultante de enfocar los ojos en una pantalla de ordenador durante largos e ininterrumpidos periodos de tiempo, creando una tensión muscular y nerviosa por un esfuerzo acomodativo excesivo.



Fatiga Muscular

Es producida por adoptar posturas incorrectas durante un periodo de tiempo prolongado que suele ir acompañado de una disposición inadecuada del equipo informático.



Golpes y Caídas

Se sucede generalmente como consecuencia del manejo manual y cargas o desplazamientos bajo tensión emocional y falta de atención.



Contacto eléctrico

Como consecuencia de choque o contacto directo con elementos con tensión o con masas puestas accidentalmente en tensión eléctrica. Por quemaduras, por arco eléctrico o caídas como consecuencia de dicho choque. Incendios o explosiones por sobrecarga de tensión.



Carga Mental

Esta parece principalmente por efectuar un esfuerzo intelectual durante periodos de tiempo prolongados o al límite de sus capacidades o conocimientos.

Actualmente es una de las principales patologías que afectan a los informáticos.

3.7. Contrato/Pliego de condiciones

CONTRATO DE DESARROLLADOR

En Zaragoza, a 04 de abril de 2022

REUNIDOS

Don Vitty, mayor de edad, con DNI 73158918N y domicilio en Zaragoza, actuando en nombre y representación de Randomly inscrita en el Registro Mercantil de ningún lugar con domicilio social en el Valhala, actuando en su calidad de dios en posesión de poderes suficientes para este acto. (El Desarrollador)

Don te están timando, mayor de edad?;, con DNI 16963886Y y domicilio en la Jota, actuando en nombre y representación de Los Manolos inscrita en el Registro Mercantil de toda la vida, en calidad de timado, en posesión de poderes suficientes para este acto. (El cliente)

MANIFIESTAN

Que las partes están interesadas en formalizar el presente contrato; que poseen suficientes poderes para la firma del mismo; que se reconocen capacidad legal necesaria para poder llevar a cabo la celebración y declaran expresamente que actúan de forma libre, voluntaria y no viciada.

EXPONEN

El cliente está interesado en que el prestador lleve a cabo el diseño y desarrollo de una aplicación conforme a las necesidades específicas indicadas en el Anexo I.

Que el prestador tiene como objeto el diseño y desarrollo de aplicaciones y cuenta con el personal necesario para ello; que las características de las aplicaciones son las indicadas en el Anexo I por el cliente; que ambas partes aceptan cumplir con sus respectivas obligaciones.

En relación a lo anteriormente expuesto, las partes otorgan el presente contrato que se regirá por las siguientes CLAUSULAS

I. OBJETO DEL C

ONTRATO

El presente contrato regula la prestación de servicio de diseño y programación de la aplicación solicitada por el cliente. La aplicación se acogerá en todo caso a las categorías, diseño y contenidos indicados en el Anexo I.

II. PROPIEDAD INTELECTUAL

El prestador garantiza al cliente que todo el material utilizado para el desarrollo del proyecto, así como el resultado obtenido, es un producto original que no vulnera ninguna ley o derechos de terceros, en especial los referidos a propiedad industrial e intelectual.

El prestador reconoce los derechos de explotación sobre la obra. El prestador renuncia de forma indefinida a ejercitar cualquier tipo de acción para recuperar sus derechos de propiedad intelectual respecto al desarrollo, a excepción del derecho de autoría por el que el desarrollador tiene derecho a exigir que aparezca su nombre en la aplicación.

En caso de ser el cliente el encargado de proporcionar los contenidos (gráficos, textos, vídeos, categorías...), éste se hace responsable de cualquier tipo de reclamación de terceros en relación a la titularidad de dichos contenidos, eximiendo de toda responsabilidad al prestador.

III. OBLIGACIONES DEL PRESTADOR

El prestador se compromete a desarrollar el presente proyecto bajo las directrices del cliente, ajustándose a los términos indicados por éste en el Anexo I y conforme a las mejores prácticas existentes en el mercado, así como con la máxima diligencia posible.

Una vez aceptadas las características de la aplicación pueden producirse variaciones en el diseño y/o contenido del mismo a petición del cliente. Salvo que conllevaran una variación sustancial del Proyecto Inicial no supondrá aumento del precio, considerando variación sustancial toda aquello que suponga un aumento del tiempo total de trabajo estimado inicialmente superior al ____%.

El prestador se compromete a finalizar el desarrollo en el plazo acordado siempre que la otra parte haya colaborado activamente en el desarrollo del mismo (aceptando los diseños, entregando los contenidos, etc)

IV. OBLIGACIONES DEL CLIENTE

El cliente se obliga a realizar el pago del precio en los términos indicados en la cláusula VI del presente contrato.

El cliente se obliga a mantener un contacto constante con el prestador entregando en tiempo y forma los contenidos del proyecto de la aplicación (Textos, imágenes, videos, categorías...), la aceptación del diseño y cualquier otra necesidad que requiera el prestador para poder finalizar el proyecto. En cualquier caso se atenderá a lo dispuesto respecto a propiedad intelectual en la cláusula II.

En caso de depender la entrega de contenidos de un tercero seleccionado por el cliente, éste deberá indicarlo en el apartado comunicaciones y comprometiéndose a responder de los posibles retrasos que pudieran darse.

V. COMUNICACIONES

Las partes se obligan a comunicarse toda la información que pudiera ser necesaria para el correcto desarrollo del proyecto. Toda comunicación entre las partes relativa al presente contrato se realizará por escrito o telefónicamente. A efectos de comunicaciones y/o notificaciones las partes designan:

Prestador

Domicilio en Zaragoza con número de Fax 73158918, correo electrónico vitty@hotmail.com y Teléfono 666666666

Cliente

Domicilio en Utebo, con número de Fax 231457912 correo electrónico manolito@hotmail.com y Teléfono 641234578

Cualquier cambio de domicilio o dirección de contacto deberá ser comunicado a la otra parte por escrito con una antelación mínima de 30 segundos hábiles.

VI. DURACIÓN Y PRECIO

El presente contrato entrará en vigor el día 04 de abril de 2022 y tendrá una duración de 90/3días/meses, siendo posible, a petición del prestador, añadir 0 días adicionales para llevar a cabo la entrega del proyecto, sin ello suponer repercusión económica alguna.

El precio a abonar por parte del cliente como pago por la prestación del servicio prestado equivale a 5000 € (5mil euros), añadiendo a dicha cantidad el IVA correspondiente (16%)

Dicho precio será abonado de la siguiente forma:

a) 60 € (euros), que corresponden al 1 % del precio, serán abonados en el momento de la firma del presente contrato.

b) El 10 % restante, equivalente a 15 € (1.5), serán abonados en el momento de finalización del proyecto.

El pago del 100 % final será llevado a cabo siempre que el prestador de por finalizado el proyecto y en cualquier caso cuando se cumpla el plazo de entrega y el cliente no haya entregado o colaborado activamente en la entrega de contenidos, validación de colores y diseño o entrega de categorías de la aplicación.

El pago será realizado mediante transferencia Bancaria en el número de cuenta XXXXXXXXXXXXXXXX

VIII. CONFIDENCIALIDAD Y PROTECCIÓN DE DATOS

Las partes contratantes declaran conocer y cumplir la legislación Europea y Española sobre protección de datos, comprometiéndose a tratar los datos personales obtenidos durante el desarrollo del proyecto de acuerdo con dicha normativa.

Se informa al cliente que sus datos quedarán almacenados en un fichero, automatizado o no, con las únicas finalidades de informarle sobre novedades y nuevos proyectos en los que se encuentre trabajando la empresa prestadora, así como para el mantenimiento de la relación contractual.

Se informa al afectado que puede ejercer los derechos de acceso, rectificación, cancelación y oposición de sus datos de carácter personal solicitándolo por escrito y acompañando una fotocopia del DNI en la dirección del prestador indicada en el presente contrato.

Toda la información relativa al proyecto, así como a los datos de carácter personal, tendrá el carácter de confidencial por lo que las partes deberán guardar el mayor secreto respecto a las mismas, garantizando en todo caso el no acceso por parte de terceros a dicha información.

XI. NO COMPETENCIA

El cliente se compromete a no realizar proyectos de equivalente o análoga naturaleza para un tercero, ni iniciar a partir del presente desarrollo una nueva línea de negocio relacionada con la realización de aplicaciones. Se considerará que el cliente incurre en dicha circunstancia siempre que opere directamente o indirectamente mediante otra empresa en la que disponga de participación social, o que actúe como mero asesor o colaborador y que en definitiva obtenga como resultado un proyecto igual, semejante o con la misma finalidad a la ofrecida por el prestador.

El incumplimiento del anterior compromiso llevará aparejada una penalización equivalente a 500 €, sin perjuicio y de las indemnizaciones que correspondiesen por los daños y perjuicios causados al prestador.

XI. EXTINCIÓN

Además de por las causas generales del Derecho, este contrato se extinguirá:

- a. Por el transcurso del mismo.
- b. Por ser declarados en situación de suspensión de pagos, quiebra o concurso de acreedores cualquiera de las partes.
- c. Por incumplimiento de las obligaciones estipuladas en el presente contrato.

Las partes aceptan expresamente el sometimiento a las penalizaciones económicas indicadas a continuación

siempre que se rescinda el contrato de forma previa a la finalización del proyecto:

XII JURISDICCIÓN Y LEGISLACIÓN APLICABLE

Todas las cuestiones litigiosas sobre el presente contrato mercantil, quedarán regidas por la legislación española, específicamente por lo dispuesto en el Código Mercantil, y en su defecto, por las disposiciones españolas del Código de Comercio, Leyes Especiales, usos mercantiles y con carácter supletorio, por el Código Civil.

En cualquier caso, será obligatorio que en caso de conflicto las partes intenten previamente resolver la cuestión de mutuo acuerdo, sometiéndose en su caso a los Juzgados y Tribunales de Justicia que por orden correspondan.

ANEXO I: PROYECTO TÉCNICO

(Incluir el desarrollo del proyecto, así como la planificación en el tiempo del mismo, debiendo quedar expresamente fijada la fecha de finalización del proyecto)

- Características del Diseño.
- Aplicaciones de referencia.
- Categorías.
- Colores Corporativos.
- Imágenes y logotipos. (Quién los entrega)
- Formato de entrega de los contenidos.
- ...

4. Documento de Análisis y Diseño

4.1. Modelado de Datos. Análisis y diseño de la base de datos

En mi caso el apartado de Unity no debería de tener diagrama E/R ya que no he trabajado con una base de datos al uso, pero si que persisto los datos y trabajo con entities y models, así que voy a adaptar a lo que seria el diagrama de E/R de la forma que estoy persistiendo los datos, por eso puede resultar algo extraño el diagrama.

Vale aclarado eso, el apartado de Unity no necesitaba de una base de datos ya que mi juego es en local y seria consumir recursos extra, repercutiendo en el rendimiento del juego, así que me decante por implementar un MVC utilizando Entity Framework de visual studio para el acceso a datos.

La idea es similar a una base de datos, pero en vez de llenar la base de datos, rellena los entities con los modelos que le pasemos, creando tablas en las propias clases de C#.

Entity Framework en realidad es un modelo relacional de objetos (O/RM) que permite serializar datos.

Para serializar los datos debo crear entities con los datos que corresponderían a una tabla en una BD, por ejemplo, para la tabla HeroData creo una clase serializable con los datos a persistir, como en el ejemplo siguiente:

```
[Serializable] //Hay que hacer la clase serializable para poder usar el formato JSON o RawData.  
public class HeroData  
{  
    public PowerUpId currentPowerUpId; //Guardo el tipo de magia actual.  
    public int powerUpAmount; //Guardo la cantidad de magia actual.  
    public int coins; //Guardo las monedas.  
    public int heart; // Guardo los corazones  
}
```

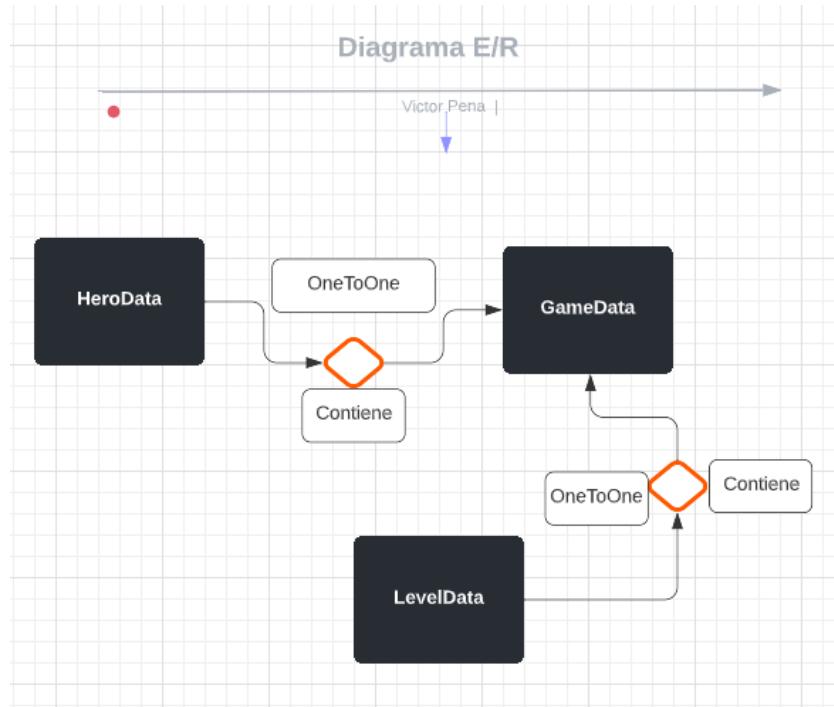
En estas clases entities se especifican lo que serian las relaciones entre las tablas. Por ejemplo, mi entity GameData contendrá la conexión de tablas de HeroData y la propia GameData, simulando lo que sería en base de datos.

Dicho esto, estos entities necesitan de clases modelos que serialicen los datos de estos entities en tablas, pudiendo ser gestionados por el Entity Framework de C#.

La idea final es trabajar con objetos en vez de con tablas y serializar estos datos en ficheros en local, que luego serán accedidos en vez de una base de datos.

4.1.1. Diagrama E/R

En Unity con (O/RM) sería algo así la idea así que lo adapto a lo que sería en E/R, pero serializando.

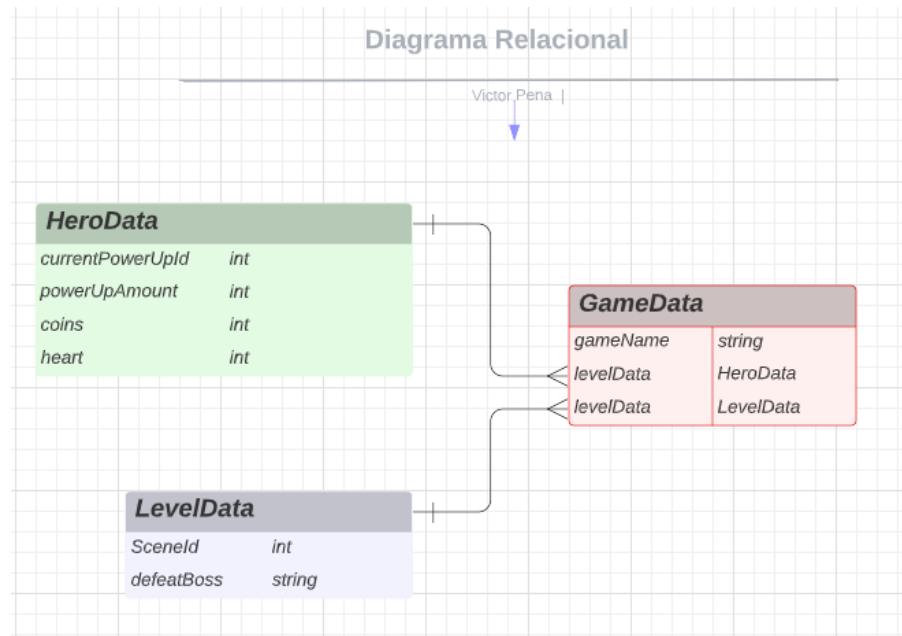


4.1.2. Diagrama Relacional

Lo mismo, adapto lo que sería similar al diagrama relacional en Entity Framework.

Guardo la información del héroe, la de los niveles y utilizo la tabla GameData para guardar ambos. Game data vendría a ser el equivalente del registro de guardado de partida, tendría el parámetro gameName con el nombre de la partida a guardar y las referencias a las otras tablas HeroData y LevelData. Por lo tanto, si quisiésemos guardar partida escribiríamos en un fichero la información de estas tablas.

Nota: a la hora de escribir en un fichero esta información podría escribirla en una base de datos también si replicase las tablas, ya que tengo las tablas montadas, pero por rendimiento y lógica de este proyecto lo he serializado y persistido en local.



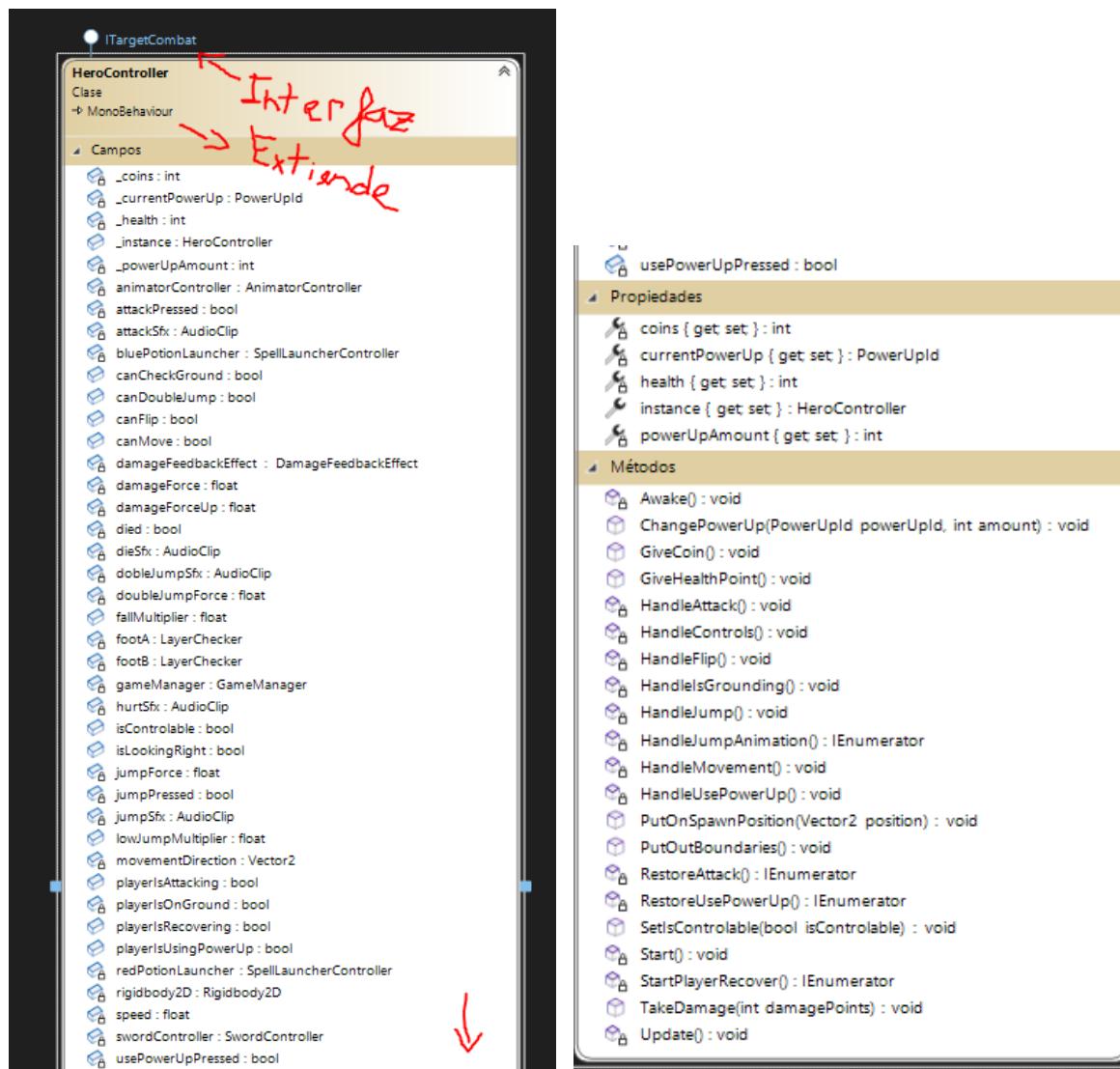
4.2. Análisis y diseño del sistema funcional

4.2.1.1. Diagrama UML

A la hora de mostrar las clases, **por su complejidad** y extensión, **explicare primero todas las clases** expandidas, con sus parámetros y métodos, y **después**, las contraeré y **mostraré las relaciones** entre ellas **con un Diagrama UML**. Para ver y comprender como se relaciona la programación y los scripts en C# con Unity salta a el apartado 4.4.1 Tecnologías/Herramientas usadas y descripción de las mismas y luego retorna a este punto para ver el UML.

