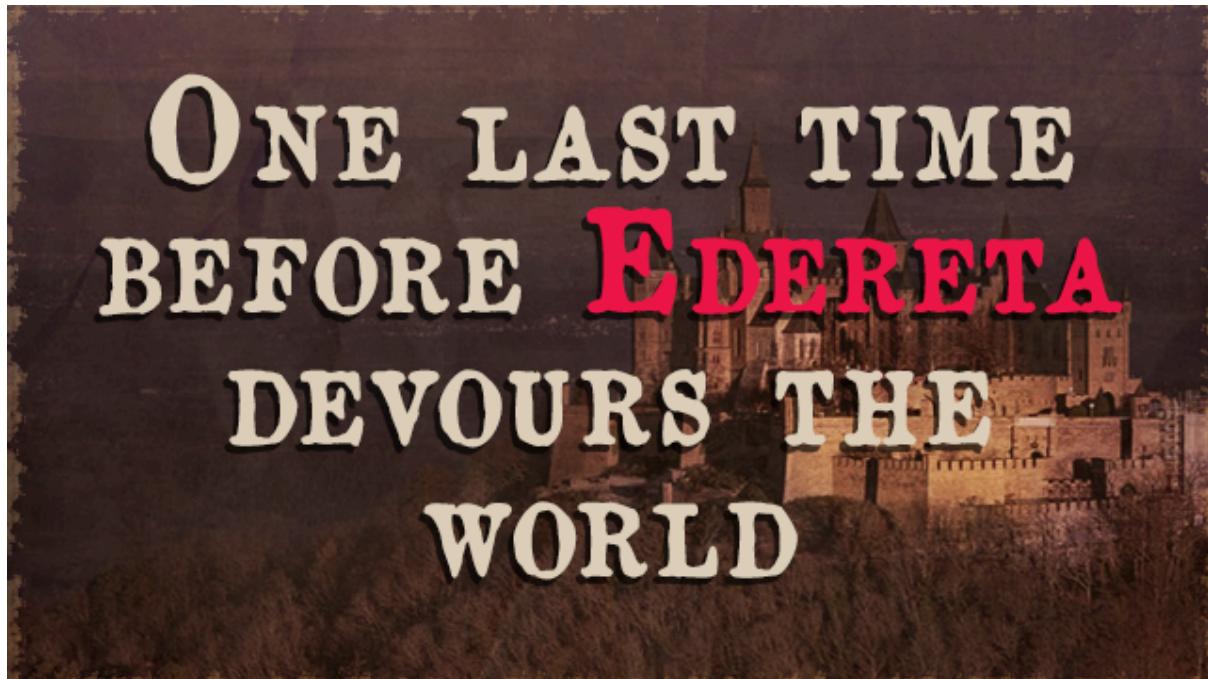


Proyecto fin de grado
I.E.S San Juan de la Cruz



One last time before Edereta devours the world

-
CRPG Roguelike 2D en Unity

Alumno:
Fernando Tarrio del Pozo

Ciclo:
Desarrollo de aplicaciones multiplataforma - 2023-2024

Fecha de entrega:
12 de junio de 2024

ÍNDICE

ÍNDICE	1
1.- Introducción	2
2.- Objetivos	2
3.- Metodología	3
4.- Análisis y diseño de la base de datos	5
4.1.- Introducción	5
4.2.- Objetivos de la base de datos	6
4.3.- Modelo entidad/relación (E/R)	6
4.4.- Modelo relacional	8
4.5.- Datos en CSV	8
5.- Desarrollo e implementación	9
5.1.- Introducción	9
5.2.- Funcionalidades	9
5.3.- Desarrollo	10
5.3.1- Estadísticas	11
5.3.2- Proceduralidad	12
5.3.3- Estados y eventos	14
5.3.4- Inventario y exploración	14
5.3.5- Finales	16
6.- Pruebas	16
7.- Aportaciones originales	16
8.- Conclusiones, metas, dificultades y mejoras	16
8.1.- Metas alcanzadas	17
8.2.- Dificultades encontradas	17
8.3.- Mejoras posibles	18
9.- Bibliografía y enlaces	19
10.- Anexos	20

1.- Introducción

El proyecto gira en torno a la creación de un juego completo en 2D (2d simulado en un entorno 3D) usando el motor de creación y edición de juegos Unity[2]. El juego consta de un ciclo de vida completo permitiendo crear una partida nueva, jugar y finalizar la partida.

El juego será de corte roguelike con permadeath y pretende emular a clásicos antiguos como Última[3] o Eye of the Beholder[4]. Este estilo de juegos es denominado CRPG (computer role playing game)[5].

Todos los recursos empleados en el desarrollo del proyecto serán creados por el alumno, destacando entre muchos otros los recursos gráficos. Para los cuales se emplea el programa de dibujo profesional en pixel art Aseprite[6] donde se llevará a cabo el desarrollo y proceso artístico referente al diseño y animación de tiles / tilesets, personaje e interfaz. Se emplearán otros programas variados para diferentes aspectos como puede ser el apartado sonoro (donde se usará Audacity[7]) o testing. También se cuenta con la ayuda / colaboración de personal externo al proyecto para algunos assets cedidos como pueden ser los bocetos de la interfaz desarrollados por Irene Lloret[8] o el uso de assets gratuitos de Kenney[9]. Todo debidamente documentado y acreditado en una lista completa que se puede encontrar en la documentación que se ha ido realizando del proyecto durante los meses de desarrollo junto a todos los recursos públicos del proyecto creados por el alumno que se dejan a disposición de cualquiera que quiera consultarlos [10].