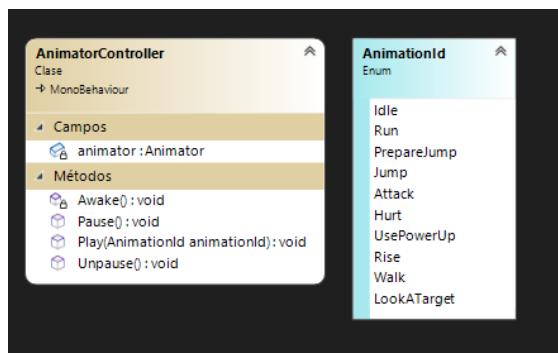
Empezare por la piedra angular del proyecto, la lógica del Héroe, casi todo está enfocado a este script, por lo tanto, es inmenso. Debido a su extensión lo dividiré en varias imágenes:

HeroController.cs



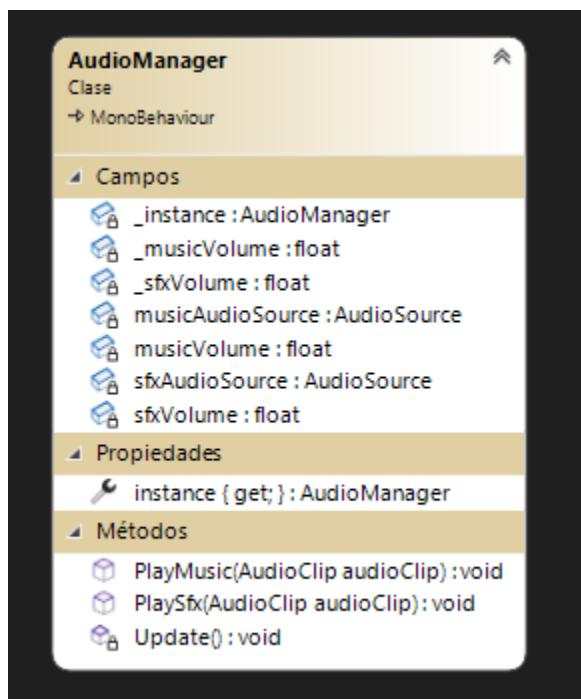
Este script tendra toda la logica relativa al heroe, desde obtener monedas, atacar, habilitar heroe o no, recibir daño, moverse, usar magias, saltar, gestionar su vida... todo.

Clases para las animaciones



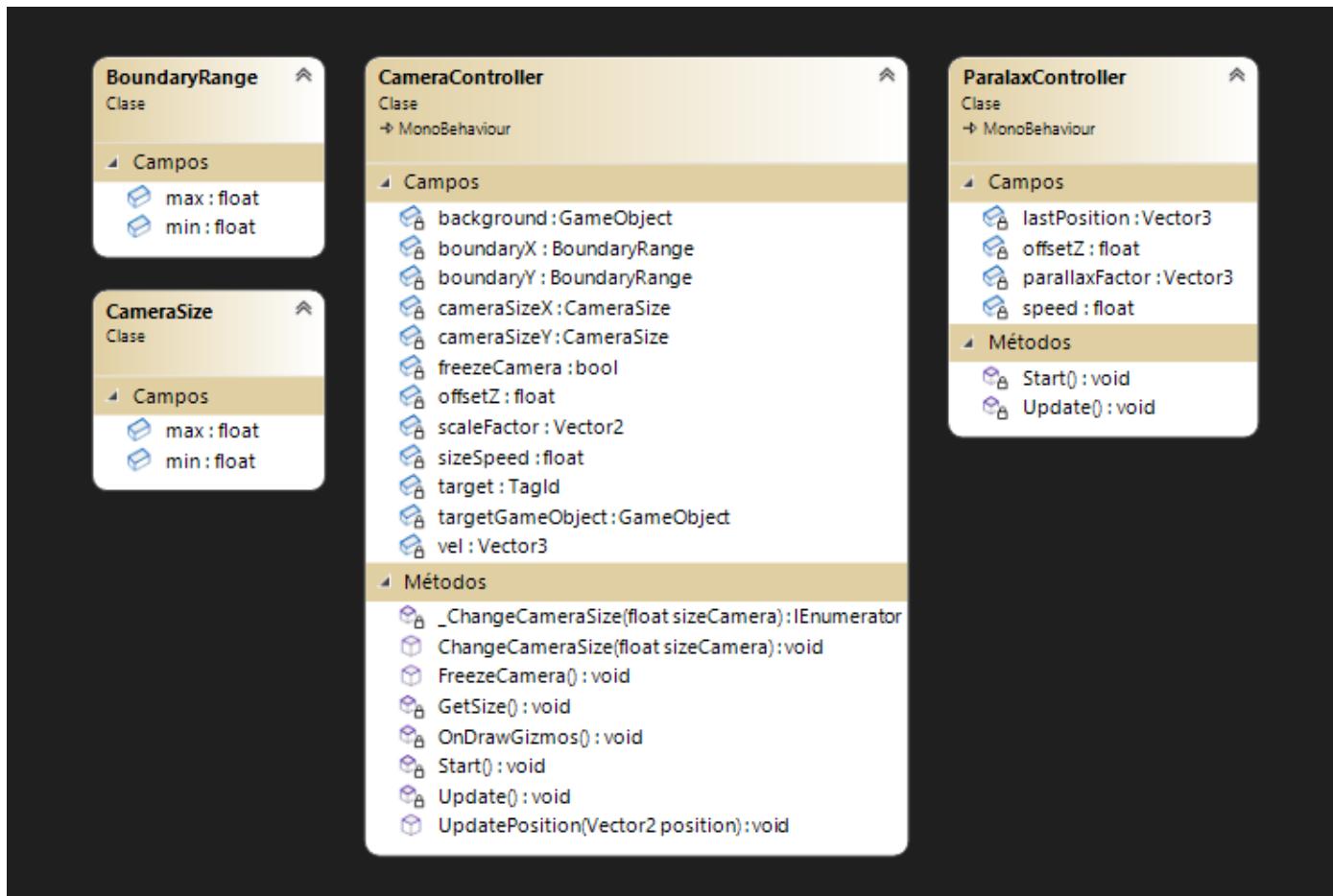
Enum sera una clase de tipo enumerado para elegir las distintas animaciones según el parametro que reciba.

AudioManager.cs



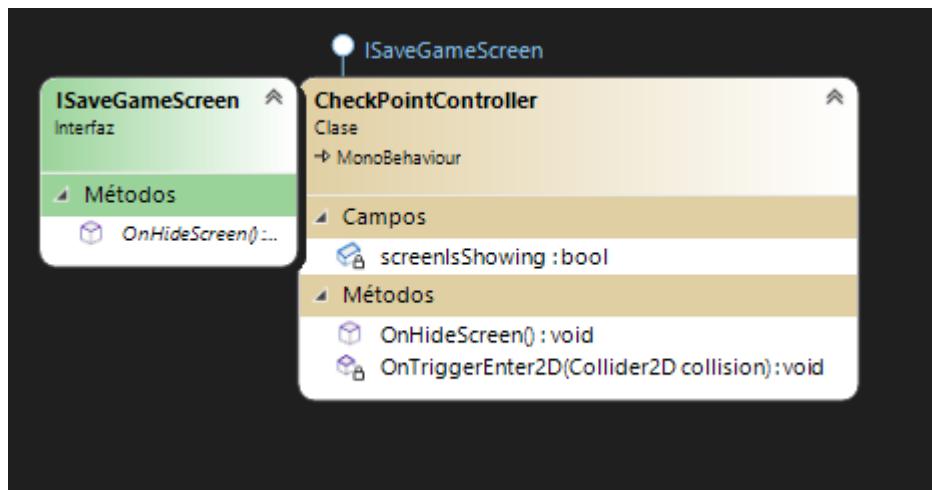
Esta clase se encargara de gestionar la logica del sonido de la aplicación. Es decir las demas clases usaran esta clase por ejemplo para instanciar la musica del nivel, los distintos efectos de sonido, parar la musica... Esta clase al igual que el heroe usara el patron de diseño Singleton, el cual consiste en asegurarse que solo se pueda crear una sola instancia de estas clases que lo implementan. Gracias a ello la informacion de las instancias de estas clases persistira entre escenas y no sera pisada por una nueva instancia.

Clases para el control de la Camara



Parece complicado, pero en realidad solo interesa la clase CameraController, las demás estan para darle apoyo y poder ser consumidas por esta, por ejemplo CameraSize para cambiar el tamaño, BoundaryRange el máximo de la cámara y parallaxController para que la camara se recoloque en la posición que indiquemos por parametro. Este controlador es bastante util para hacer que la camara siga al jugador por la escena.

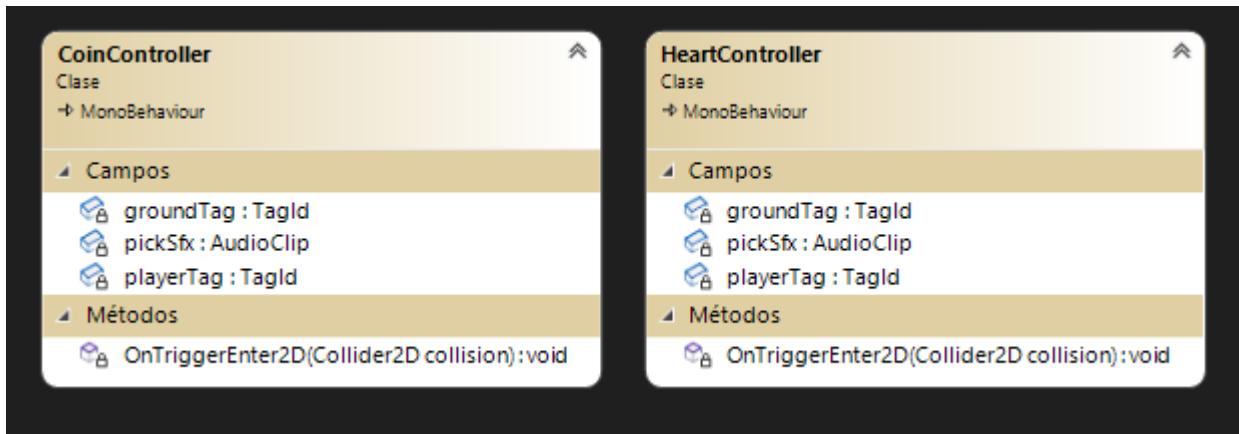
CheckController.cs



Esta clase sera usada en las salas seguras, permitiendo ocultar la interfaz de guardado UI que se invocara al colisionar con el OnTriggerEnter2D que se puede observar en la clase. Aprovecho para decir que los OnTriggerEnter2D son los colisionadores y dentro de estos metodos se referencia al colisionador que se

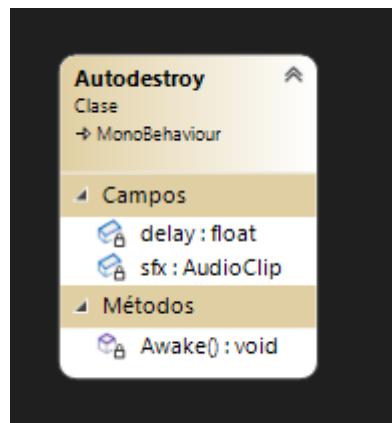
pone dentro del juego y se puede implementar la logica al acercarse a un objeto por ejemplo.

Clase para los coleccionables



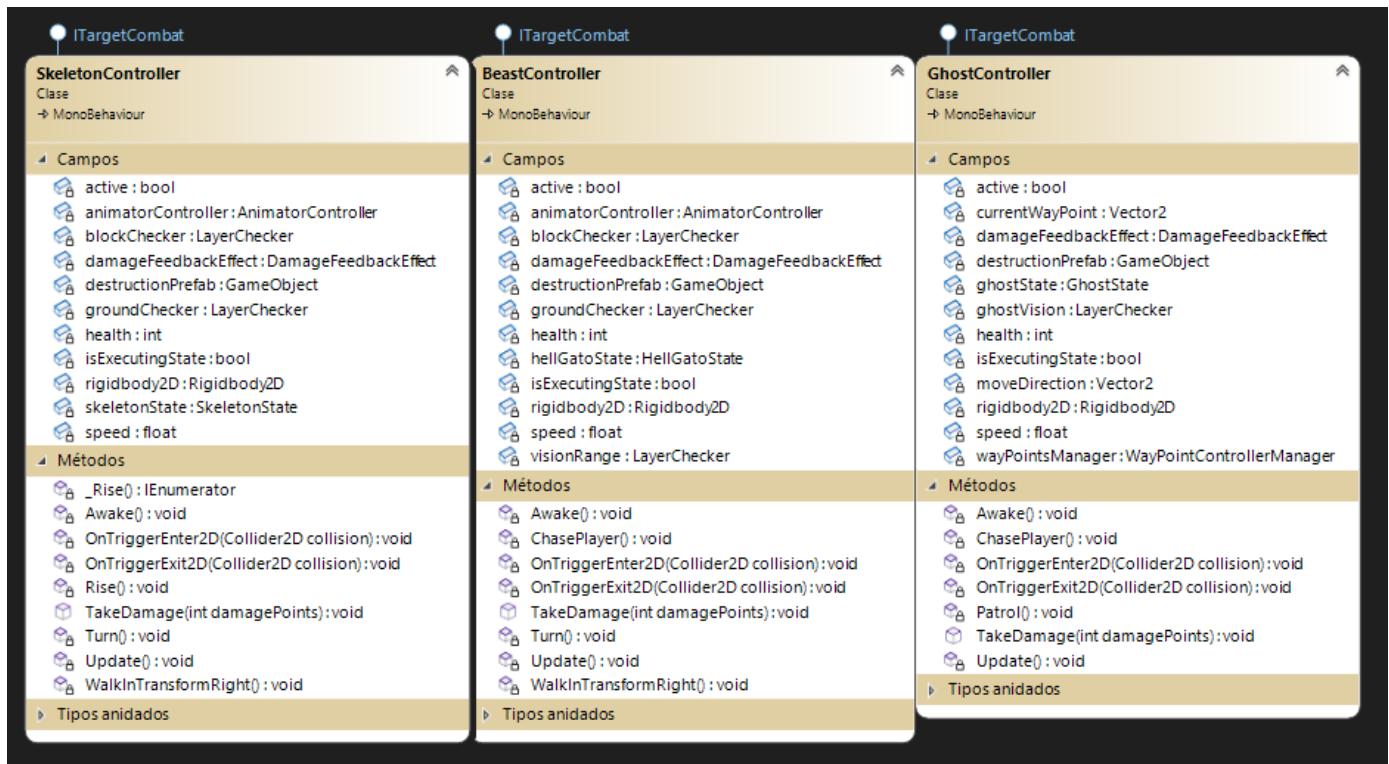
Estas clases pertenecen a los objetos dentro del juego (corazones y monedas) e implementan su logica. Se puede observar tambien que tienen colisionadores para que cuando nos acerquemos podamos sumarle la cantidad de monedas o de corazones al heroe si implementamos la logica dentro de estos metodos.

Autodestroy.cs



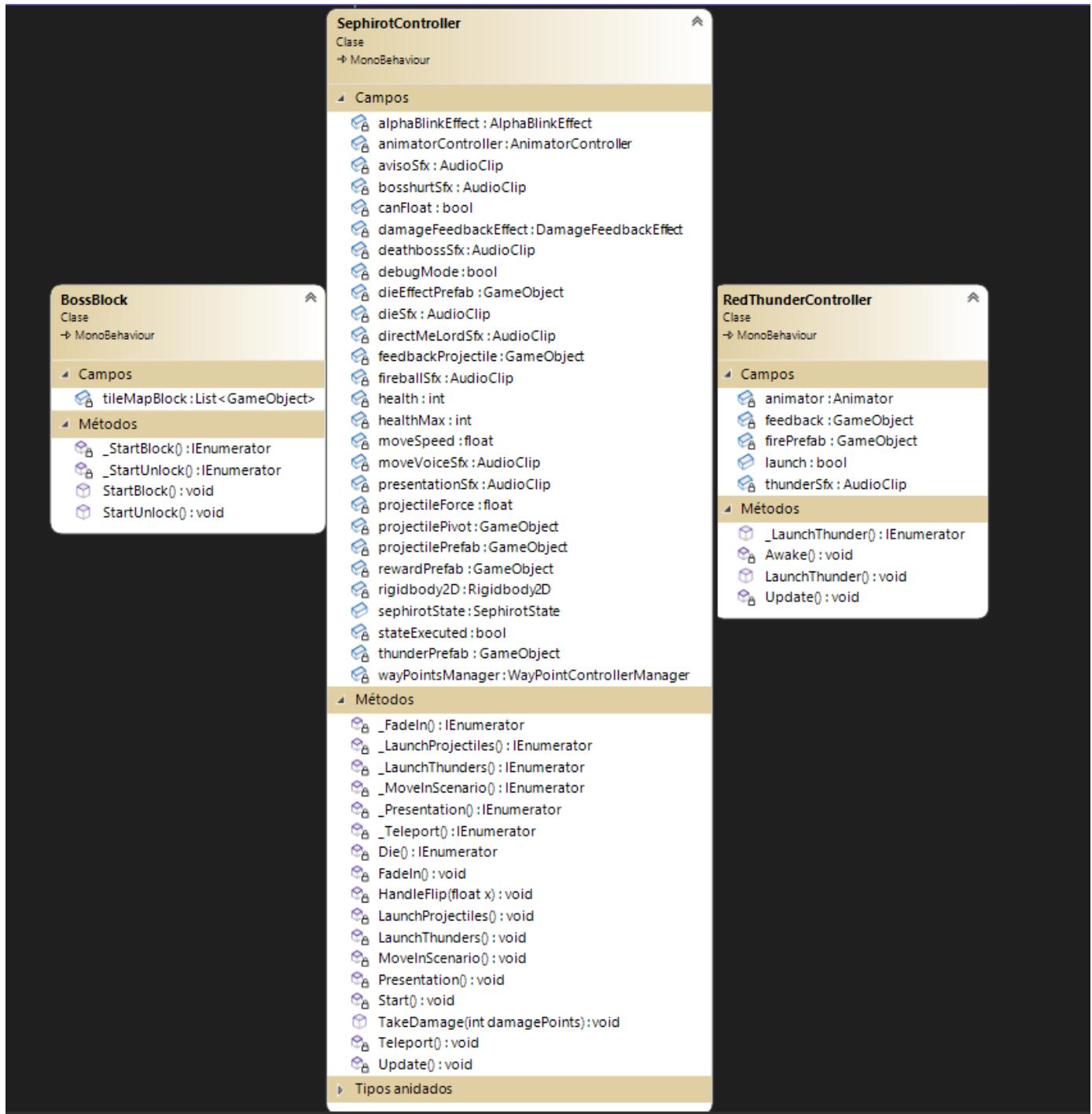
Implementa la logica necesaria para destruir un objeto en el juego si se le invoca.

Clases de los enemigos

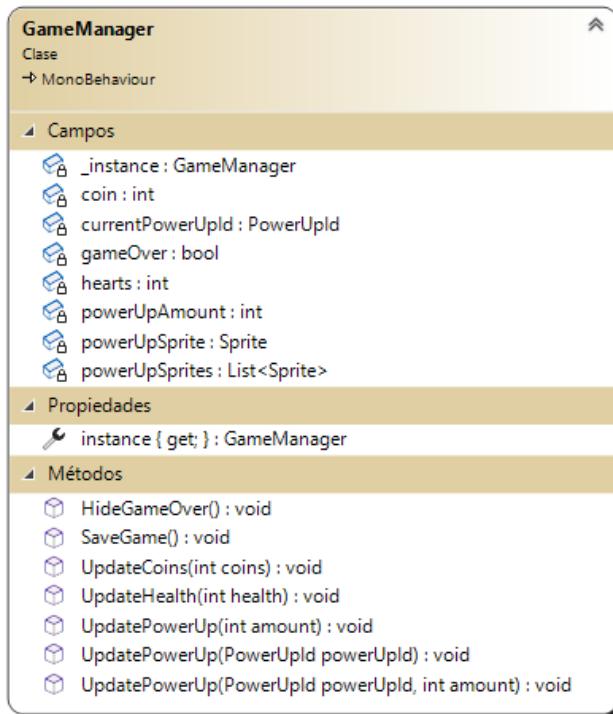


Antes de nada se puede observar que implementan la interfaz `ITakeDamage`, al igual que el heroe, como su nombre indica permite recibir daño. En resumen estas clases contienen la logica del esqueleto, bestia y fantasma, asi como su inteligencia artificial, colisionadores y todo lo necesario.

Clases Boss Final

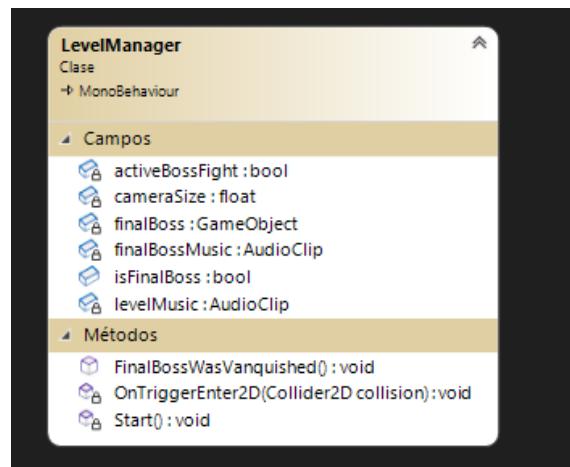


Estas clases contienen la logica necesaria para el boss final, la clase BossBlock bloqueara las entradas del mapa para que no podamos huir mientras el boss viva, alejara la camara para tener una visión más global ya que el boss es grande y cambiara la música. La clase RedThunderController tiene la logica para la magia del boss, por lo que la clase del SephirotController (es el nombre del boss) instanciará estas clases. El combate del boss lo explicare en la presentación a fondo con su ia ya que abarca muchas cosas, pero resumiendo tiene un automata para controlar las distintas fases y generarlas con un patrón aleatorio dependiendo de las distintas condiciones, por lo tanto su comportamiento cambiara en función de las necesidades.



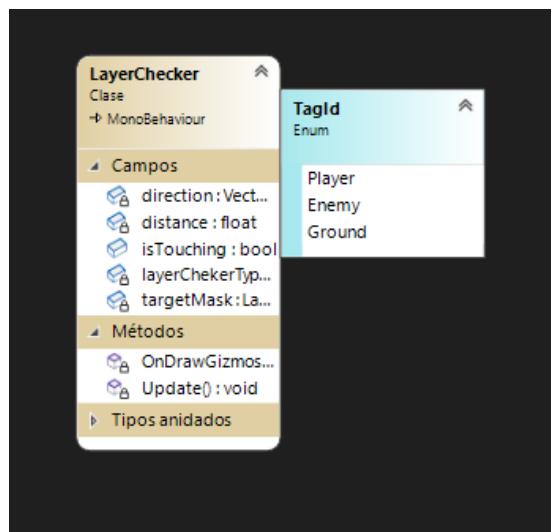
Esta clase se implementara en la escena y se encargara de gestionar las cosas que pasan en la escena, como guardar partida, upgradear los parametros del heroe (vida, monedas...) cuando ocurren x condiciones, instanciar algunos objetos...

LevelManager



LevelManager trabajara en conjunto con GameManager para controlar algunas cosas del nivel, como la musica o parte del comportamiento del boss final, ya que interactua con el nivel.

LayerChecker.cs



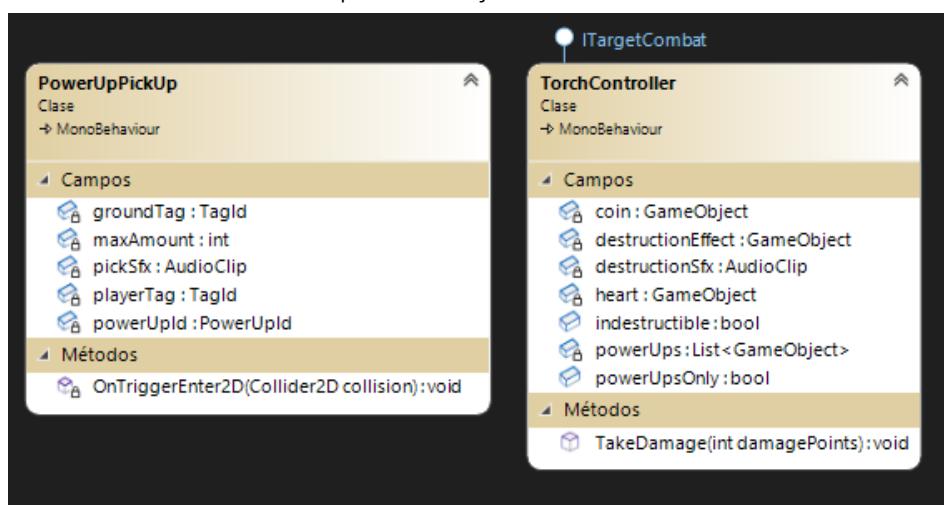
Se encarga de gestionar las colisiones de forma global, es decir las otras clases usan la logica de esta clase para saber si esta tocando algun objeto. La interfaz TagId es para identificar que tipo de objeto se esta tocando.

SwitchParentPlatform



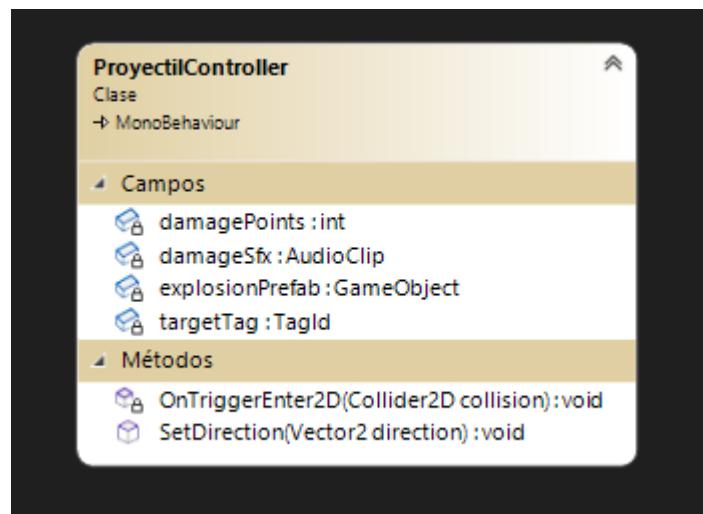
Se encarga de la logica de las plataformas moviles.

Clases para los objetos consumibles



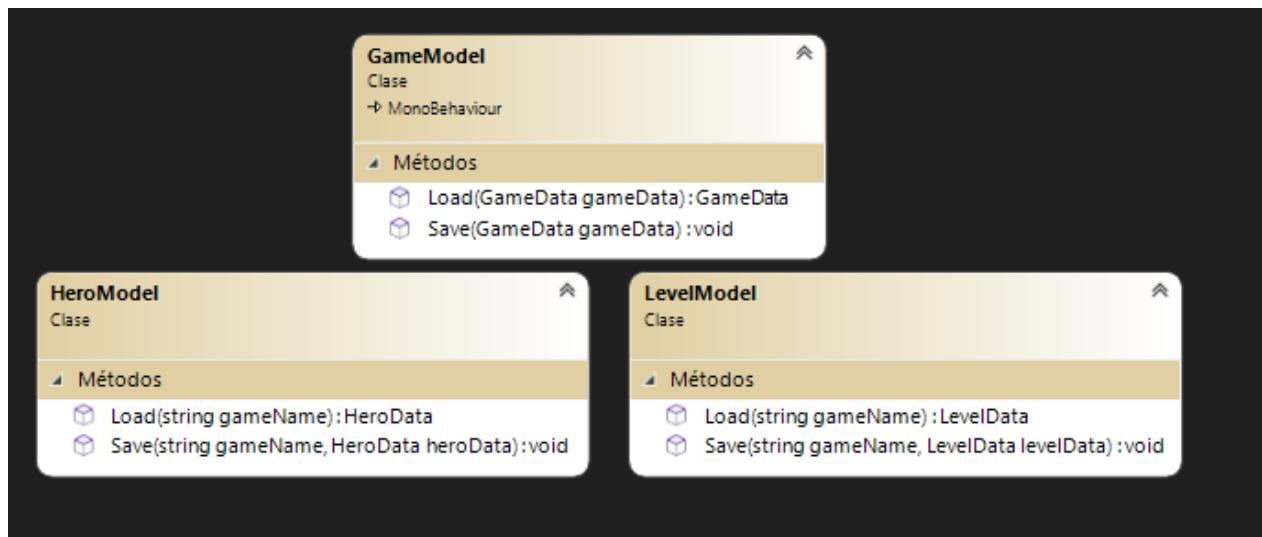
Implementa la logica de los objetos consumibles, PowerUpPickUp son las pociones y Torch las antorchas que pueden soltar pociones, monedas o corazones.

ProyectilController



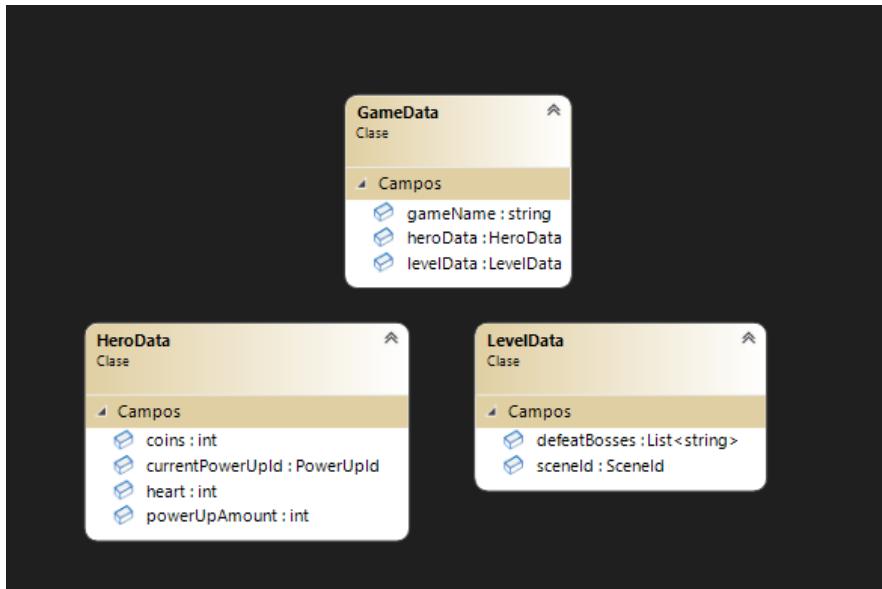
Contiene la logica para que los proyectiles salgan hagan daño y tengan una explosión al impactar.

Models



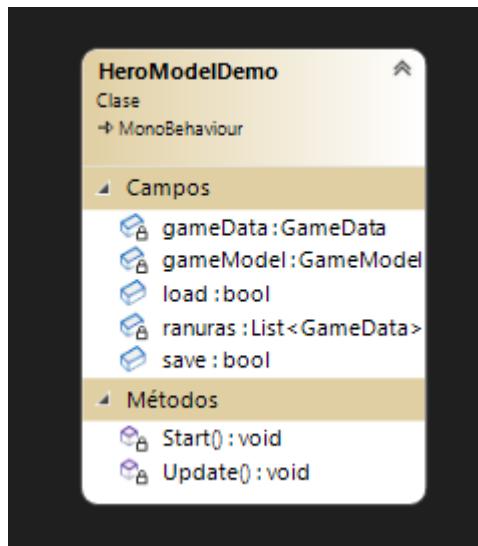
Estas son las clases de los modelos para persistir datos. Usan los entities para serializar los datos. Por lo tanto también debe de haber entities con los parametros necesarios para mapear las tablas.

Entities



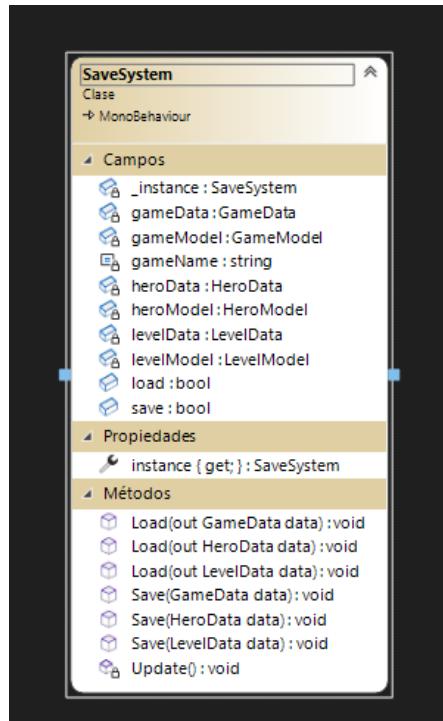
Entities que representan las tablas a serializar, son rellenados en el model y usados por el controlador que veremos a continuación.

HeroModelDemo.cs



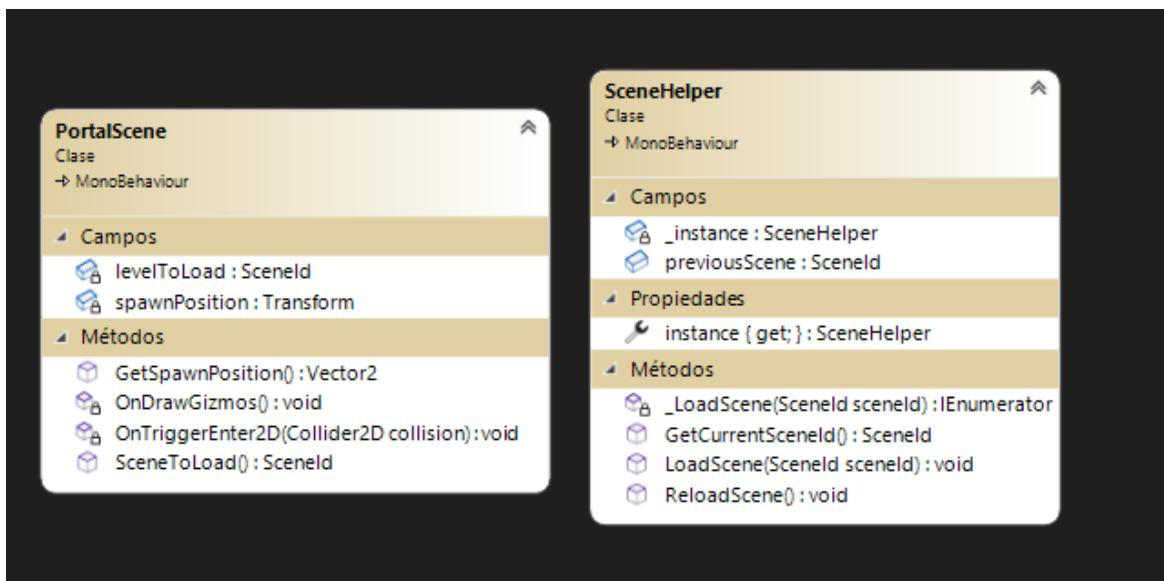
Se encarga de hacer de provider (como puente a otros controladores) para realizar acciones sobre los models y entities.

SaveSystem.cs



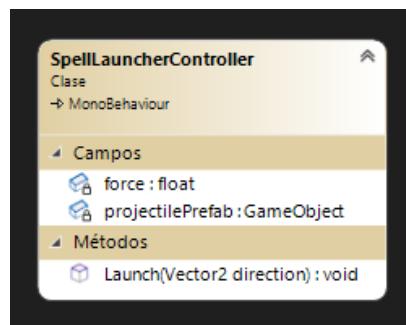
Usado para guardar la partida haciendo uso del HeroModel.

Clases para las escenas



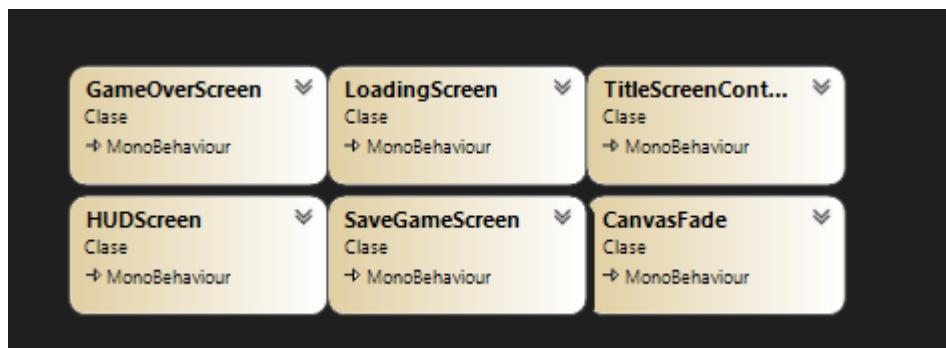
PortalScene se encarga de la logica de teletransportar al jugador de una escena a otra y posicionarlo al llegar a x punto del mapa y la clase SceneHelper se encarga de cargar la escena correspondiente.

LauncherController



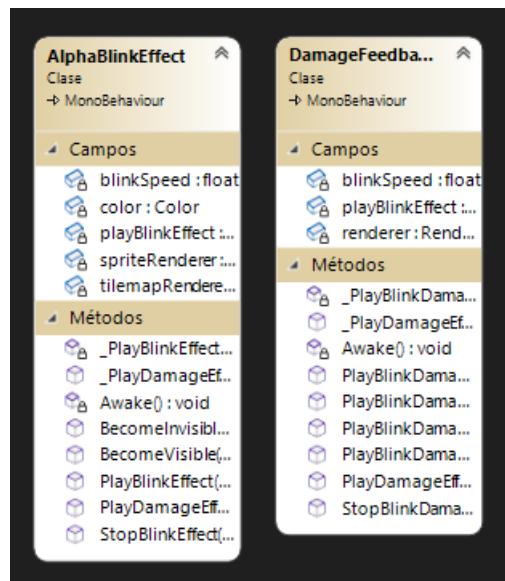
Trabaja en conjunto con ProjectileController, pero esta clase indica la dirección hacia la que saldrá disparado y dispara los proyectiles.

Clases para la UI



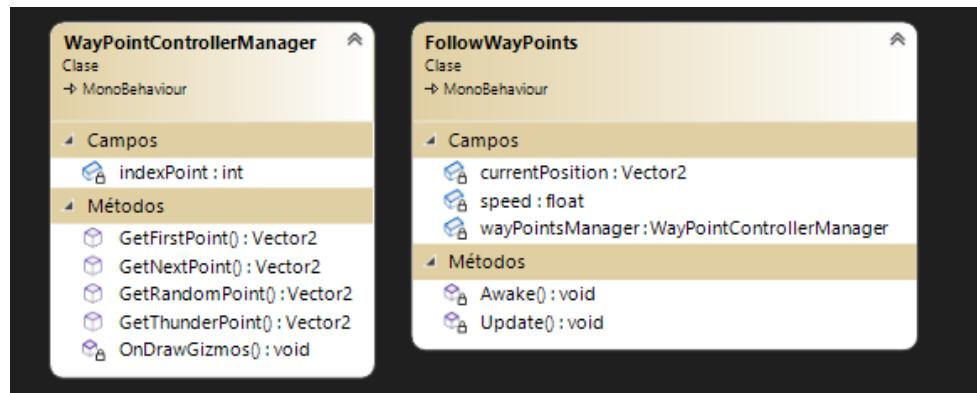
Debido a su extensión, las dejo contraídas, pero contienen canvas y botones. Canvas es para dibujar la interfaz con paneles, fondos, imágenes... y los botones para interactuar con ellas. Cada script representa a una pantalla de la UI, a tener en cuenta HUDScreen, que es la interfaz dentro del juego que se ve constante con la vida del jugador, monedas. CanvasFade es un controlador para dar un efecto visual de fade.

Clases de efectos Visuales



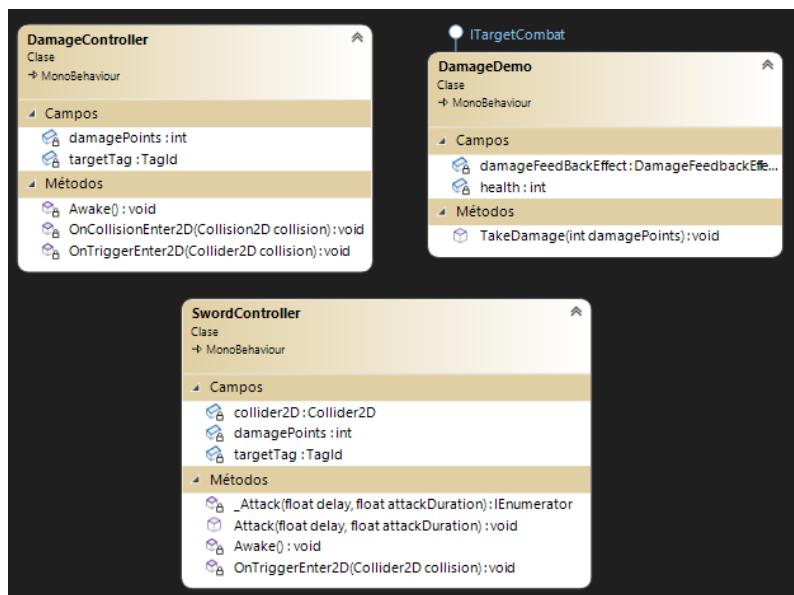
DamageFeedback.cs instancia efectos visuales para recibir daño y usa AlphaBlinkEffect.cs para aplicar un efecto de parpadeo a la imagen.

WayPointsManager



Utilizo estas clases para la IA de los objetos y los enemigos, ya que proporcionan puntos invisibles que han de seguir, por lo tanto marcan la ruta del enemigo u objeto por el mapa, aportandole patrones de movimientos.

SwordController y DamageController



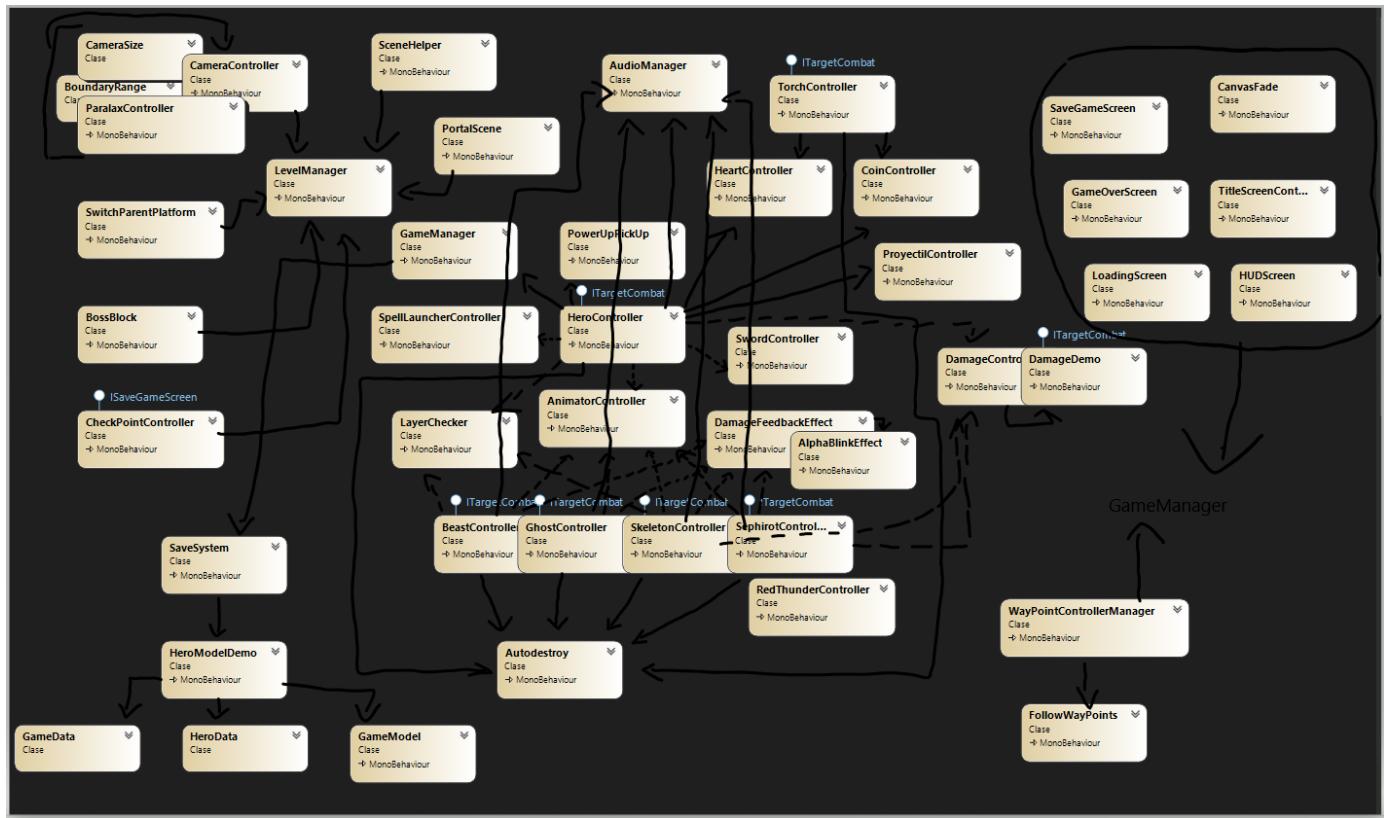
DamageController se apoya de damageDemo para hacer daño a los objetos y SwordController implementa la logica del daño del jugador, así como su rango de ataque.