2.- Objetivos

Se pretende poner a prueba la capacidad del alumno para llevar a cabo un proyecto en su totalidad pasando por las diferentes fases de análisis, diseño, desarrollo, testing y otras. De esta forma se prueba que el alumno ha comprendido e interiorizado todo lo aprendido en el curso y hasta donde llegan sus conocimientos. También se quiere demostrar la capacidad de organización, anticipación y preparación que tiene el alumno para afrontar una tarea compleja como es un desarrollo completo en un tiempo finito (y corto) como es el estipulado por el periodo en que se tiene acordado el desarrollo a su vez que este desarrollo es compaginado con otras actividades (en el caso de la prueba, con la jornada laboral de la formación en centro de trabajo).

La aplicación final debe cumplir varios objetivos, siendo estos:

1. Multi-idioma (inglés y español)
2. Multi-plataforma (Windows y Android al menos)
3. Uso de base de datos con posibilidad escalable a entorno distribuido/online
4. Flujo completo de juego y persistencia de datos entre partidas (sin savegame, con estadísticas persistentes)
5. Posibilidad de completar el ciclo de juego (iniciar programa, crear partida, completar partida, cerrar programa)
6. Lista de especificaciones y requisitos básicos que debe tener un videojuego, interfaz, menú con opciones, música y posibilidad de silenciarla, reinicio de datos, créditos finales, entre otros.

3.- Metodología

Debido a que el proyecto será desarrollado enteramente por una sola persona, al margen de que entidades externas puedan contribuir en diferentes partes del proceso como puede ser en la fase de testing o en el uso de assets de terceros, la metodología que se emplea para el desarrollo de la aplicación será una metodología **en cascada**.

La metodología en cascada se particulariza en que una fase del desarrollo no se completa hasta acabar la anterior. Gracias a esto será posible controlar mejor posibles fallos en etapas futuras y asegurar que el proyecto avanza a buen ritmo permitiendo así cumplir con las fechas previstas.

Las fases en que se divide el proyecto son las siguientes:

1. Fase de análisis inicial:

Durante esta fase se estudia la viabilidad del proyecto así como los márgenes que debe cumplir, entendiendo con ello que se fijan unos objetivos concretos a los cuales debe llegar la aplicación en mayor o menor medida teniendo en cuenta cuáles de ellos pueden ser sacrificados en pos de otros. Este proceso está documentado y puede encontrarse en la carpeta de recursos públicos [10].

2. Fase de diseño:

Fase que ocupará un gran porcentaje inicial del desarrollo. Se pretende

enmarcar los límites jugables así como las mecánicas del programa. Qué tipo de juego será, como se llevará a cabo la interacción de sus componentes y cuales son (inventario, mapamundi, gestión de acciones, interacción con entorno, otros). Se realizarán pequeñas iteraciones dentro de la fase de diseño para crear gráficos temporales y probar sistema de resoluciones dando así paso a una estandarización en los gráficos que vamos a usar (paleta de colores, shaders gráficos, dimensiones de los tiles y tilesets, cantidad de frames de las animaciones, entre otros). Durante esta fase se realizará un diseño inicial a modo de boceto de la base de datos.

3. Fase de prueba de concepto:

Se realizará una prueba de concepto que cubre, con datos básicos y sencillos, todo lo que debe alcanzar el juego final. Esté proceso está documentado y puede encontrarse en la carpeta de recursos públicos [10]. Gracias a este proceso se pueden detectar bloqueos / errores en diseño o código a fin de solucionarlos antes de empezar el desarrollo final y así ahorrar tiempo y recursos.

4. Fase de desarrollo:

Fase principal del proyecto, una vez demostrada la viabilidad gracias a la prueba de concepto, se pasará al desarrollo de la aplicación final. En un nuevo proyecto de Unity y con todo lo aprendido será posible llevar a cabo todos los objetivos definidos en el marco inicial. Se diseñarán assets finales para la interfaz, mapa, animaciones, estructura de carpetas, addons y scripts de terceros así como la implementación final de shaders e interacción con APIs externas.

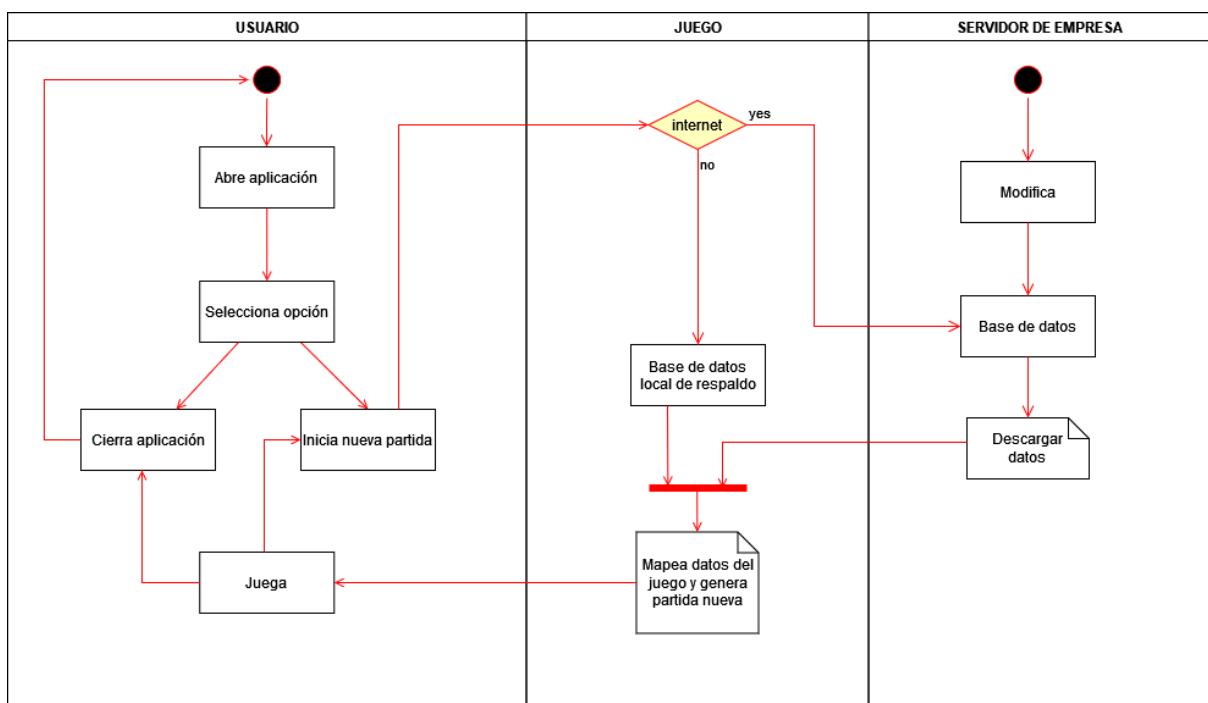
5. Fase de pruebas:

Si bien la fase de pruebas también se realiza muy ligeramente durante la fase de prueba de concepto, una vez terminada la aplicación definitiva se someterá dicha aplicación a diferentes pruebas tanto test unitarios como pruebas ciegas con personal totalmente ajeno al desarrollo. Cabe resaltar que debido al poco margen que se dispone a la hora de desarrollar la aplicación (3 meses para todas las fases), algunos aspectos de la fase de pruebas no pueden llevarse a cabo. Estas fases, lamentablemente, son de vital importancia en el desarrollo de un videojuego pero la suma de tiempo y falta de recursos por parte del alumno hacen imposible su realización. Estas pruebas son, por ejemplo, la prueba en diferentes hardwares o casuística de configuración de equipos. Un videojuego es un programa que depende de muchos factores (renderización de gráfica, acceso a memoria virtual, acceso a ficheros del sistema operativo, entre otros) y cada equipo es un mundo, el testing se ha realizado en un entorno lo más amplio posible pero nunca se puede garantizar que esté terminado. Un ejemplo de ello son las grandes compañías que lanzan juegos al mercado y en el día de salida dicho juego da error en un sinfín de equipos. Incluso ellos, que cuentan con recursos ilimitados a la hora de realizar esta fase de testeо, se quedan cortos.

4.- Análisis y diseño de la base de datos

4.1.- Introducción

La base de datos es un aspecto crucial del juego a desarrollar. En lugar de disponer de una base de datos al uso que se encuentra detrás de la aplicación, el proyecto emulará una base de datos distribuida que se encuentra controlada por la empresa tras el juego. Cuando el jugador empieza una partida nueva se conecta a la base de datos online y descarga la información del juego (objetos, lugares, descripciones, etc.) para posteriormente mapear dicha información en objetos de C# (el lenguaje usado por Unity) y crear el escenario.



Para explicarlo mejor se ha creado un diagrama de caso de uso. El usuario iniciará la aplicación y pedirá al juego una partida nueva. En caso de tener internet, el juego se conectará a la base de datos de la empresa y descargará toda la información de la base de datos. En caso de no tener internet, existe una base de datos local de respaldo con datos básicos para poder iniciar una partida y acceder al juego. Este diseño permite a la empresa modificar la base de datos sin notificar al usuario, de una forma totalmente abstracta para el programa. Esto se traduce en una facilidad a la hora de “actualizar” el contenido del juego, siendo posible para la empresa modificar las tablas sin obligar al usuario a descargar un

parche y cuando el jugador inicie una partida nueva dispondrá de todos los datos actualizados sin saberlo.

4.2.- Objetivos de la base de datos



La base de datos se diseña orientada a un propósito sencillo, ser capaz de almacenar grandes cantidades de información de la forma más resumida posible. Mientras que otras bases de datos más al uso tienden a segmentar la información en muchas tablas, el objetivo de nuestro modelo debe ser la rapidez de acceso a los datos y la posibilidad de manejar grandes bloques de texto. Es por ello que durante la fase de análisis inicial se realiza una extensa investigación sobre sistemas gestores de bases de datos y así poder elegir con mayor precisión cuál es la que mejor se adapta al proyecto. Esté proceso está documentado y puede encontrarse en la carpeta de recursos públicos [10]. Finalizado el análisis y teniendo en cuenta los objetivos para la base de datos, se decide usar **SQLite** tanto en distribuido como en local. De cara a la defensa del proyecto **se realizará todo con la base de datos local de respaldo** de forma que sea más sencillo de presentar tanto en cuestión de tiempos de carga como de casuística y problemática con los entornos de red.

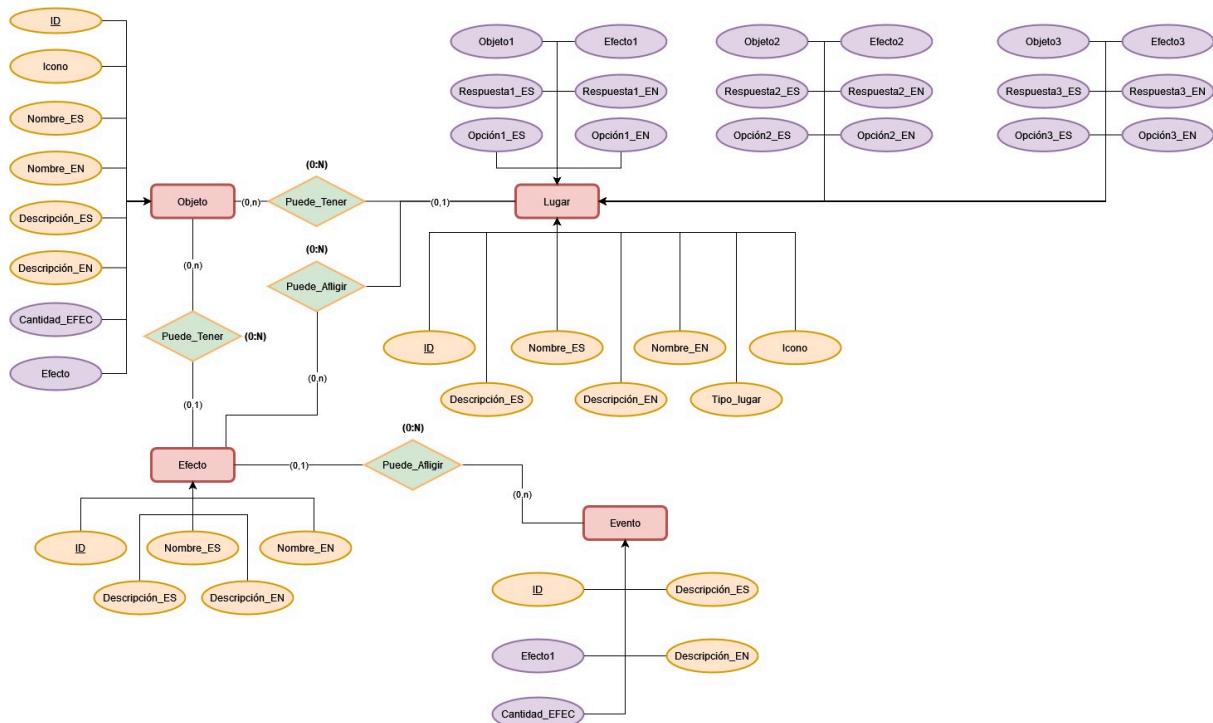
4.3.- Modelo entidad/relación (E/R)