Nota: TODAS ESTAS CLASES ESTAN REALIZADAS POR MI Y NO SON PROPIAS DE UNITY.

Ahora adjuntare las relaciones entre las clases contrayendolas para que se vean de un vistazo:

UML

Aclaración: En este diagrama solo aparecerán las principales, por ejemplo, DemoController al ser Usada por DemoController siempre, solo aparecería DemoController.



He usado Paint, para asociar, ya que he generado las clases con Visual Studio y no permite dibujar flechas.

4.3. Análisis y diseño de la interfaz de usuario. Mockups

En vez de realizar un Mockup, debido a las características de mi proyecto y que para que se aprecie bien el detalle de las interfaces de las escenas, aportare directamente las interfaces que he diseñado en Unity.

De esta forma puedo explicar mejor el diseño de niveles e interfaces, con más detalle, ya que me sería imposible dibujar a ese nivel detalle las cosas en un Mockup y además lo que voy a aportar vendría siendo como el Mockup propio que tiene Unity ya que es un timelaps en el que tu dibujas. Luego ya le das lógica y programación, pero el timelaps es el dibujo puro, así que lo veo conveniente en mi proyecto.

Dicho esto, mi aplicación tiene distintos tipos de interfaces, las UI, que serán las típicas pantallas de Game Over, Pantalla Inicial, Pantalla de Carga, de guardado y la UI con los elementos como vida, corazones y monedas disponibles.

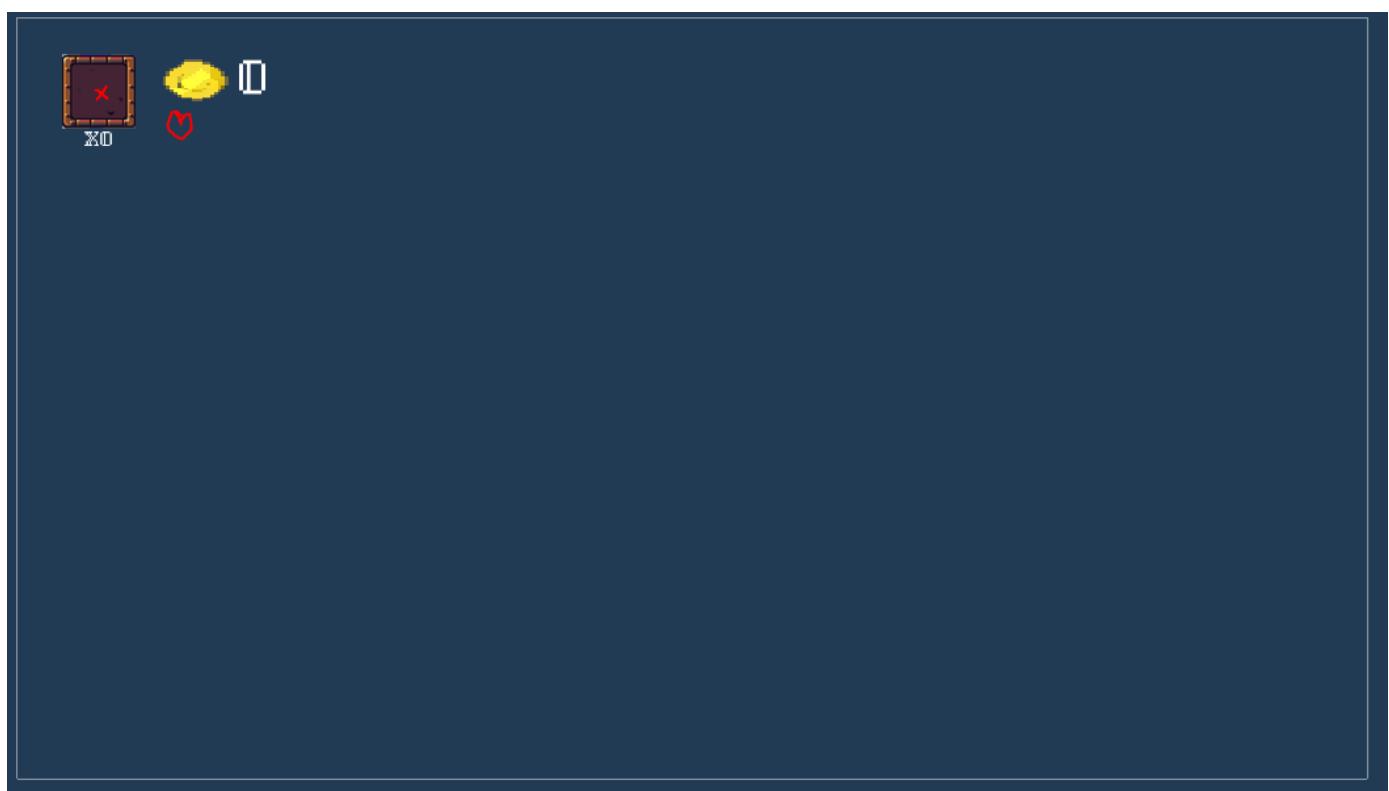
Por otro lado, tiene las interfaces que luego aportándole los scripts y las colisiones pasan a ser el mapa del juego con interacción. Estas interfaces serán las escenas, pero al principio son lienzos sin ningún tipo de lógica. Serán 9 niveles y dos pantallas de checkPoint.

Interfaz GameOver



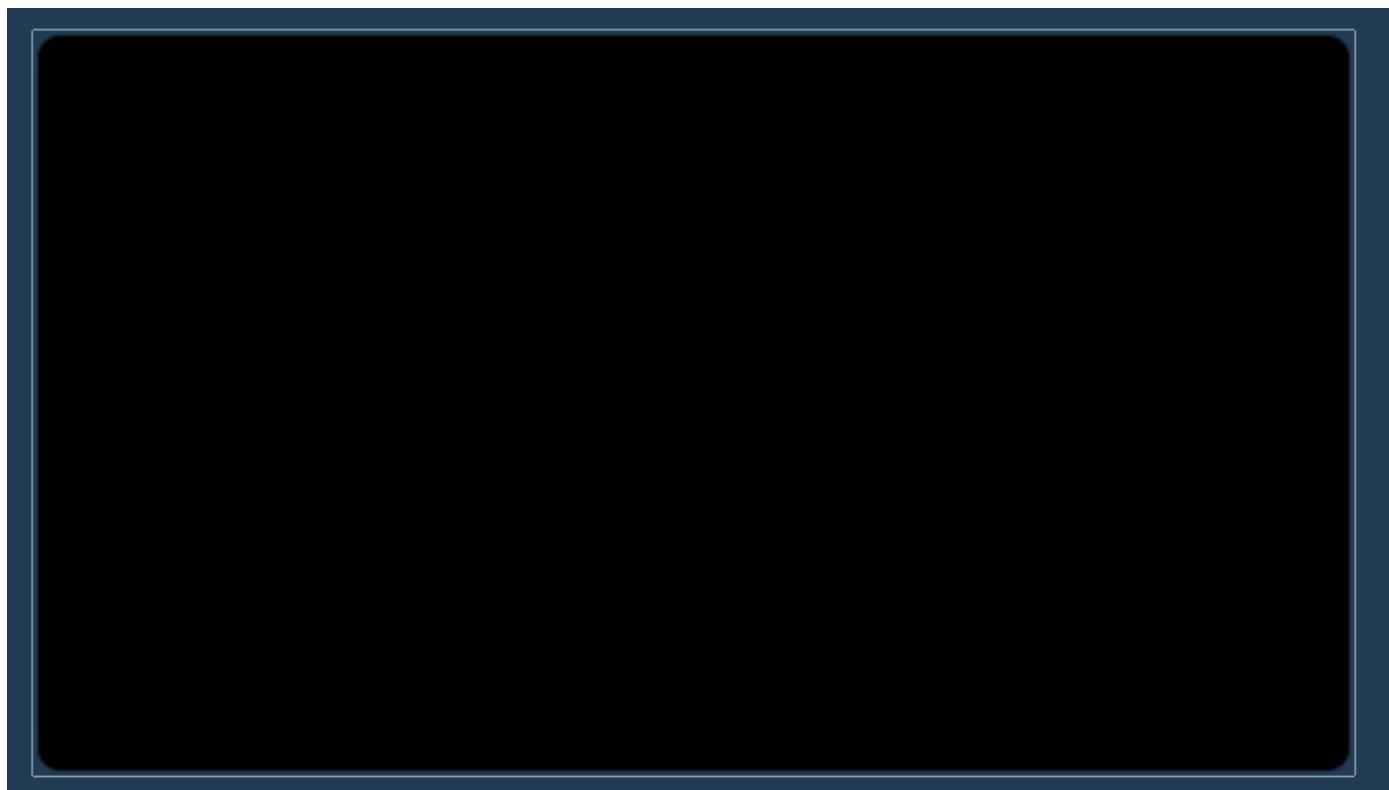
Esta interfaz será llamada al cumplirse la condición de que el jugador muera. Dispondrá de dos botones, “Yes” cargara el nivel en el que estuviese de nuevo y “No” cargará la interfaz de Pantalla Principal.

HUD



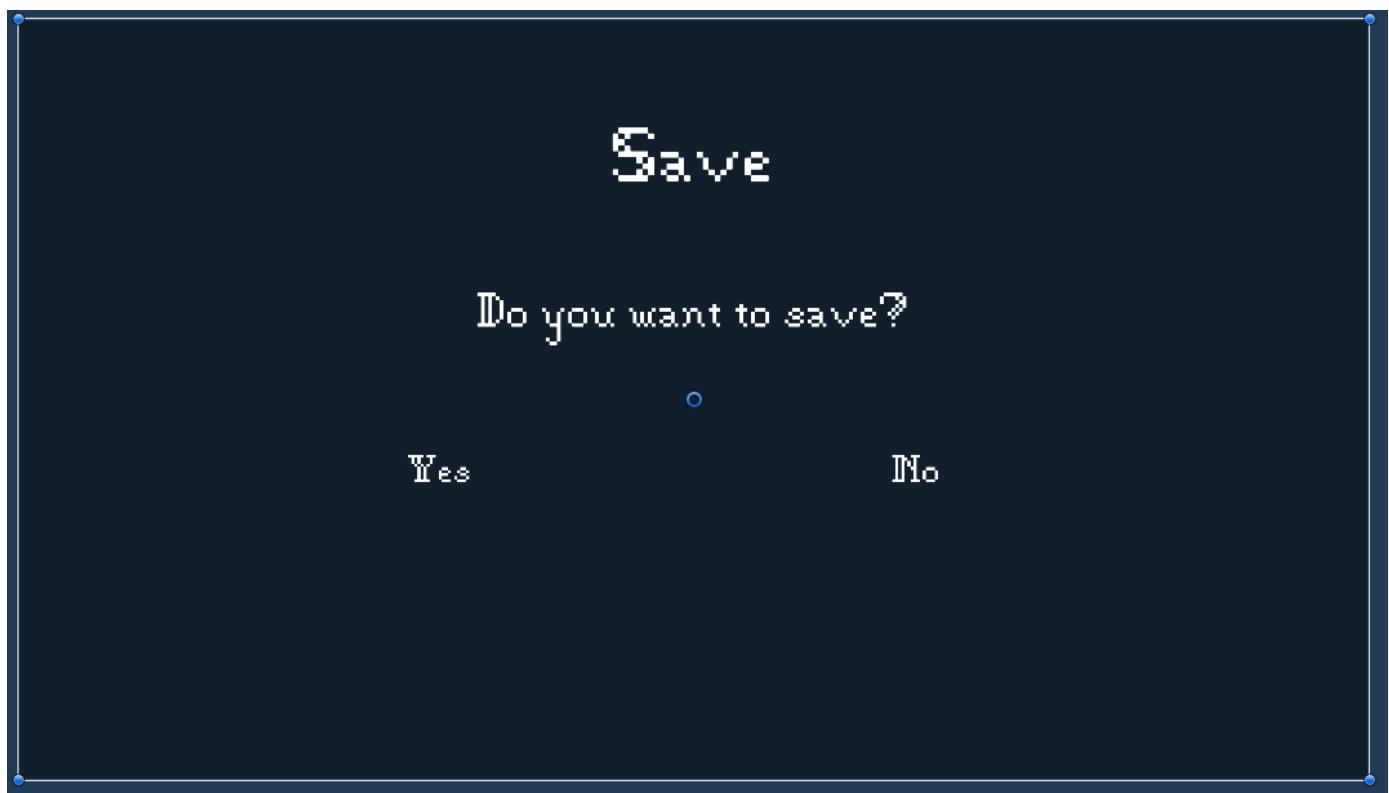
Esta interfaz pondrá la magia donde la X, actualizara el numero de monedas y dibujara un corazón por cada punto de vida que tenga el heroe. Se mostrará en funcionamiento en el apartado de pruebas.

LoadingScreen



A simple vista parece una pantalla negra y ya. En parte es así, pero al aplicarle la logica se llamara entre escenas para simular que carga la pantalla y se le ira quitando el color con un degradado lo que aportara un efecto de cortina entre escenas.

SaveGameScreen



Sera invocada en las salas seguras al acercarse al fuego central de la pantalla y permitira guardar la partida, aplicando persistencia de los datos.

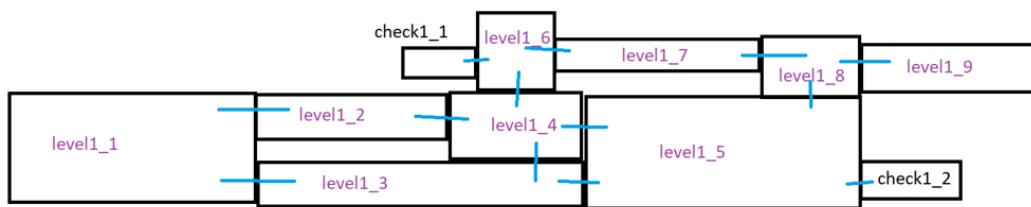
TitleScreen



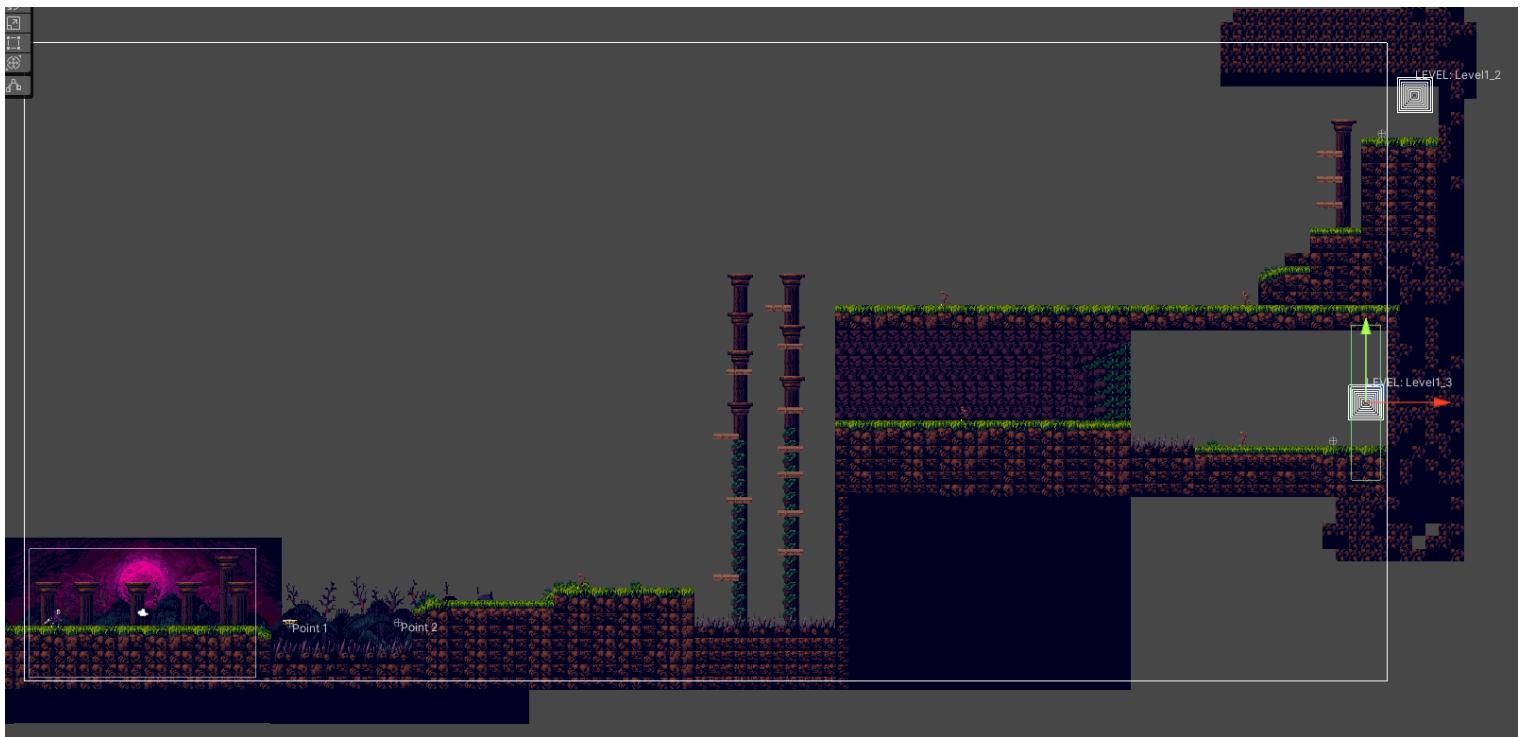
Sera la pantalla inicial del juego, al pretar Enter pasara a la siguiente pantalla que tendra tres opciones, New Game, Continuar (cargara una partida gracias a la persistencia de los datos) y Exit. Decir tambien que el boton de continuar solo aparecera si hay una partida guardada.



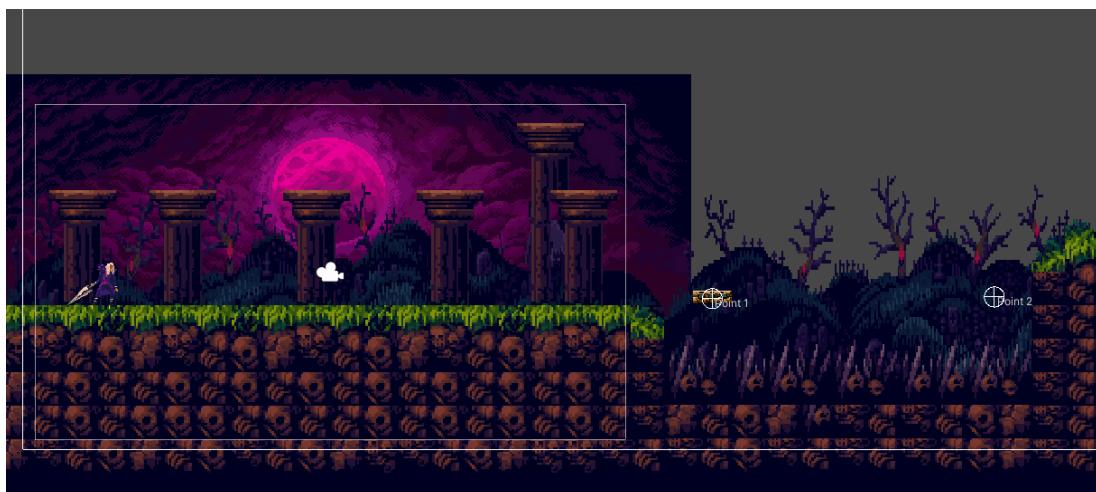
Adjunto mapa conceptual de los niveles para entender la interconexión entre ellos:

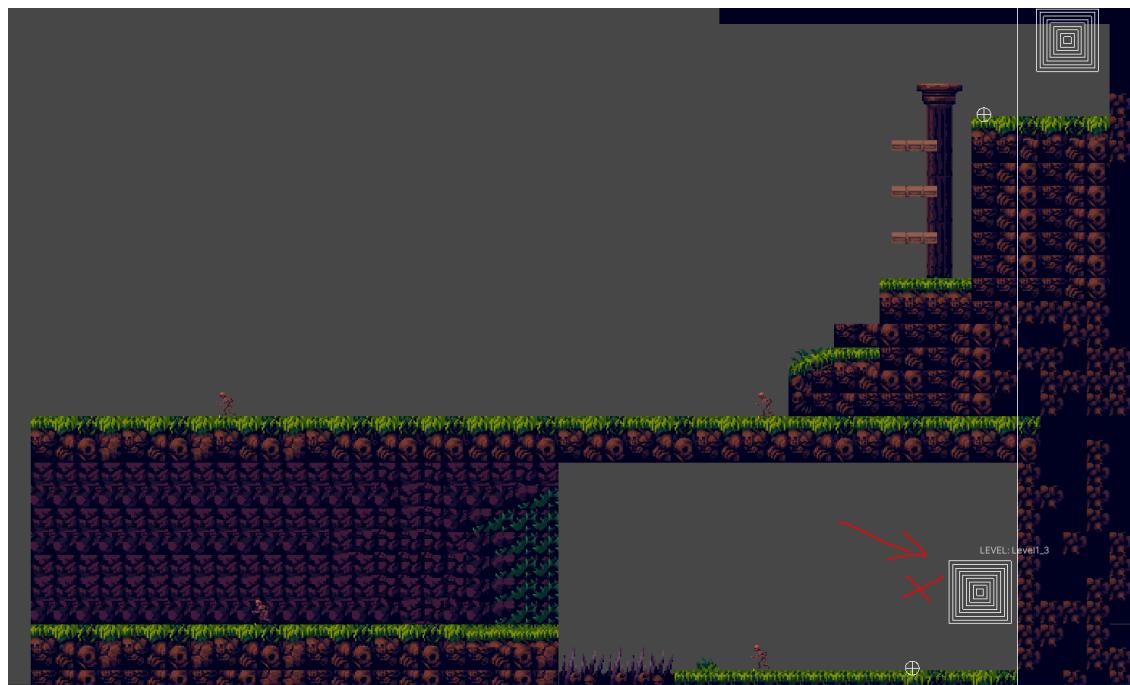
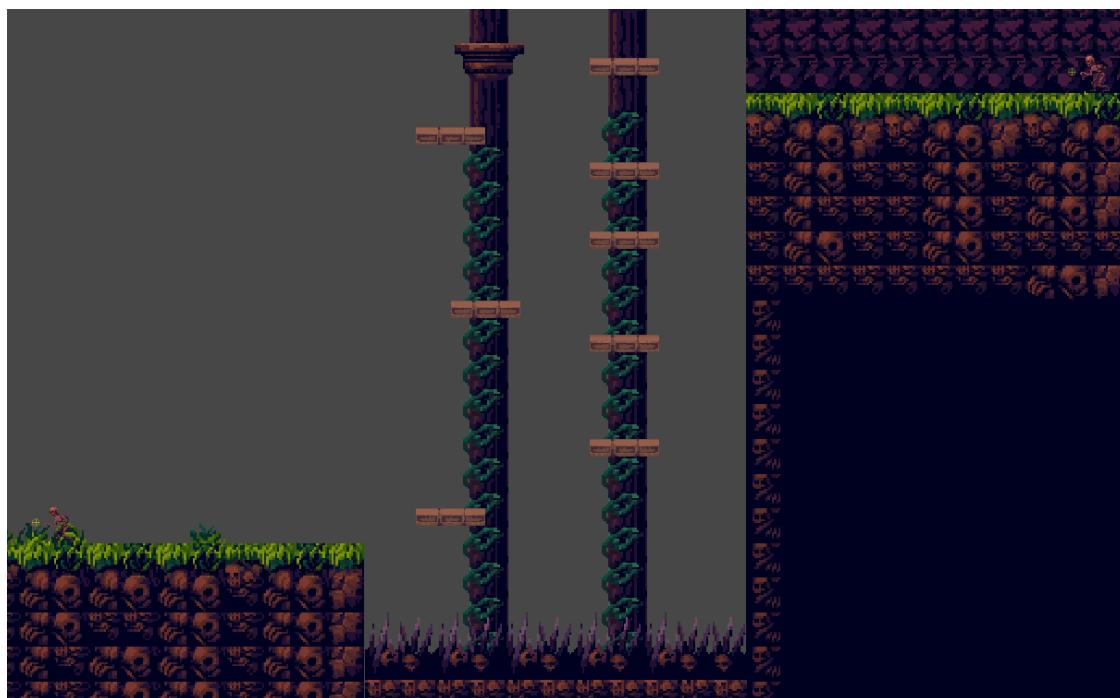


Interfaz level1_1



En este nivel habrá dos caminos uno al level 1_2 y otro al level 1_3 y los enemigos al ser el primer nivel solo serán esqueletos. Adjunto imágenes más de cerca de algunos puntos de la interfaz:





Aclaración, la X representa un portal, al llegar allí, se pasara a la escena correspondiente.

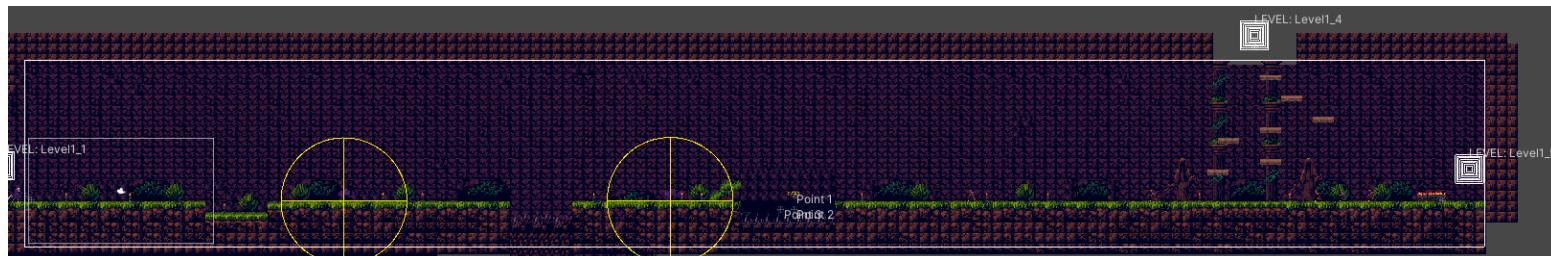
Interfaz level1_2



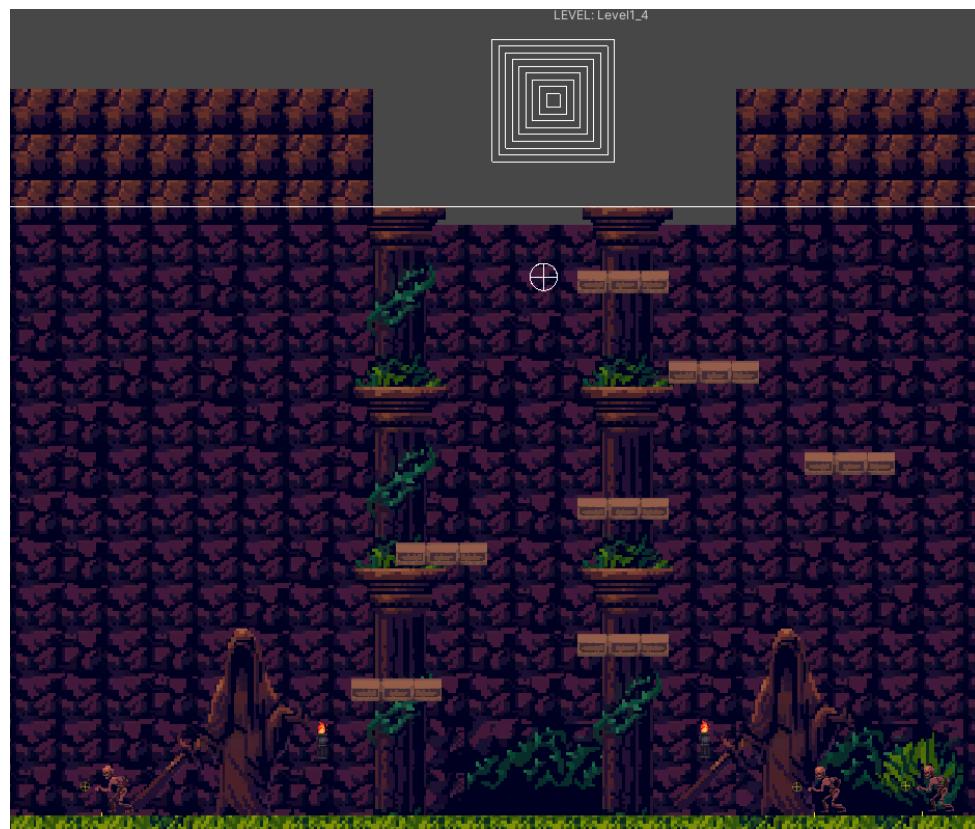
Aquí ya aparecen los enemigos beast.

Aclaración: Las áreas son el área de visión del enemigo, si entras en esa área su IA está programada para que te sigan.

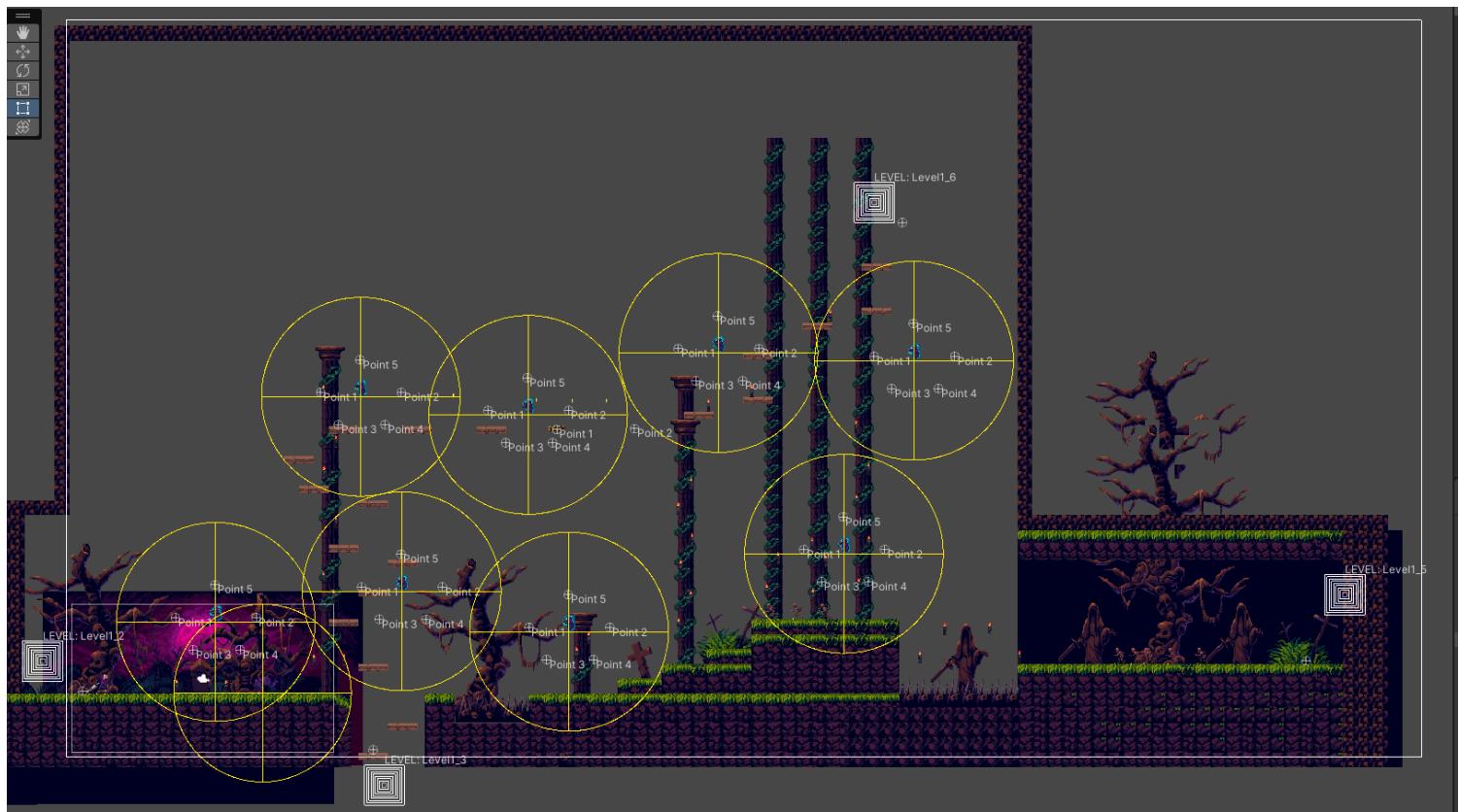
Interfaz level1_3



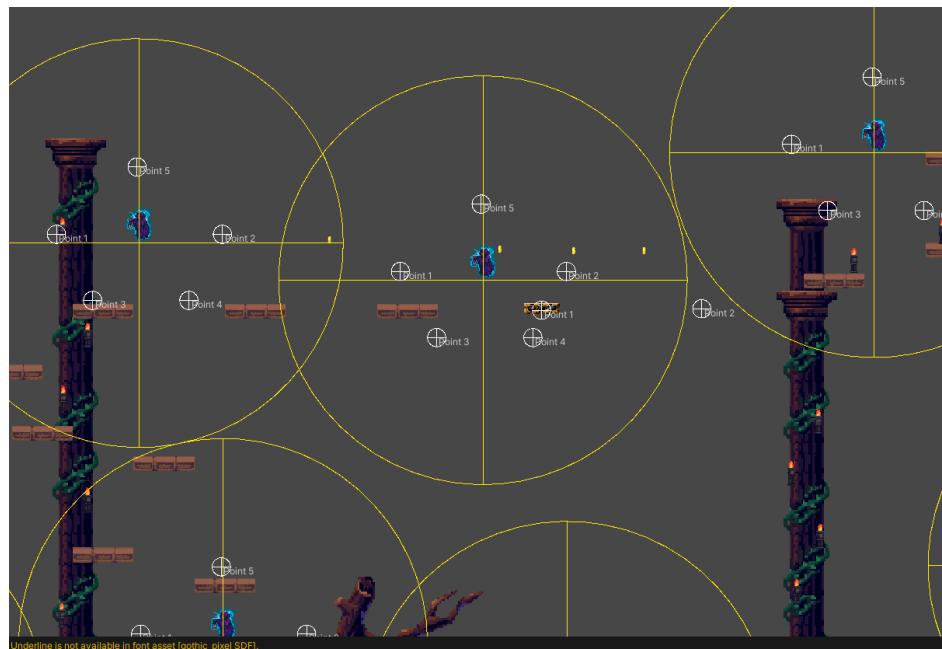
En este nivel hay antorchas, más beats y nuevos sprites de decoración y se puede acceder por la parte de arriba al level1_4 y por la derecha al level1_5. Adjunto imágenes más a detalle:



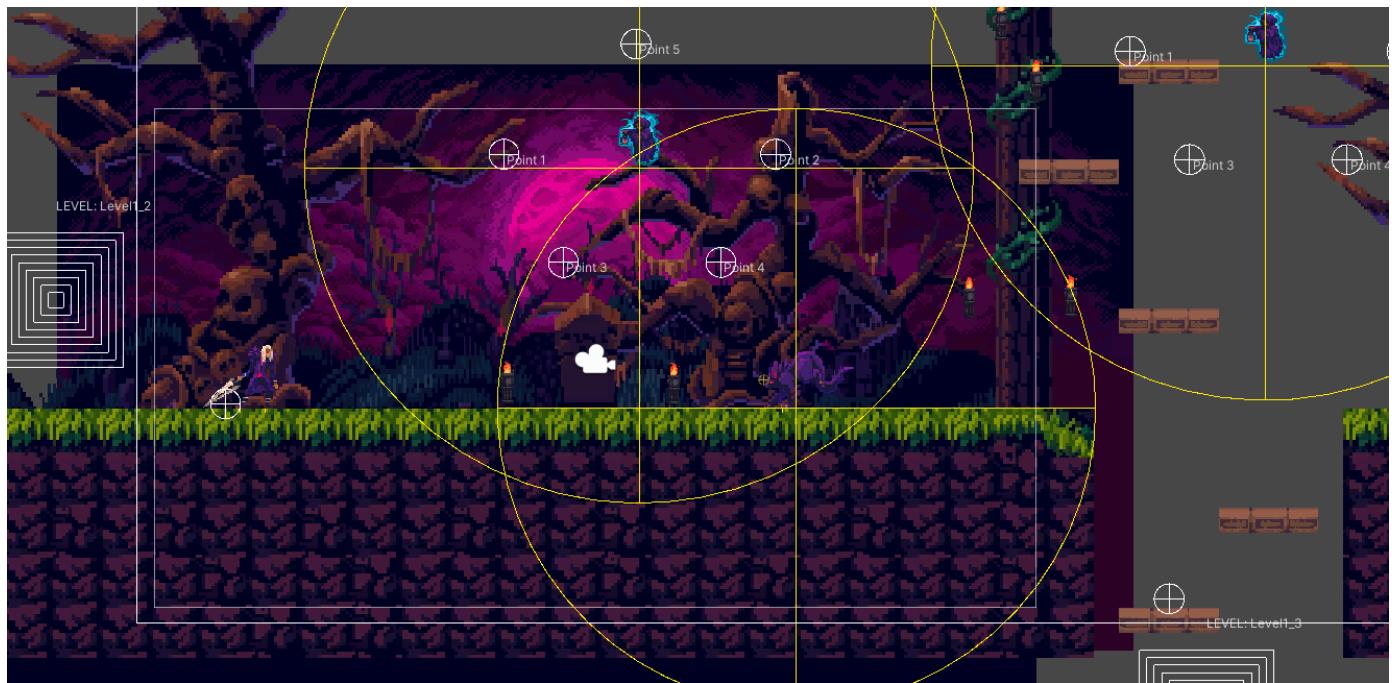
Escena level1_4



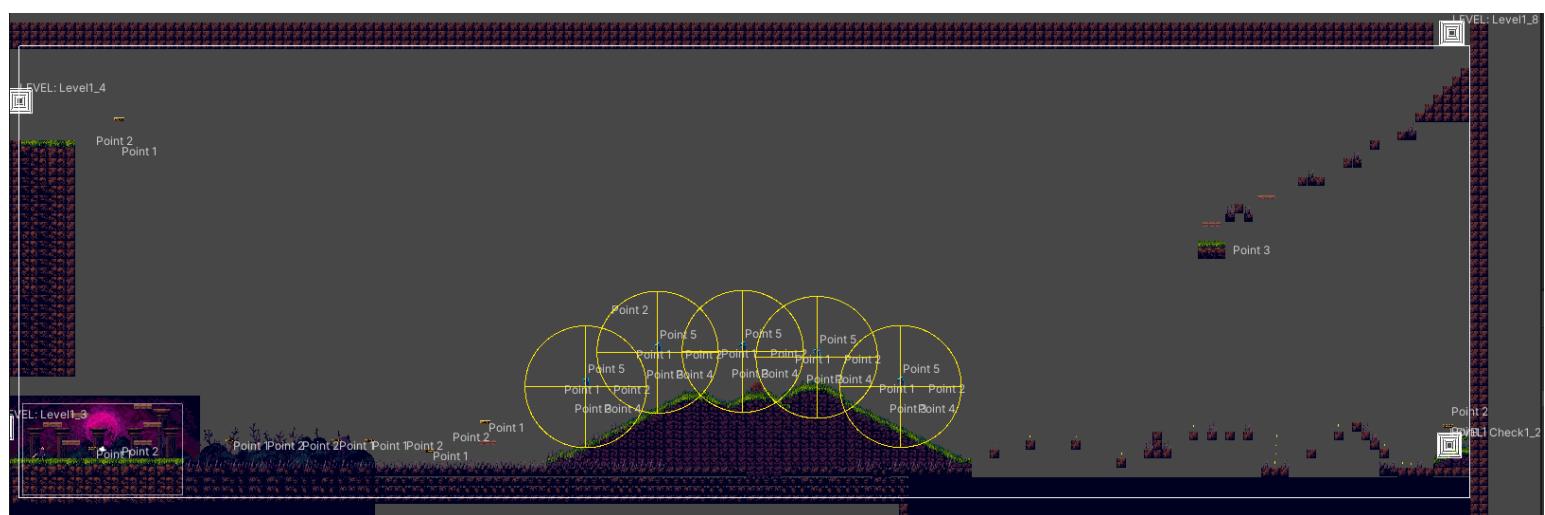
Este nivel tiene 4 conexiones, las dos para volver a level1_2 y _3, por arriba a level1_6 y a la derecha a level1_5. Este nivel está lleno de fantasmas que te siguen, y tiene algunas monedas. Adjunto imágenes más a detalle:



Aclaración: El fondo te sigue, es decir no se vería lo gris de fondo. Ejemplo del fondo (Varia según la interfaz):



Escena level1_5



En este nivel se puede retornar al level1_3 y _4, hacia arriba ir al level1_8 y a la derecha hay una sala de CheckPoint.