Gracias a la realización de la prueba de concepto y una base de datos temporal (consultar documentos en recursos públicos [10]) que ha valido al alumno como boceto del proyecto se han podido dibujar mejor los límites y funcionalidades de la base de datos que se usará en el proyecto final.

Si bien la base de datos usada tiene un propósito sencillo anteriormente descrito (guardar datos extensos en el menor espacio posible) puede parecer en un primer vistazo que tenemos demasiados datos en tablas que pueden segmentarse (como vemos en la tabla “lugar”). En primera instancia podemos pensar en separar esta tabla y añadir tablas intermedias (objetos_en_lugar, decision_en_lugar, otros) pero gracias al testing realizado en la prueba de concepto con la base de datos provisional se ha detectado que, en cuestión de conexiones online, integridad de datos y facilidad de comprensión de datos tanto desde un perfil humano

como de código, este formato es más sencillo y apto para las funcionalidades del proyecto en el estado actual.

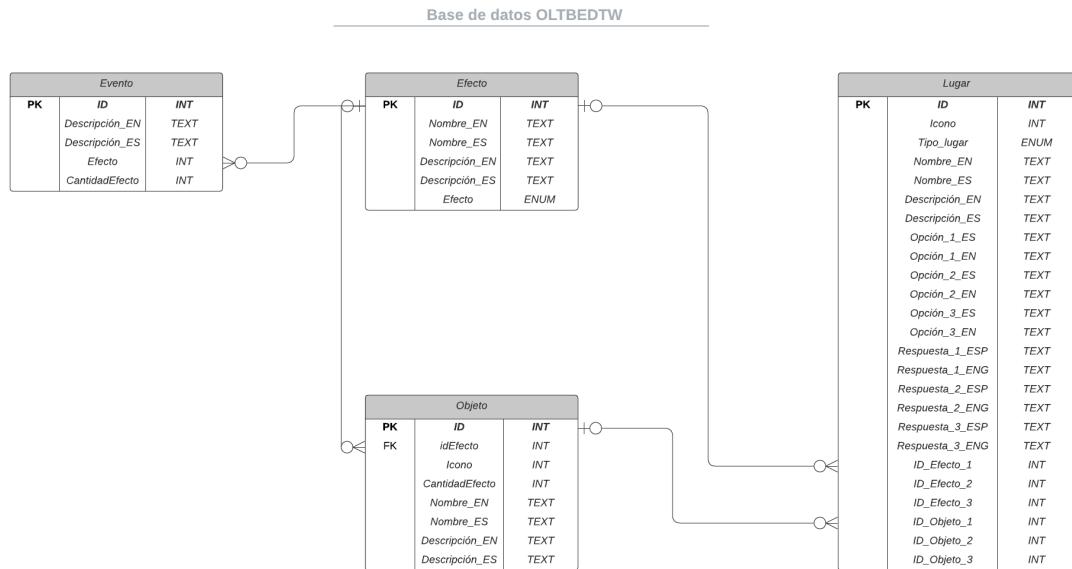
Esto no quiere decir que sea la mejor forma de organizarlo o que no existan otras, en la sección de conclusiones se hablará de esto un poco más a fondo. Si no que, debido al poco tiempo que se tiene y los límites actuales del proyecto, es el diseño que se ha decidido emplear de forma que permite simplificar conexiones y rebajar casuística de errores tanto de integridad como de funcionalidad general.



La base de datos estará al servicio de los objetos del juego y no al contrario. Esto quiere decir que la base de datos debe ser capaz de guardar de forma flexible y clara todo lo que nuestro juego usa. Dispondremos de una tabla “base” con los **efectos** que pueden ocurrir en el juego. A su vez, los **objetos** y **eventos** con los que interactúa el jugador de diferentes formas guardan, a parte de datos propios, relación con los efectos. Y por último tenemos los **lugares**, piedra central de la interacción jugador / juego. Los lugares guardan amplia información sobre cadenas de texto que cuentan trasfondo del mundo y relacionan los datos con las tablas de objetos y efectos. Estas relaciones las entenderá el código y recuperará de la base de datos para mapear la información a objetos de C# que luego el juego usará para generar una cadena lógica de acciones e interacciones que el jugador podrá usar durante su partida.

4.4.- Modelo relacional

El modelo relacional nos permite ver de una forma mucho más sencilla la arquitectura que tendrá la base de datos así como sus relaciones. Este formato es una gran ayuda a la hora de orientar el código así como el diseño de las clases y objetos que Unity deberá gestionar.