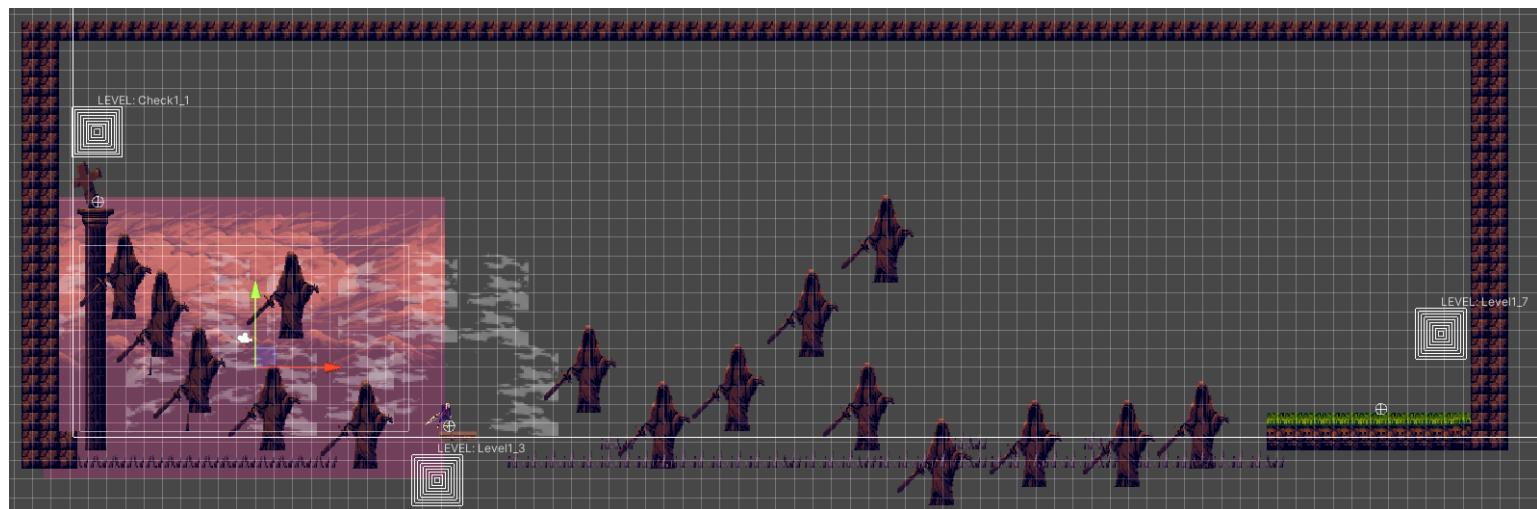
Este es el típico nivel de plataformas móviles con pinchos debajo y caída libre. Adjunto imágenes en detalle de la interfaz:



(Las plataformas se mueven entre los puntos)

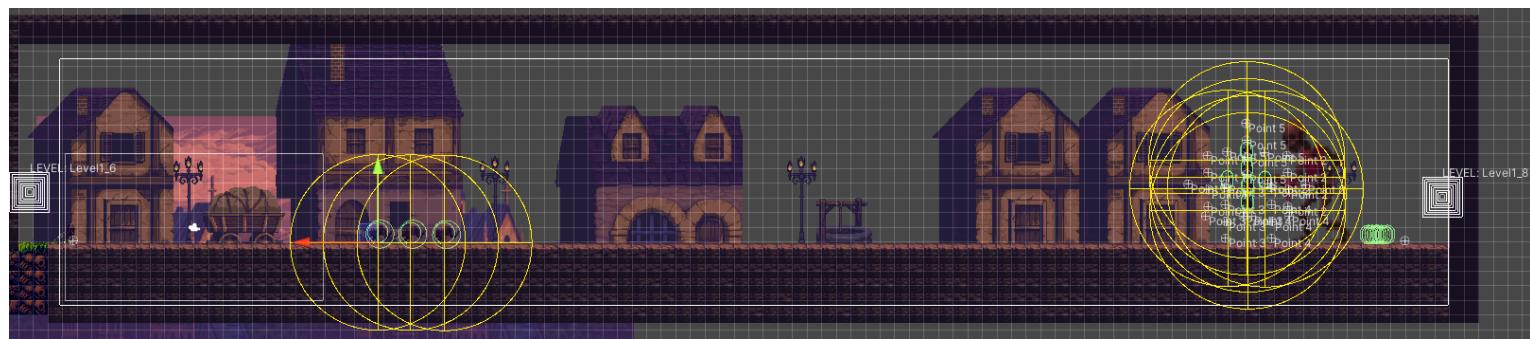


Escena level1_6



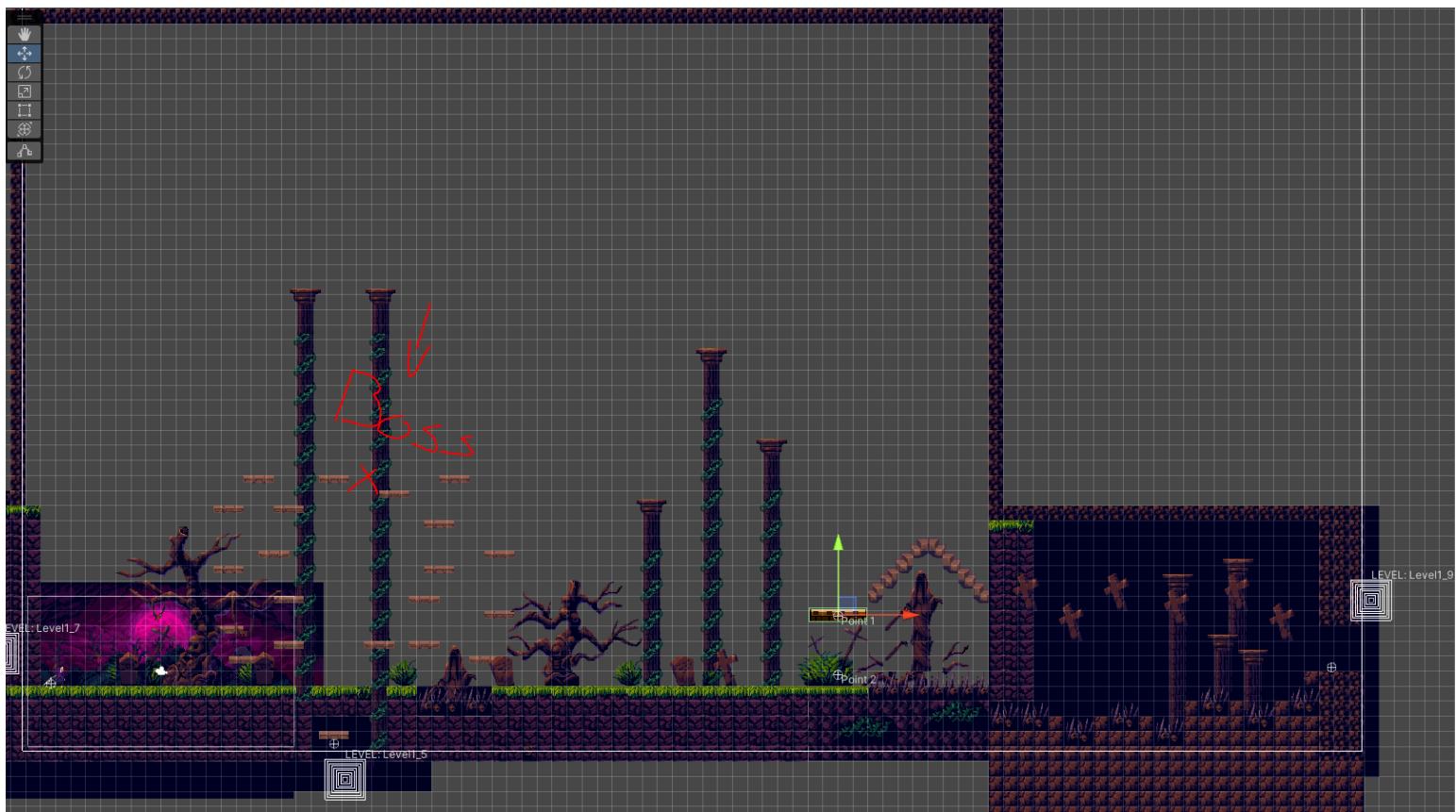
Esta interfaz puede retornar al level1_4, tiene acceso por la derecha al level1_7 y contiene una sala de check a la izquierda. En este punto el fondo sería el cielo, ya que en teoría según el cambio de escenas el personaje está bastante arriba. Aclarar que en este nivel debe saltar entre los ángeles sin caerse.

Escena level1_7



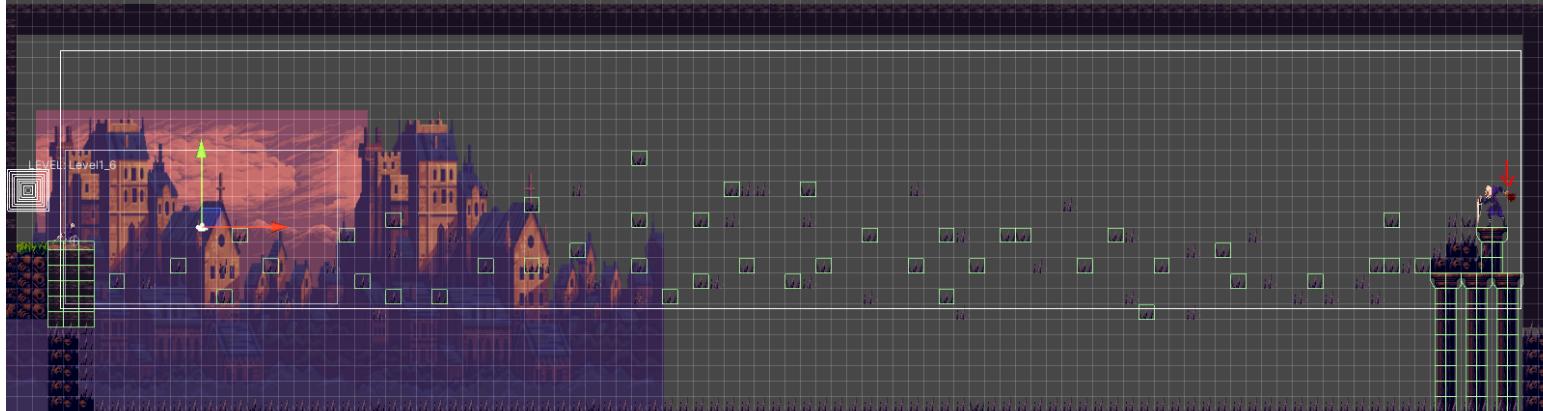
Esta interfaz conecta con el level1_6 y _8. Este nivel es la antesala del boss, representa un pasillo lleno de enemigos.

Escena level1_8



Es la sala del boss final, conecta con el nivel 1_9 y permite retroceder al 1_7 y 1_5. En este nivel las entradas quedarán bloqueadas cuando empiece el combate con el boss final, que aparecerá en la X.

Escena level1_9



En este nivel solo habrá un pasillo, lleno de pinchos el cual unos serán de tipo ground y otros mataran al personaje al pisarlos, la diferencia entre el aspecto de ellos es mínima, así que identificarlos para llegar al final es muy complicado. En la imagen resalto los que no dañan. Al final del todo el héroe ve que lo único que hay es un reflejo suyo de mayor tamaño (como una deidad). Tu otro yo, estará utilizando la magia del anterior boss final, la idea es que él ocupa su lugar ahora. La gracia de este nivel es descubrir eso y que no se puede alcanzar a esta deidad, porque los pinchos para llegar hasta él hacen todos daño mortal, por lo tanto, descubres esto y tienes que morir sí o sí.

Escenas Check

Ambas escenas son iguales, pero cambiando la dirección de los elementos, así que adjunto una. El fuego del medio al acercarte ofrece guardar la partida.



4.4. Diseño de la arquitectura de la aplicación

4.4.1. Tecnologías/Herramientas usadas y descripción de las mismas

Assets: Cuando hablo de assets en Unity hablo de elementos que están físicamente en la carpeta del proyecto, pero que aplicándole una lógica con Unity pueden pasar a ser ítems del juego, ya sean imágenes, scripts, fuentes de texto, archivos de sonido y otros elementos creados con Unity que puedan ser usados en otros proyectos. Por ejemplo, podríamos guardar ítems completos con su lógica implementada en forma de prefab (que son archivos de ítems en el proyecto físico) en las carpetas, el héroe completo mismo. Con arrastrar ese archivo prefab en otro proyecto tendría integrado el héroe exportado de mi propio juego sin hacer nada con la lógica con la que cree un archivo físico de ese ítem.

Sprites: Son imágenes estáticas, que sirven para posteriormente darle animación interponiéndolas

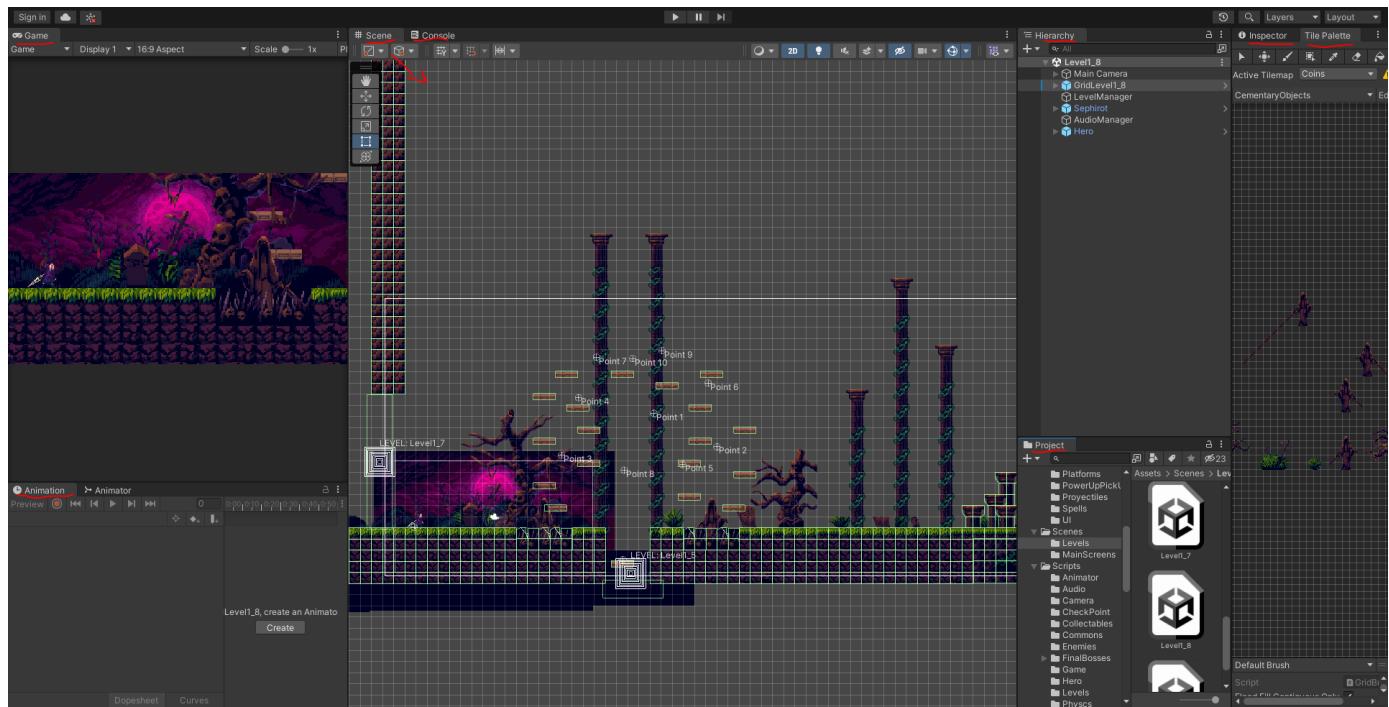
rápidamente con ayuda del ide de Unity:



Unity: Es una herramienta, framework, que permite crear videojuegos para distintas plataformas, ya sea Consolas, Móviles, Pc, Mac... gracias a un editor visual (el ide) apoyándose de los scripts.

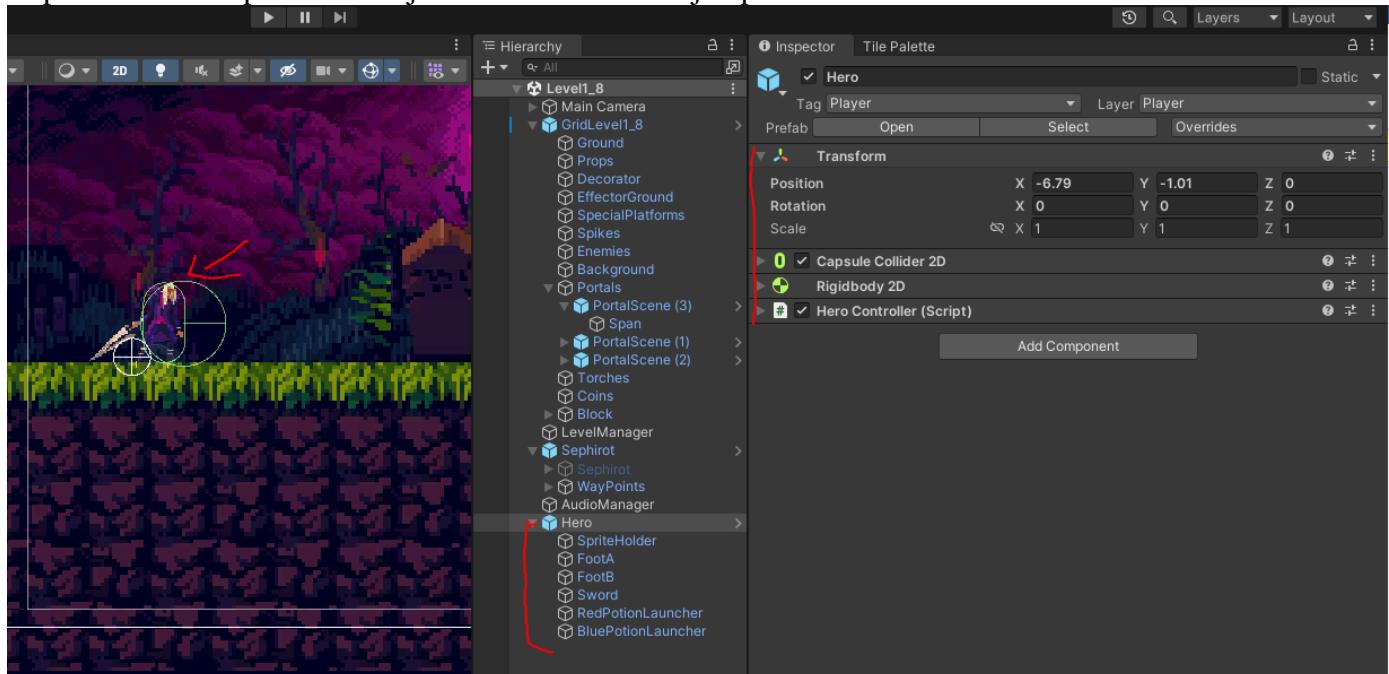
¿Como se da vida en unity a un lienzo?

El IDE de Unity se compone de Game, donde ves el juego ejecutándose, Animaciones, donde gestionas los tiempos de las animaciones, Hierarchy donde ves los objetos que hay en el lienzo, project con la arquitectura física del proyecto, el inspector donde ves los parámetros de los objetos de Hierarchy, la paleta, donde se encuentran los assets que pongas para pintar en el lienzo y editar el lienzo, la consola con los errores de los scripts y la escena, donde pintas y pones los objetos.



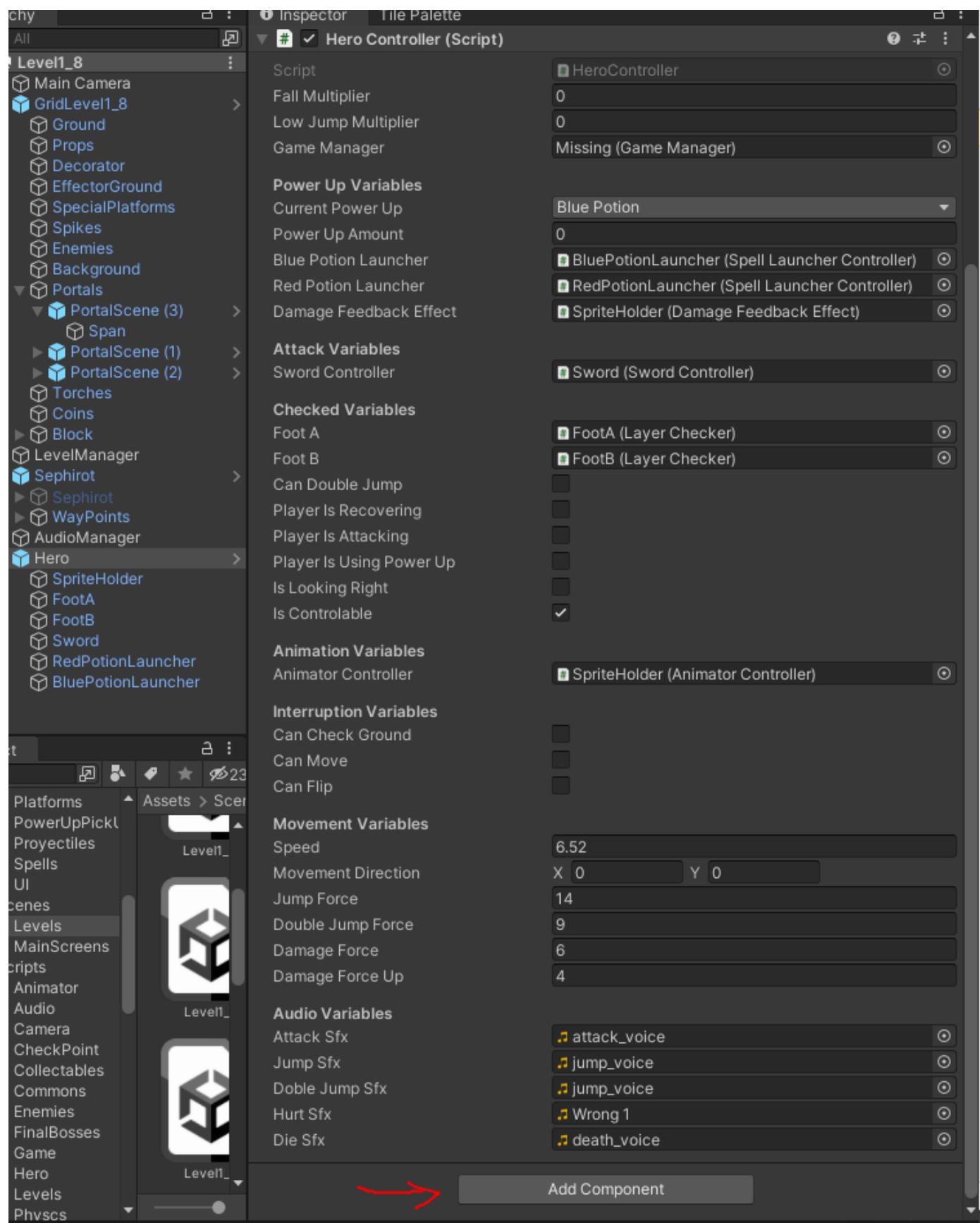
La idea de unity es pintar el lienzo, una vez hecho esto creas objetos, a estos objetos le introduces sprites para que tengan forma física, colisionadores y tal. Una vez configurados los pones en la escena juntándolos con el lienzo. ¿Pero cómo contienen la lógica? La lógica la tienen gracias a que estos objetos aparte de colisionadores y demás pueden recibir scripts para modificar la lógica de los objetos que ya poseen o establecer parámetros que se podrán alterar desde el ide. Estos scripts son clases, controladores, interfaces... y se programan en C# usando el código común de dicho lenguaje y apoyándose de librerías internas de unity para acceder por ejemplo a los colisionadores instanciándolos en dichos scripts. ¿Y como se accede a estos objetos del IDE? Se accede gracias a el nombre que le pones del objeto, ya que C# tiene métodos integrados de unity para buscar dichos objetos en la escena. También se pueden acceder mediante tags.