4.5.- Datos en CSV

Gracias a las facilidades de conexión que permite SQLite y Unity, todos los datos serán añadidos mediante código mapeando información de diferentes CSV. Si bien este paso en el producto final será transparente para el usuario dado que lo hará la empresa que se encarga de la gestión de la base de datos, en el momento de la defensa lo hace el programa para poder demostrar que todo se realiza correctamente. Este formato y forma de trabajar es algo estandarizado y comúnmente extendido en la creación de videojuegos.

	color	column 2	column 3	column 4	column 5	column 6	column 7
1	Icon	Name_ESP	Name_ENG	Desc_ESP	Desc_ENG	Effect	Amount
2	0	Pescado deshidratado	Dehydrated fish	Al llegar por primera vez a este lugar, Kara y Elbura intentaron repoblar las masas de agua que encontraron al noroeste de la meseta con algunos peces parecidos a los que tenían en su hogar. El resultado fue ligeramente satisfactorio, por desgracia no fueron capaces de replicar el sabor a la perfección.	Upon first arriving here, Kara and Elbura attempted to restock the bodies of water they found northwest of the plateau with some fish similar to the ones they had at home. The result was mildly satisfactory, unfortunately they were not able to replicate the taste perfectly.	6	2
3	1	Queso algo pasado	Cheese a little overdone	Caciero siempre fue un maestro quesero, por desgracia fue el primero en separarse del grupo. Edereta intentó emular su técnica y fermentar la leche lo mejor posible. No siempre lo conseguía con éxito.	Caciero was always a master cheesemaker, unfortunately he was the first to break away from the group. Edereta tried to emulate his technique and ferment the milk as well as possible. She was not always successful.	6	1
4	2	Conservas	Homemade	If hay algo que Anunia echa de menos	If there is one thing Anunia misses from her childhood, it's probably the taste of the canned goods she grew up with.	6	3

5.- Desarrollo e implementación

5.1.- Introducción

Una vez se han completado las etapas de análisis inicial, diseño y prueba de concepto se empezará con la etapa del desarrollo final. En esta etapa se programará un proyecto completo en Unity que corresponde al juego a desarrollar. Dicho juego usará recursos gráficos definitivos y tendrá diferentes funcionalidades para justificar todo el trabajo realizado.



Menú inicial del juego en la versión que se usará para la defensa (0.9.2) - Junio 2024

5.2.- Funcionalidades

Las funcionalidades que debe cumplir el juego final son:

1. Menú inicial
2. Menú de opciones accesible desde menú inicial y desde el juego
3. Estadísticas persistentes y control sobre ellas (reiniciar estadísticas)
4. Iniciar nueva partida con generación semi-procedural del mapa
5. Dentro de la partida: Sistema de registro textual
6. Dentro de la partida: Sistema de interacción con el entorno
7. Dentro de la partida: Sistema de inventario
8. Dentro de la partida: Varios finales y ramificaciones en la partida

9. Dentro de la partida: Sistema de elección al explorar
10. Dentro de la partida: Sistema de propagación de entidades nocivas para el jugador a modo de presentar amenazas por el mapa
11. El juego debe poder ejecutarse en varias plataformas siendo las plataformas objetivo Windows y Android

5.3.- Desarrollo

Mediante código se han generado diferentes algoritmos que permiten una generación semi-procedural del mapa. Esto, junto a la cantidad de elecciones que disponen los lugares que encontramos, permiten una variación en las partidas que crean interés a la hora de jugar.

Al iniciar una partida nueva los lugares se posicionarán de forma aleatorizada por el mapa y el jugador deberá decidir que explorar y cuando, optimizando sus recursos para poder llegar al final de la partida.



Estado de la partida al ser iniciada en la versión 0.9.2 - Junio 2024

Desde el editor de Unity tendremos diferentes modos de lanzar el juego. Estos modos también han sido creados en nuestro código. Existe la posibilidad de lanzar el juego en modo **debug** de forma que genera un mapa completo a pintar en pantalla (lo cual afecta bastante al

rendimiento) y añade todos los objetos existentes al inventario del jugador. Con esto podemos realizar diferentes pruebas de integridad de datos y realizar testing.

En caso de lanzar el juego sin modo debug, se generará un mapa de X por Y, por defecto 50x50, y se rellenará de forma aleatorizada. Solo se dibuja en pantalla lo que el jugador ve (algoritmo de renderizado) lo cual permite generar mapas de dimensiones personalizadas. Pudiendo con ello crear un escenario pequeño como el mostrado durante la defensa o un escenario gigante, el cual es el objetivo del juego final cuando se dispongan de más datos para llenar contenido.

En esta etapa se genera el código y programa toda la lógica de acciones del juego, debido a la limitación de máximo 20 págs para el documento es imposible argumentar aquí todo lo que se ha trabajado, este documento se apoya en la presentación que se puede encontrar junto al resto de recursos [10] para mostrar como funciona a grandes rasgos el programa. Aún con ello, se procede a mostrar brevemente los apartados del desarrollo.

5.3.1- Estadísticas

Las partidas del juego diseñado no disponen de opción de guardado, esto es algo inherente de muchos títulos de corte roguelike¹. Esto quiere decir que el ciclo de juego de iniciar una partida y terminarla debe ser completado de una sola vez. No es posible guardar la partida y continuar más tarde. Es algo intencionado, esta mecánica de juego impuesta al jugador hace que las decisiones cobren peso y tenga que gestionar correctamente sus acciones e inventario para poder llegar al final de la partida. A su vez, nos genera una tensión entre la exploración y el recorrido que nos falta para alcanzar la meta. Quizás el jugador quiera explorar más y entender el mundo del juego pero esto pueda terminar irremediablemente con su partida antes de tiempo.

Para compensar esto, algo también ampliamente extendido dentro de los CRPG de diferentes cortes, tanto roguelike como otros subgéneros, es la progresión en segundo plano de estadísticas generales entre partidas.

¹ Consultar el anexo [1] si se busca informarse más sobre diferentes CRPG y sus temáticas así como mecánicas y ejemplos. Dicho enlace nos lleva a un proyecto comunitario llamado “The CRPG Book” donde más de 150 personas han colaborado a lo largo de los años para documentar y extender los diferentes CRPG que han existido desde sus inicios con PLATO hasta la fecha.

Las estadísticas generales llevan la cuenta acumulativa de todo lo que hemos hecho durante las diferentes partidas que hemos jugado (comúnmente denominadas “run” dentro del lenguaje urbano del videojuego). Gracias a este panel podemos consultar, por ejemplo, los pasos que hemos dado, objetos que hemos consumido y la cantidad de veces que hemos llegado al final de forma satisfactoria. Esta opción se integra, no solo desde una perspectiva jugable, si no también para mostrar que la aplicación tiene la posibilidad de generar datos persistentes entre sesiones. Estos datos se almacenan incluso después de cerrar el juego y se recuperan al abrirlo de nuevo de forma local en la máquina del jugador.



Panel de estadísticas en la versión 0.9.2 - Junio 2024

5.3.2- Proceduralidad

En términos sencillos, consideramos que el juego transcurre sobre una parrilla (**grid**) de X x Y en dos dimensiones la cual poblará nuestro mapa con diferentes losetas (**tiles**) para generar el mapa donde el jugador se va a mover.

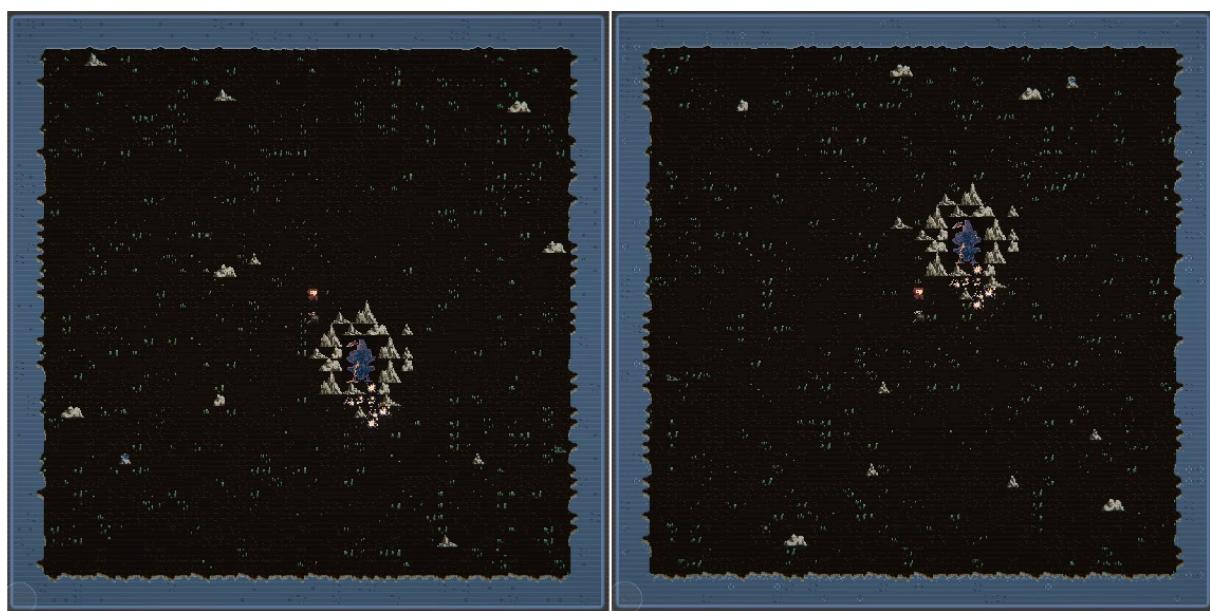
Está grid, al generar el escenario creando una nueva partida, se llenará siguiendo unos algoritmos y patrones dictados en el código. Primero, el código dibujará los bordes del mapa de forma que las 3 casillas más adyacentes a un borde siempre serán agua y precipicio. Un

tipo de casilla por la que el jugador no puede moverse y así prevenir que el personaje salga del marco jugable.

A continuación se generan las posiciones del inicio de jugador, casa del jugador y posición del final del juego. El final del juego es la ubicación llamada “Stillness I” y su emplazamiento será siempre a 20 casillas del borde en dirección interior partiendo de una de las cuatro esquinas del mapa. Esto quiere decir que el final no siempre estará en el mismo lugar.

A continuación se pintan los denominados “espacios vacíos y válidos”, rellenando dichos espacios con una mezcla aleatoria de tiles de tierra o hierba animada. Estos tiles permiten situar la pantalla mejor al mover el personaje. Una vez el mapa se ha completado, se genera una cantidad equivalente al largo x ancho del mapa entre cinco (**width x height**) / 5) y se pinta esa cantidad de montañas para dar más variedad al entorno.

Una vez acabado el mapa se itera sobre la lista de lugares obtenida de la base de datos y se dibujan los lugares en emplazamientos semi-aleatorios a lo largo del mapa siguiendo unos algoritmos sencillos de ubicación (distancia unos de otros, masas del mismo tipo etc). En un futuro se mejorarán estos algoritmos para crear emplazamientos más dinámicos.

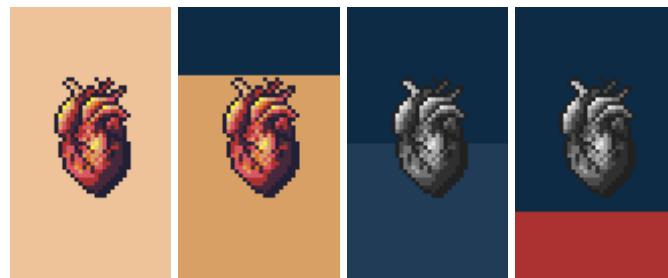


Ejemplo de dos mapas generados con la base de datos de respaldo en la versión 0.9.2 - Junio 2024

5.3.3- Estados y eventos

Los estados son el principal motor del juego. Disponemos de cinco estados. Vida, Cordura, Hambre, Descanso y Acciones. Los cuatro primeros nos indican cómo se encuentra la protagonista de la aventura y si cualquiera de ellos llega a 0 la partida terminará en derrota. El quinto estado indica cuantas acciones nos quedan por realizar (andar, interactuar, otros).