Explicare el concepto de los objetos en el ide con el ejemplo entero de Hero.



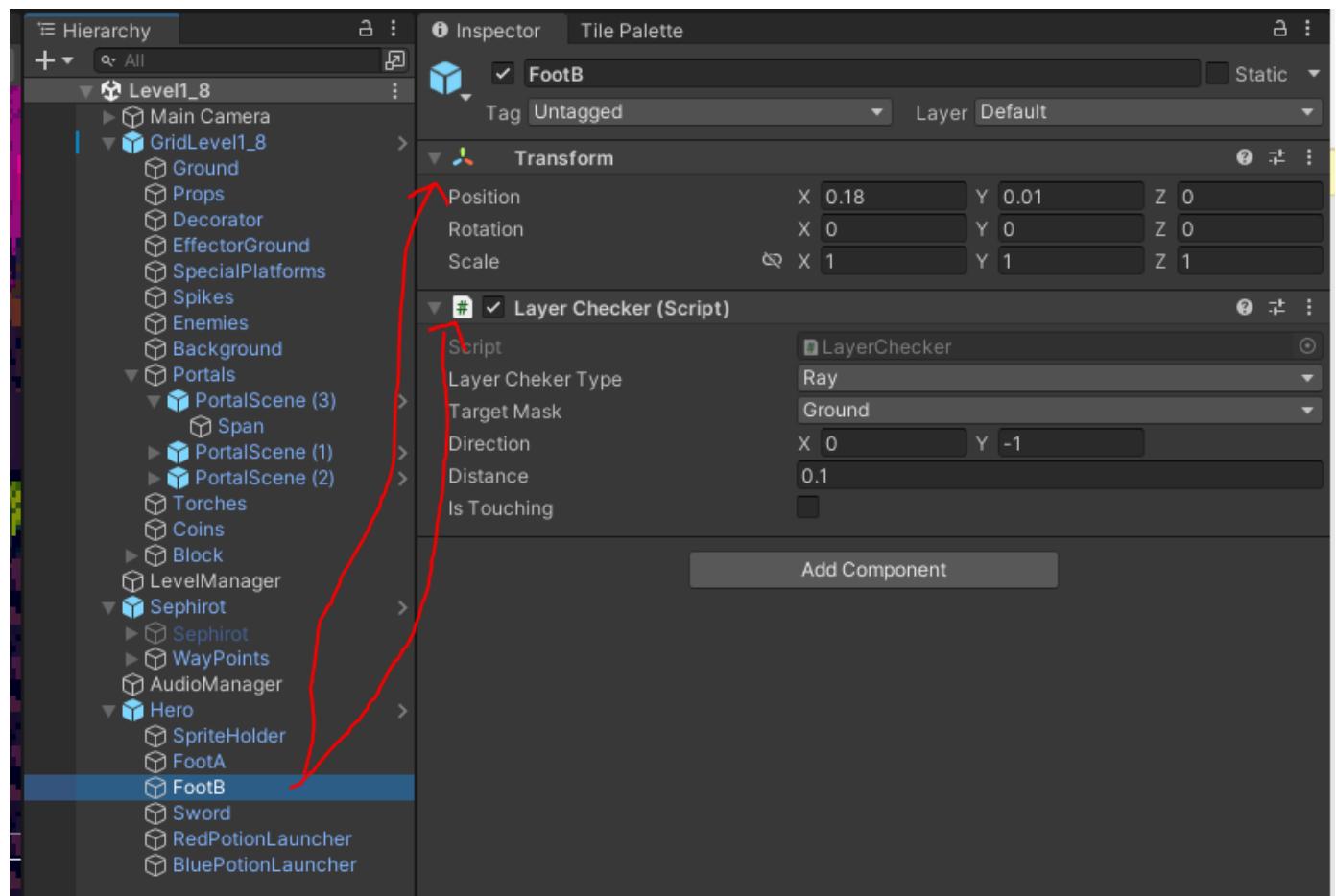
Como se puede ver si despliego Hero este se compone de otros objetos hijos, dandole funcionalidad extra al objeto padre Hero. A la derecha se puede observar en el inspector que el objeto padre contiene transform

(para el aspecto del sprite), capsule Collider 2D el cual es el area circular que se ve en el heroey detecta cuando hay colisiones con otros objetos que tienen areas similares, rigidBody, que es el darle forma tangente al sprite, es decir, que otros objetos puedan chocar y no avancen al chocar con el area más pequeñita del heroe y por ultimo y más importante HeroController el cual contiene la logica y creo conveniente explicar a fondo. Nota, tambien se pueden añadir componentes con otras funcionalides en Add Component.

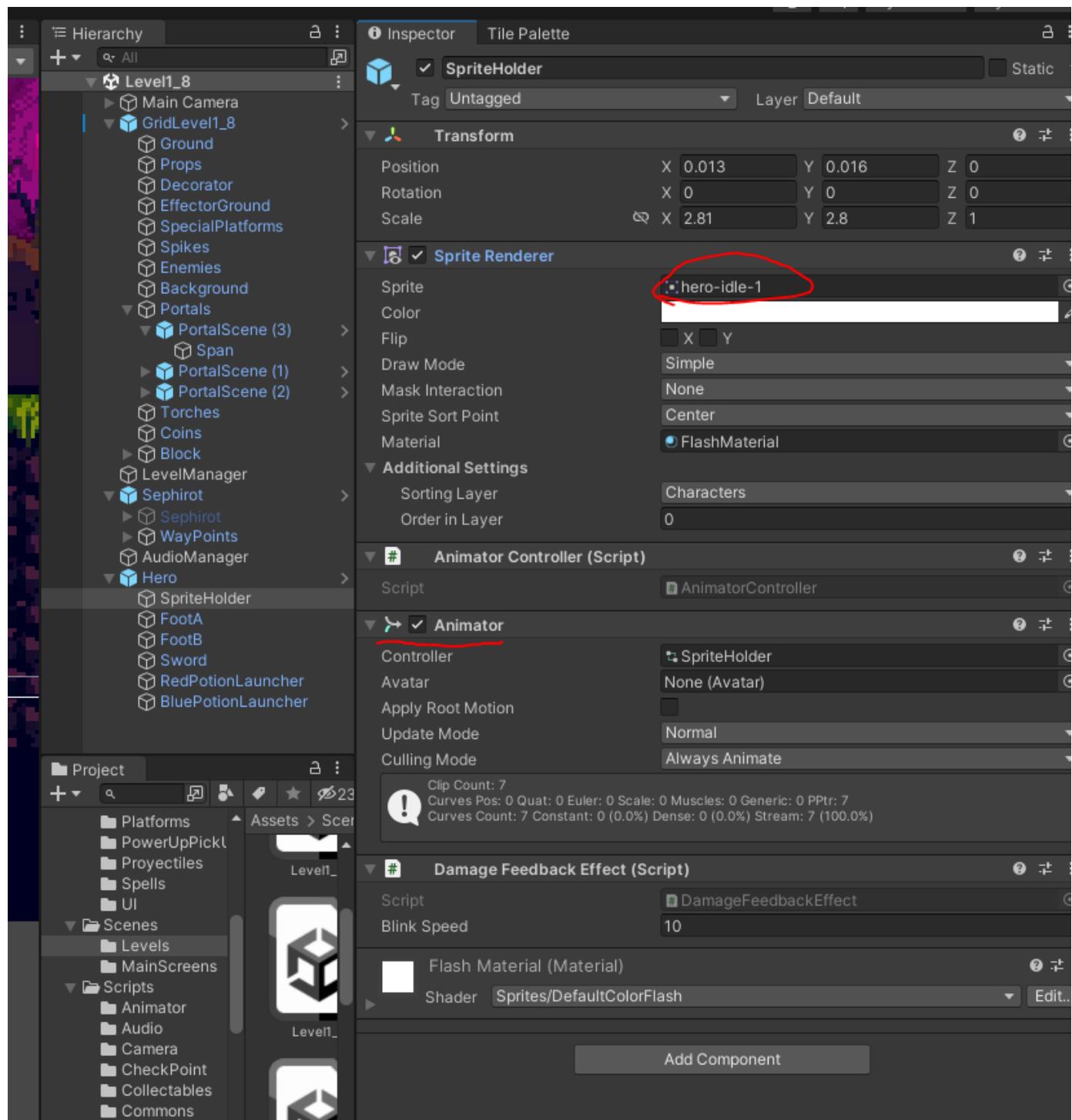


Como se puede observar al desplegar este script se pueden observar las variables que pueden recibir parametros. Esto se debe a que podemos setearlas desde el propio id, parametrizando el script, incluso con otros scripts como se puede ver con el animator controller o incluso con otros objetos del IDE como se

puede ver con FootA y footB que corresponden a objetos hijos del heroe. Aquí solo se puede parametrizar los scripts, ya sea desde el ide o de forma programatica desde dentro en C#, los metodos y la logica que se consigue usando estos parametros se maneja desde dentro del script (clase). A continuación voy a desplegar un hijo de Hero para que se pueda ver que los hijos tambien implementan otros objetos y scrips que aportan logica al objeto padre:

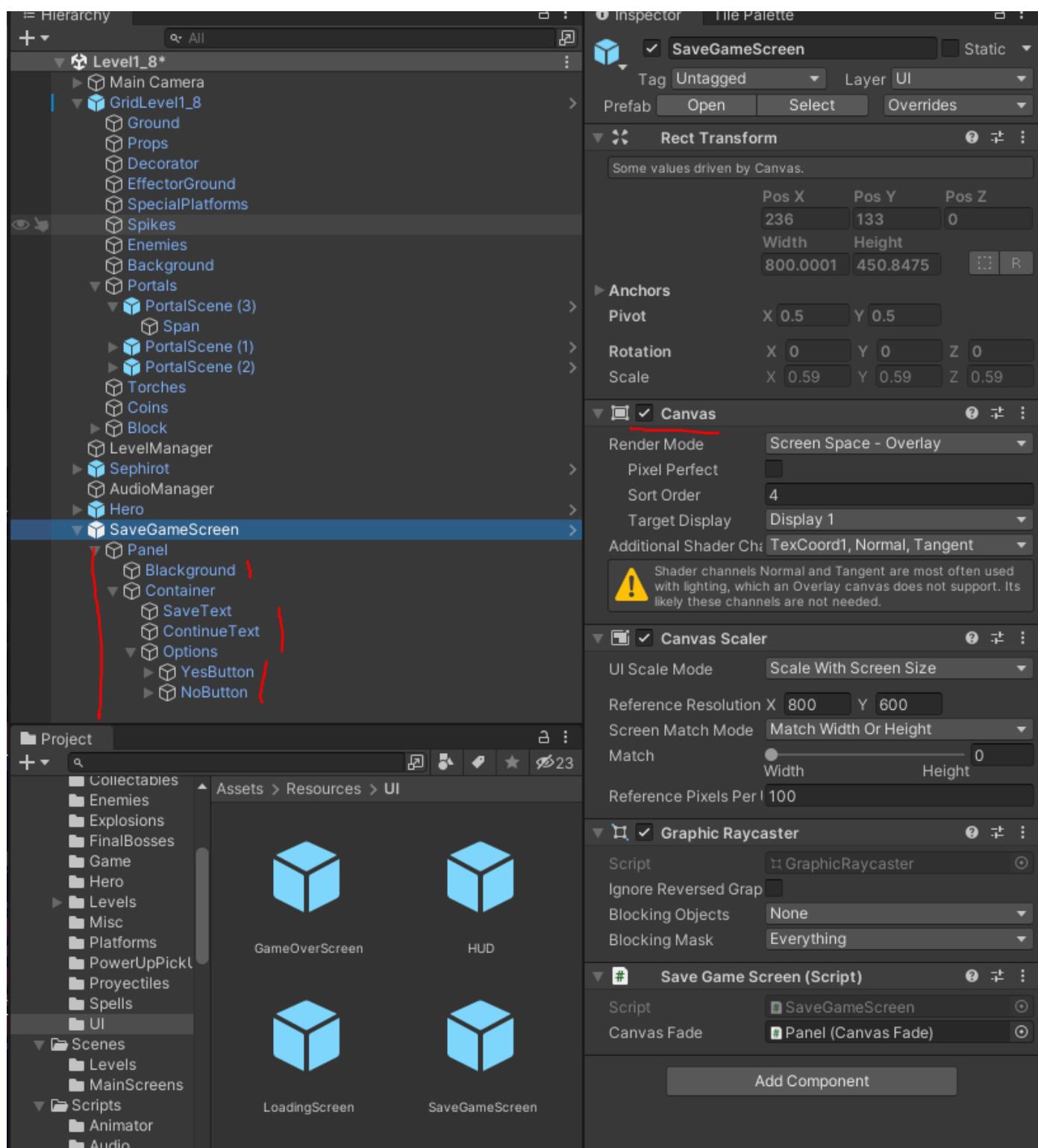


Esto seria la logica, y la forma visual se controlaria con el objeto hijo SpriteHolder, que contendra el sprite (ya explique lo que era) con sus animaciones:

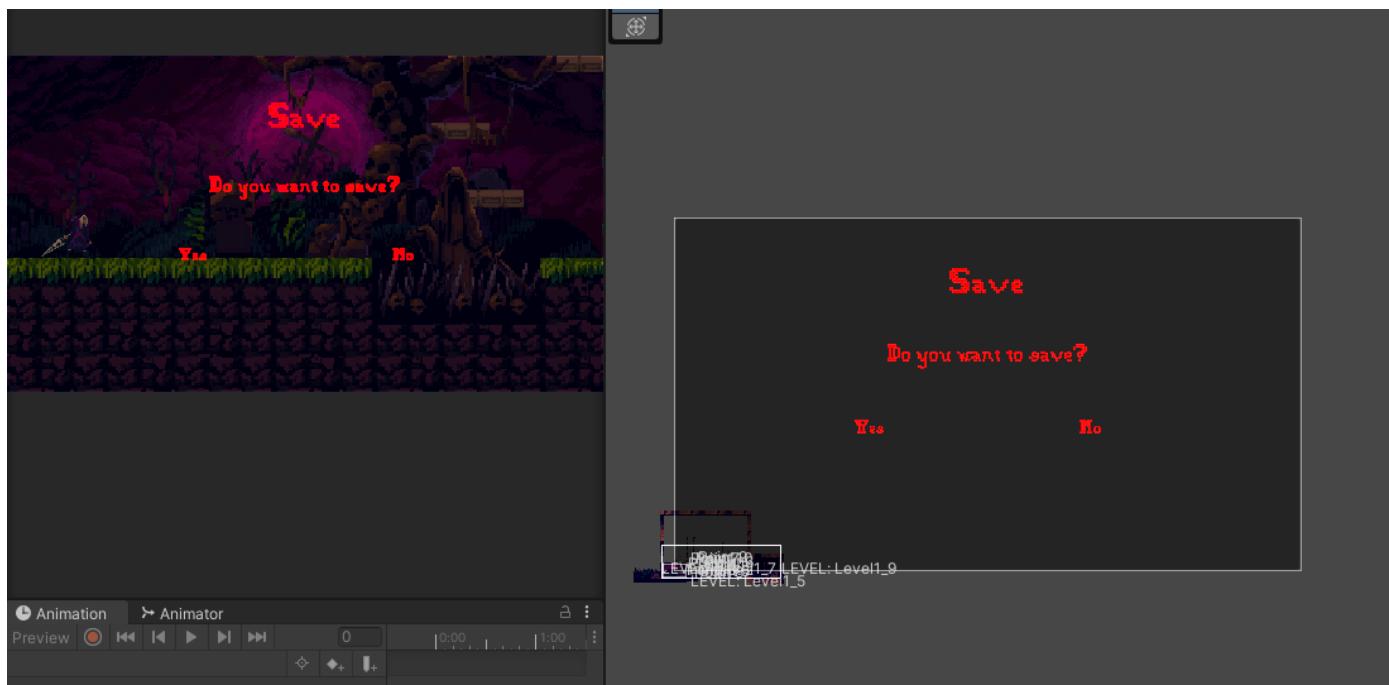


Lo resaltado en el círculo representa a la imagen que se le introduce a este objeto.

Explicado el objeto Hero, el suelo, el fondo, la camara... todo sigue la misma lógica , incluso las interfaces realizadas con canvas. Las interfaces realizadas con canvas se añaden como objeto de tipo UI a la escena y se manejan de forma parecida a android studio o visual studio, añadiendo los botones, paneles... adjunto ejemplo de objeto de interfaz:



Como se puede observar contiene paneles, textos y botones, cada hijo con su propia logica en el inspector de la derecha. El resultado de esta UI con dos textos y los botones seria el siguiente:

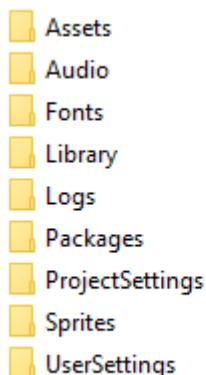


Me parecía muy conveniente y necesario entrar a detalle acerca del framework de Unity para poder llegar a una comprensión más alta y enlazar la logica con el lienzo.

Visual Studio 2022:

Es el ide de codigo que he utilizado para programar los scripts con sus clases, interfaces... en C#. Realmente se podria programar el codigo en un bloc de notas y funcionaria igual, pero visual studio ofrece muchas ventajas aportando diversas herramientas como el autocompletado, marcando errores de complicación... y sobre todo por el que lo he elegido facilitando la creación del UML, aportando una vista general de los scripts del proyecto, facilitando su comprensión y ver de forma visual los parametros y metodos de dichos scripts.

4.4.2. Arquitectura de componentes de la aplicación



- Assets lo he explicado en un punto anterior, audio es el sonido, fonts las fuentes de texto, library las distintas librerías necesarias, sprites esta explicado, user settings y project settings son configuraciones del ide, logs historial de errores y packages son archivos necesarios de unity.
- La carpeta de Assets merece la pena reseñarla por dentro:

 Animations	01/06/2022 21:58	Carpeta de archivos
 Audio	01/06/2022 21:58	Carpeta de archivos
 Fonts	01/06/2022 21:58	Carpeta de archivos
 Materials	01/06/2022 21:58	Carpeta de archivos
 Resources	01/06/2022 21:58	Carpeta de archivos
 Scenes	01/06/2022 21:58	Carpeta de archivos
 Scripts	01/06/2022 21:58	Carpeta de archivos
 Sprites	01/06/2022 21:58	Carpeta de archivos
 TextMesh Pro	01/06/2022 21:58	Carpeta de archivos
 TitlePalettes	01/06/2022 21:58	Carpeta de archivos

Animations contiene las animaciones que he ido creando.

Audio el audio del juego, fuentes igual,

Materials contiene distintos componentes para dar efectos visuales a los objetos.

Resources es donde se guardan los prefabs ya explicados (guardar ítems ya completos de tu juego para poder ser invocados en el propio juego u en otros).

Scenes es donde se guarda las escenas creadas.

Scripts contiene todos los scripts usados.

Sprites las imágenes, textMeshPro los textos de las UI y TitlePalettes contiene los lienzos utilizados en las interfaces.

Dentro de la carpeta scripts he dividido los scripts por carpetas según la función que cumplen:

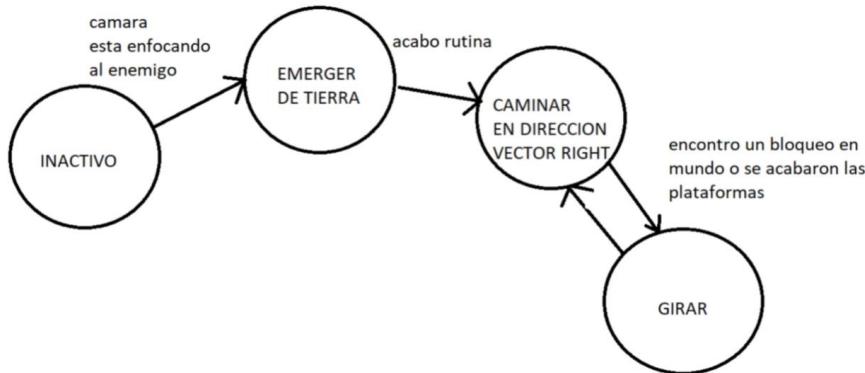
 Animator	01/06/2022 21:58	Carpeta de archivos
 Audio	01/06/2022 21:58	Carpeta de archivos
 Camera	01/06/2022 21:58	Carpeta de archivos
 CheckPoint	01/06/2022 21:58	Carpeta de archivos
 Collectables	01/06/2022 21:58	Carpeta de archivos
 Commons	01/06/2022 21:58	Carpeta de archivos
 Enemies	01/06/2022 21:58	Carpeta de archivos
 FinalBosses	01/06/2022 21:58	Carpeta de archivos
 Game	01/06/2022 21:58	Carpeta de archivos
 Hero	01/06/2022 21:58	Carpeta de archivos
 Levels	01/06/2022 21:58	Carpeta de archivos
 Physcs	01/06/2022 21:58	Carpeta de archivos
 Platforms	01/06/2022 21:58	Carpeta de archivos
 PowerUps	01/06/2022 21:58	Carpeta de archivos
 Proyectiles	01/06/2022 21:58	Carpeta de archivos
 SaveSystem	01/06/2022 21:58	Carpeta de archivos
 Scenes	01/06/2022 21:58	Carpeta de archivos
 Spell	01/06/2022 21:58	Carpeta de archivos
 UI	01/06/2022 21:58	Carpeta de archivos
 VisualEffects	01/06/2022 21:58	Carpeta de archivos
 Waypoints	01/06/2022 21:58	Carpeta de archivos
 Weapons	01/06/2022 21:58	Carpeta de archivos

5. Documento de Implementación, Pruebas e Implantación del sistema

5.1. Implementación

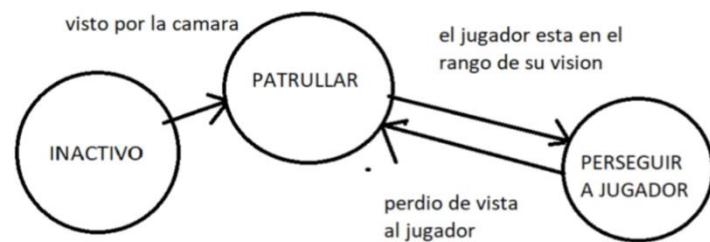
En el apartado de implementación me conviene explicar la lógica de la IA de los enemigos, ya que tienen ciertos patrones de comportamiento que creo conveniente explicar. La más interesante sin duda es la del boss final, ya que su lógica es de un nivel de dificultad a la hora de programar elevadísimo.

IA del esqueleto:



Comenzara inactivo, cuando le veamos, emergera de la tierra con una animación y empezara a caminar hacia la derecha hasta que choque con un obstaculo, girando hacia la otra dirección.

IA del Fantasma:



Empezara estando inactivo, y cuando nuestra cámara entre en contacto con el fantasma, pasara al estado de patrullar, en el cual se movera entre 6 puntos distintos de forma aleatoria. Cuando entremos en su rango de visión comenzara a perseguir al heroí, esto lo logra siguiendo la cámara, ya que esta fija en el heroí. Cuando le pierda de vista, volvera al estado patrullar.

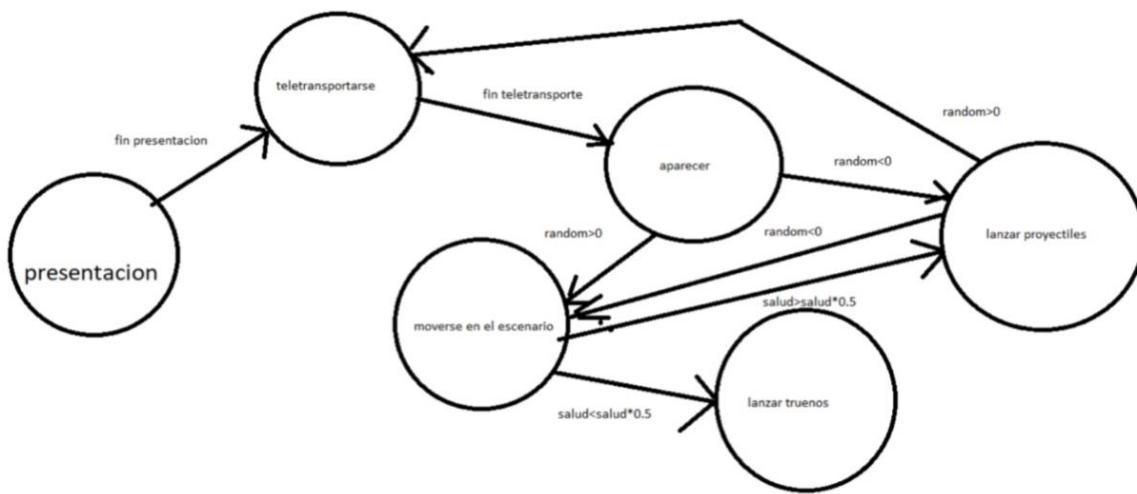
IA de la Bestia:

Sera igual a la del esqueleto con la diferencia de que cuando estemos cerca nos seguirá.

IA Boss Final Sephirot:

Este será el más especial de todos, ya que, al entrar en la sala, cuando nos acerquemos al boss final, se alejará la cámara y sellaran las salidas del mapa, por lo tanto, el propio nivel tendrá que tener IA, el cual no se desbloqueara hasta que el boss muera. La música del escenario también cambiara al aparecer este.

En la IA del boss podemos encontrar los siguientes estados:



Al comienzo, hará una presentación, en la que se volverá visible, comenzará a parpadear, se reirá maliciosamente y pasará al bucle de los estados.

En este bucle primero se teletransportará a un punto aleatorio del mapa, volviéndose invisible y apareciendo en otro punto en invisible, haciendo daño al contacto y después se volverá visible.

Al ser visible con un algoritmo aleatorio elegirá el siguiente estado, o lanzar proyectiles o moverse por el escenario, si lanza proyectiles lanzará una bola en dirección al héroe y seguido se moverá por el escenario arrollando a su paso. En este punto si la salud del boss es superior al 50 % volverá a lanzar proyectiles y comenzará el ciclo de nuevo.

Al llegar a menos del 50% de su salud después del estado de moverse por el escenario lanzará truenos, cuando realice esto, avisará con una frase y se moverá al centro de pantalla tiempo que tendremos para percatarnos de lo que viene, ya que es una habilidad con bastante rango de mapa. En este punto lanzará muchísimos rayos de forma aleatoria cerca del héroe. Estos rayos tendrán una explosión en su punto máximo, pero infligirán daño en toda su amplitud. Repetirá el proceso de los rayos 4 veces seguidas alterando donde caen los rayos y de forma rápida.

Con menos del 50% de vida el estado de lanzar proyectiles cambia, ya que lanzará 3 tandas más rápidas, teledirigidas al héroe y en la 2 tanda lanzará varios proyectiles a la vez.

Cuando el boss muera, ejecutará una animación, otorgará recompensa y a continuación el nivel desbloqueará sus salidas y la música y cámara volverán a su estado inicial.

He usado GitHub como sistema de control de versiones, pero al principio tuve problemas con ello ya que le di seguimiento también a la carpeta con las librerías, lo que provocó que al crear más .meta me corrompiera el proyecto ya que hay que ignorar el seguimiento de este tipo de carpetas en Unity. Finalmente lo subí a GitHub sin esa carpeta, ya que la generar al importar el proyecto.

5.2. Pruebas

5.2.1. Pruebas Unitarias

5.2.2. Pruebas Funcionales

En este punto pondré el requisito funcional al que pertenece la prueba (de los que puse al principio) y lo marcare como probado y validado. Adjuntare en drive en la entrega de documentación una carpeta llamada 'pruebas/' que contendrá clips con los nombres según el tipo de requisito funcional (si necesita video, si no

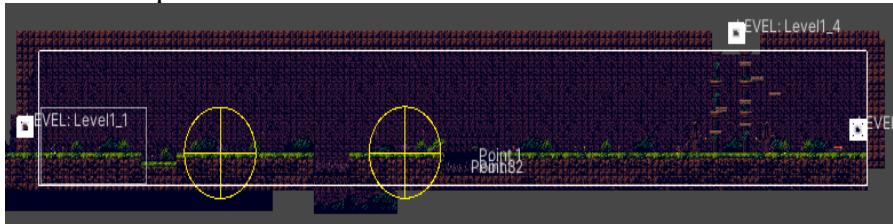
en una imagen aquí), ej RF1.video para comprobar en un clip que funciona correctamente. Debido a la naturaleza de mi proyecto he tenido que hacerlo con videos. En los que es conveniente intervengo hablando con la explicación pertinente, que es en el 99% de ellos. Interesante ver RF15, RF20, RF29

- RF1Assets -> Sí.



El estilo de los assets es correcto.

- RF2 Timelaps -> Sí



Las escenas estan limitadas.

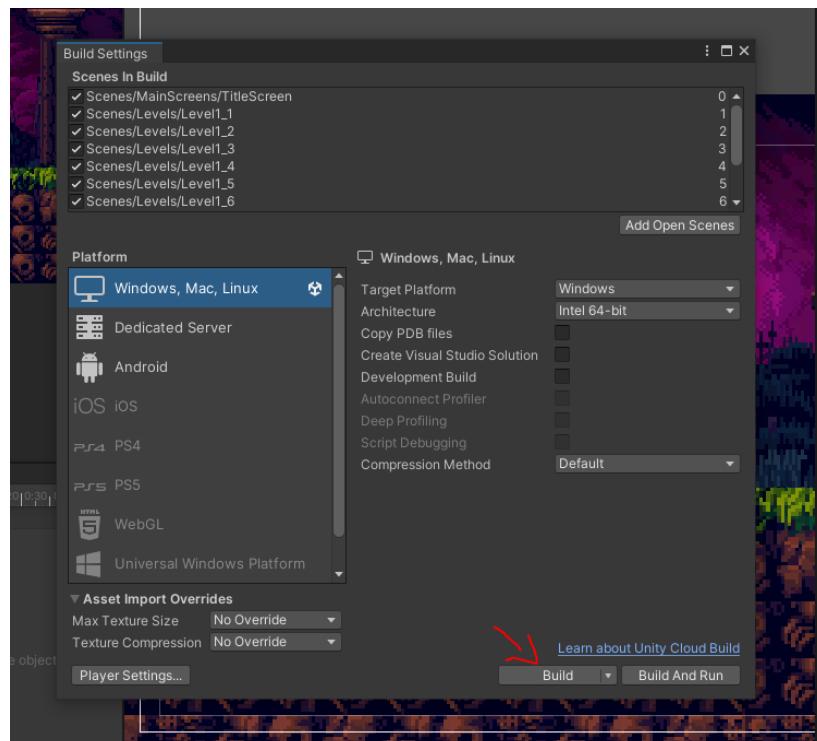
- RF3 Movimiento Héroe. (Si no hay imagen esta en video en la carpeta videos/ de la docu entregada) -> Sí
- RF4 Salto Heroe. -> Sí
- RF5 Detectar Colisiones Entorno Heroe. -> Sí
- RF6 Detectar Colisiones Daño Heroe. -> Sí
- RF7 Detectar Colisiones Daño Arma Heroe. -> Sí
- RF8 FeedBackDaño Héroe -> Sí (Se comprueba en el video de RF6)
- RF9 Animaciones Héroe -> Sí
- RF10 Sonido Héroe -> Sí, Se puede apreciar en los demás videos.
- RF11 Parámetros Héroe -> Se puede apreciar en los demás videos.
- RF12 Girar Héroe -> Se puede apreciar en los demás videos.
- RF13 Persistir Heroe-> Funciona, pero bajo x condiciones tiene algunos bugs, debo fixearlo. He detectado el problema y se debe a que creo instancias encima de la que ya tiene el héroe, entonces reinicia sus parámetros cargándolos de cero. Debido a la complejidad de mi código arreglar este bug requiere de tiempo así que intentare tenerlo al 100% para la presentación. Solucionado con patron singleton actual.
- RF14 Programar Enemigos General -> Sí, se puede apreciar en los videos de los enemigos.
- RF15 Enemigo Fantasma -> Sí.
- RF16 Enemigo Esqueleto -> Sí.
- RF17 Enemigo Bestia -> Sí.
- RF18 Comunicación Escenas -> Sí
- RF19 Suelo Dañino -> Sí
- RF20 Plataformas Móviles -> Sí

- RF21 Magias -> En principio había dejado de funcionar así que he tenido que fijearlo. Ahora Funciona.
- RF22 Coleccionables -> Sí
- RF23 Boss Final -> Sí, se aprecia en los videos siguientes.
- RF24 Estados Boss Final -> Sí, se aprecia en los videos siguientes.
- RF25 Presentación Boss Final -> Sí
- RF26 Teletransporte Boss Final -> Sí
- RF27 Movimiento Boss Final -> Sí
- RF28 Lanzamiento Proyectiles Boss -> Se aprecia en los otros videos, Sí
- RF29 Fase Final Boss -> Sí
- RF30 Muerte Jugador -> Sí, pero tuve problemas debido a que a la segunda muerte no mostraba la pantalla de muerte o no era interactiva.
- RF31 Pantalla de Inicio -> Sí
- RF32 Pantalla de GameOver -> Sí
- RF33 Pantalla Guardar Partida -> Sí
- RF34 Persistencia de los Datos -> Sí, se demuestra en Guardar Partida y la carga en Pantalla de Inicio.
- RF35 Programar UI -> Sí
- RF36 Cámara Móvil -> Sí, se puede apreciar a lo largo de los videos.

5.3. Instalación/Despliegue y Configuración

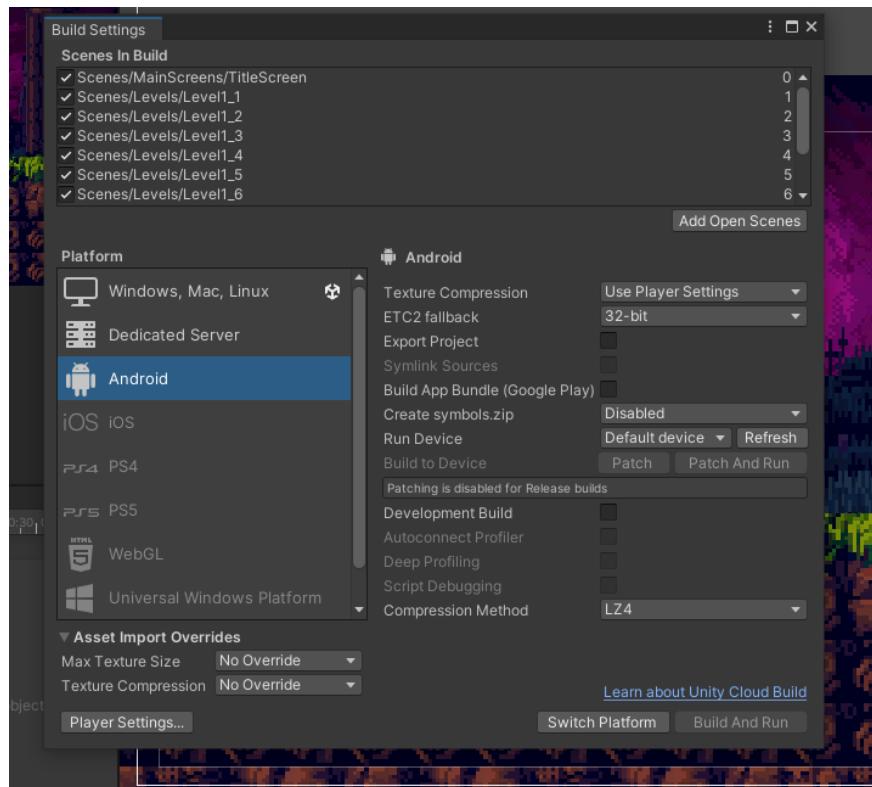
Se adjuntan en la entrega de documentación archivo de instalación para Windows y apk para móvil. No necesitan de configuraciones especiales.

Para generar el de Windows:



Este equipo > Disco local (C:) > built				
Nombre	Fecha de modificación	Tipo	Tamaño	
MonoBleedingEdge	02/06/2022 22:42	Carpeta de archivos		
randomly_BurstDebugInformation_DoNo...	02/06/2022 22:42	Carpeta de archivos		
randomly_Data	02/06/2022 22:42	Carpeta de archivos		
randomly.exe	02/06/2022 22:42	Aplicación	639 KB	X
UnityCrashHandler64.exe	02/06/2022 22:42	Aplicación	1.099 KB	
UnityPlayer.dll	02/06/2022 22:42	Extensión de la ap...	28.163 KB	
victorpenna.txt	02/06/2022 22:42	Documento de te...	0 KB	

Para generar el de Android (alterando los controles):



Este equipo > Disco local (C:) > built

Nombre	Fecha de modificación	Tipo	Tamaño
randomly_BurstDebugInformation_DoNo...	02/06/2022 23:00	Carpeta de archivos	
randomly.apk	02/06/2022 23:01	Archivo APK	78.251 KB

5.4. Manual de Usuario

En este punto debido a las explicaciones de todo que ya he hecho y a que es un juego y realmente no hay que conocer mucha cosa como usuario, explicare la finalidad y lo que debes de hacer en las UI. Y diré los controles para jugar.

Primero empiezas en la pantalla de inicio, aquí debes de prestar enter.

Una vez hecho esto se te muestran distintas opciones, new game, continuar y salir. Continuar solo saldrá en caso de que tengas ya una partida guardada.

Si empiezas partida empezaras en la escena 1 y deberás elegir la ruta para seguir avanzando entre las escenas, las cuales al final de cada escena te trasladara a la siguiente escena.

Si en algún punto de el trayecto de avanzar mueres, te saldrá la interfaz de Game Over y deberás elegir si reiniciar el nivel donde has muerto o si salir al menú principal.

A lo largo del mapa hay dos salas de guardado, en estas salas si te acercas al fuego azul central podrás guardar, para continuar desde aquí cuando quieras.

En el propio juego veras un cuadrado, unos corazones y una moneda. Si los corazones llegan a cero mueres, las monedas representan las que has cogido y en el cuadrado aparecerá la magia actual que poseas y la cantidad de la que dispones.

Explicado un poco el concepto de como jugar, explicare los controles.

En Windows, alt -> Lanza Magias, Espacio -> Saltar, A -> Movimiento a la izq, D-> Mov A la Derecha, Clic -> Atacar

En Móvil salen los típicos paneles de los juegos para moverte y realizar acciones.

6. Documento de cierre

6.1. Resultados obtenidos y conclusiones

Los resultados obtenidos en general han sido muy satisfactorios. Siendo sincero he identificado algunos bugs que salen en ciertas ocasiones, pero debido a la complejidad del juego tengo que fixearlos todavía. He conseguido identificar que se trata de que crea copias del personaje en la escena y pisa la anterior del personaje, reiniciando sus parámetros entre escenas. Luego la pantalla de morir a veces crea clones de ella y al tener varias instancias no sale nada al morir y tienes que cerrar y volver a abrir. He estado buscando información y no es problema de lógica, si no que en la versión de unity que uso (la más reciente) debo implementar el patrón singleton de otra forma, pero al ser tan reciente hay muy poca información al respecto y la gente tiene muchos problemas con ello. De todas maneras, intentare que este solucionado para la presentación.

En general estoy muy contento con el resultado y siendo sincero, lo más normal en un juego que está en fase Alpha es que tenga mil y un bug. De normal los juegos se hacen entre muchísimas personas, pasan una Alpha y los usuarios encuentran los bugs, identificándolos y subsanándolos con el paso del tiempo.

En mi caso al estar probándola yo solo, expongo al personaje a menos situaciones de las que expondría si hubiese jugando 1000 personas, así que seguramente haya bugs sin identificar.

Aunque comente esto, el resultado me parece increíble y estoy muy muy contento con todo lo que he aprendido de Unity y de C#.

He tenido que dedicar muchas horas a la formación para poder hacer el proyecto y digamos que era arriesgado debido a los pocos conocimientos que tenía.

En cuanto al diseño del boss final estoy contentísimo con el resultado. No presenta bugs y tiene una IA bastante interesante.

Respecto a la dificultad, menos la parte final que me parece un poco complicada, es bastante asequible.

La verdad que pensaba abarcar más temas en el proyecto, pero al ver que usaba un montón de cosas del curso y la cantidad de tiempo que he dedicado a ello, me quede con menos cosas. Unity y su forma de trabajar es totalmente orientada objetos e incluso el ide tiene esa filosofía, tienes que orientar a objetos tanto en el código como en el ide, así que se puede decir que ese punto lo he cumplido con creces.

Respecto a la base de datos, tengo una en la pagina web, que esta un poco por cumplir el objetivo de la base de datos, pero realmente no me sentí muy agusto implementándola al proyecto.

Dicho esto, estoy muy contento con el proyecto y sobre todo con los conocimientos que me llevo aprendidos, aunque me haya tenido que tirar de los pelos y frustrar en muchos momentos cuando no salían las cosas.

Comentar que hay un nivel secreto que no he colocado ni documentado en el proyecto, por eso es nivel secreto, con el verdadero boss final del juego.

6.2. Diario de bitácora

7. Bibliografía

C#:

[Documentos de C#: inicio, tutoriales y referencias. | Microsoft Docs](#)

Recursos Unity:

[Unity Asset Store - The Best Assets for Game Making](#)[Gothicvania Patreon's Collection - YouTube](#)[Game Art 2D - Royalty Free 2D Game Assets](#)

Visual Studio:

[Visual Studio: IDE y Editor de código para desarrolladores de software y Teams \(microsoft.com\)](#)[Aregar diagramas de clases a proyectos \(Diseñador de clases\) - Visual Studio \(Windows\) | Microsoft Docs](#)[Visual Studio Tip #8 - How to Create a UML Class Diagram #Shorts - YouTube](#)[Is there a Class diagram? : Unity3D \(reddit.com\)](#)[Generate class diagram from code - Unity Forum](#)

Unity:

[Generar ejecutable de tu Videojuego con Unity/Sacar .exe o .apk para PC o Android/Compilar - YouTube](#)[¡Divírtete mientras aprendes a programar! | Programación de juegos en c# para principiantes | Unity](#)[Unity - Manual: Creando y usando scripts \(unity3d.com\)](#)[Plataforma de desarrollo en tiempo real de Unity | Motor de VR y AR en 3D y 2D](#)

Gantt:

[Asana](#)[Gothicvania Patreon's Collection by ansimuz \(itch.io\)](#) -> De aquí he sacado assets.

Creditos a ansimuz debido a que he sacado bastantes imágenes para poder hacer el juego de sus recursos disponibles. No se dibujar ni hacer diseño3d y encontrar sus diseños me ha supuesto no presentar un churro de diseño de personajes y aspecto del mapa. Eso no quita que los mapas no los haya diseñado yo, pero si que es verdad que he usado parte de sus edificios y elementos. De normal se suele contratar a un artista y el programador hace la parte lógica.

Especiales thankius -> [ansimuz - itch.io](#)



8. Anexos