Una vez nos quedamos sin acciones debemos descansar, al realizar esta acción gratuita se sucederá un evento aleatorio (mapeado desde la base de datos) que, normalmente, penalizará una de las estadísticas con su consciente descenso y posibilidad de terminar el run. Por ello el jugador debe explorar y encontrar objetos con los que recuperar estadísticas y así llegar al final del recorrido.



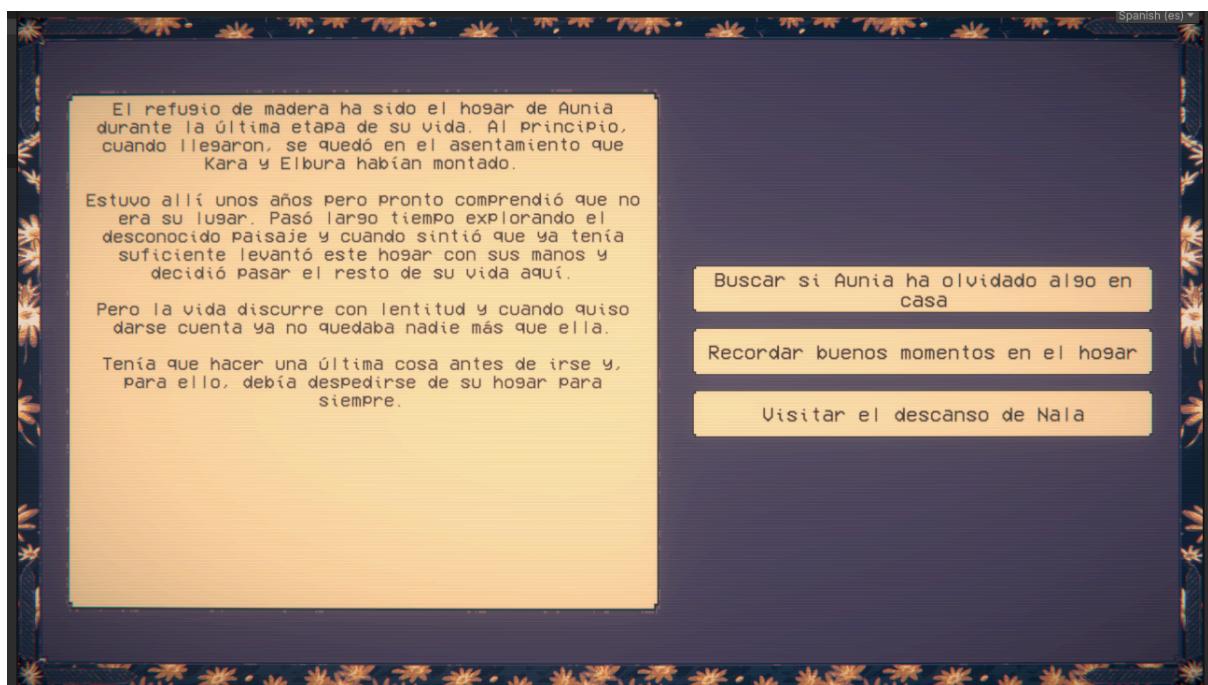
Ejemplo de las cuatro variaciones del estado “Vida / Health” - Junio 2024

5.3.4- Inventario y exploración

Al explorar el mundo diseñado en el videojuego nuestro personaje podrá interactuar con el entorno, descubriendo así nuevos lugares que le brindan una pequeña descripción de lo que ha encontrado junto a un número finito de decisiones entre una y tres.

Es trabajo del jugador leer detenidamente las opciones que se plantean en estas interacciones en conjunto con las opciones que se le dan para poder avanzar de forma correcta. No todas las opciones tienen garantizada una respuesta benigna y en ocasiones algunos objetos pueden acarrear efectos nocivos en el personaje. Otra de las particularidades del género roguelike es precisamente la abstracción en cuanto a explicaciones. Se deja que sea el jugador quien descubra qué sucede en el mundo, que hacen algunos objetos. Estas responsabilidades se ven delegadas a interacciones con la historia y el entorno entre el jugador y está. La opción que elegimos modifica la recompensa que obtenemos y la única forma de anticipar qué

recompensa será o qué estado nos proporciona un objeto al consumirlo será leer y comprender el texto que acompaña al objeto o localización.



Ejemplo de interacción con un lugar y sus tres opciones a elegir - Junio 2024



Ejemplo de inventario lleno de objetos y la descripción de uno de ellos - Junio 2024

5.3.5- Finales

Como cierre a todo el ciclo de juego, la partida termina con una decisión dual que nos llevará a una secuencia en 3D donde podremos leer un texto con dos posibles finales. Esto añade otra capa de rejugabilidad al programa permitiendo que el jugador elija uno diferente al anterior y entienda algo más de historia y trasfondo.

6.- Pruebas

El juego se ha visto sometido a diferentes iteraciones de pruebas siendo las más destacables las pruebas ciegas en las cuales se ha entregado el juego a gente ajena al proyecto (familia o amigos del alumno) para conocer opiniones y comprobar si se han integrado bien las mecánicas de los CRPG dentro del proyecto. Las primeras iteraciones de pruebas controladas se realizaron sobre la prueba de concepto para afianzar funcionalidades y conocer sectores a mejorar. Las pruebas tanto controladas como no en entornos ajenos al alumno se realizaron directamente sobre el proyecto final en la misma versión que se usará para la defensa. Estas pruebas permitieron pulir ligeramente algunos errores menores y esclarecer partes del programa para un mejor entendimiento global de la entrega.

7.- Aportaciones originales

Conociendo de antemano los márgenes y metas que suelen tener los proyectos de fin de grado que los alumnos de DAM / DAW suelen escoger, se comprende antes de empezar el desarrollo que este proyecto tiene metas, especificaciones y funcionalidades bastante más amplias que la media. El alumno que desarrolla el proyecto no conocía nada de programación antes de empezar el curso en 2022-2023 y con la realización final del proyecto se pretende animar a futuros alumnos a ir un poco más allá y aprovechar la oportunidad brindada por los centros de formación y el profesorado para mostrar lo que son capaces de hacer.

8.- Conclusiones, metas, dificultades y mejoras

Una vez terminado el proyecto y mientras se realiza la documentación final, disponemos de un pequeño momento de reflexión donde la introspección sobre los meses de trabajo nos permiten ser críticos con el programa realizado.

8.1.- Metas alcanzadas

Si nos situamos 3/4 meses atrás de la redacción de este documento, en el momento en que se redactó el documento de pre-proyecto, se pueden desgranar todas las metas y como se ha llegado a cumplir con ellas.

Por un lado tenemos las funcionalidades clave que debía cumplir el programa, siendo estas la ejecución tanto en plataformas de escritorio (Windows) como Android. Si bien la versión de Android necesita más pruebas, ambas se cumplen. Por otro lado tenemos el requisito de que la aplicación sea multi idioma. Tanto los textos de contenido que se encuentran en la base de datos como los textos injertados mediante addons de lenguaje dentro de las interfaces de Unity cumplen también dicho requisito.

La base de datos se ha estructurado de forma orientada a servir al juego y permitir una escalabilidad potente gracias al código comprensivo con los datos de entrada. El flujo de juego puede ejecutarse sin problemas de inicio a fin y, pesé a que en el estado actual de la entrega apenas se dispone de contenido jugable (entendiendo por ello cantidad, no calidad ni funcionalidades inexistentes) se ve que la interacción de todos los componentes se cumple. El inventario funciona, con todas sus opciones, la generación procedural también, el mapa y los finales están completos.

Con todo ello podemos afirmar sin lugar a dudas que las metas fijadas desde un inicio, si bien ambiciosas en producción, han sido alcanzadas con éxito. La aplicación muestra el esqueleto totalmente funcional del juego siendo lo único faltante la generación de contenido de forma manual referente a historia y trasfondo del juego para poder llenar los lugares y hacer más interesante la partida. Cabe destacar de nuevo que, gracias a como se ha orientado el proyecto, código y base de datos, para hacer este paso no hace falta programar nada. Solo tenemos que insertar más datos en el archivo .CSV de “lugares” y la base de datos los entenderá de forma nativa para luego trasladarlos a nuestras partidas.

8.2.- Dificultades encontradas

La principal dificultad que se ha encontrado ha sido el tiempo del que se disponía para realizar el proyecto en su totalidad. Un proyecto como el aquí descrito en solo tres meses,

teniendo en cuenta que dicho tiempo estaba compartido con una jornada laboral completa y otros temas personales, es un marco bastante escaso. Ha requerido una gran organización así como moderación en todo lo que se pretendía hacer para evitar que el proyecto ramificarse o incluyese demasiados datos y funcionalidades imposibles de implementar en el tiempo estipulado.

Durante el desarrollo del proyecto se han ido detectando diferentes bloqueos menores con problemas en ámbitos desconocidos para el alumno. Bloqueos previsibles inherentes de la integración de tecnologías o campos aún no explorados, bloqueos que siempre existen a la hora de diseñar una aplicación nueva. Estos bloqueos son, por ejemplo, problemas a la hora de gestionar la base de datos y sus conexiones con un entorno nuevo, distribución en la nube, diseño general y del ciclo jugable del videojuego etc. Todos estos bloqueos se han ido solventando en mayor o menor medida con el paso del tiempo hasta aislar los problemas causados por dichos bloqueos y eliminarlos en su totalidad.

8.3.- Mejoras posibles

El programa necesita aún más horas de trabajo siendo la principal carencia que presenta la falta de contenido jugable real. En el momento de entregar el proyecto, el juego se encuentra en su versión 0.9.2 entendiendo por ello que no ha alcanzado la versión 1.0, también denominada “gold”. El contenido que falta es, en términos generales, cantidad de lugares a explorar. Si bien el trasfondo general del juego está escrito e integrado (inicio, algunos lugares y finales) falta más contenido para dotar de atractivo al juego. Se han incluido varios lugares de diferentes tipos para mostrar las funcionalidades, todo está programado y se integra con los parámetros que indiquemos (dimensiones del mapa, cantidad de datos en base de datos, y otros) sin problemas gracias a que el código no ha sido escrito de una forma estricta si no con valores flexibles.

Otra mejora posible es la refactorización de la base de datos. La integración web está realizada y probada, aunque en la versión 0.9.2 funciona en local para más sencillez.

También existe la intención de probar y compilar el juego para que funcione en Linux pero debido a la falta de tiempo y facilidades para realizar pruebas en dicha plataforma se ha optado por realizar las funciones solo en Windows y Android.

Por último, se destaca la posibilidad de añadir nuevos idiomas de forma sencilla en cualquier momento con poco mantenimiento de código gracias al diseño de la base de datos.

9.- Bibliografía y enlaces

Lista de recursos empleados en la realización del proyecto, programas o assets creados por terceros. Todos los recursos aquí listados son o bien de dominio público, versiones open source o recursos debidamente adquiridos por el alumno de forma que la licencia empleada permite su uso en un entorno tanto educativo como profesional.

1. Mila del Monte. *Castillo en colina* [imagen digital en línea]. Pixabay. 15 de noviembre de 2020 [fecha de consulta 26 de marzo de 2024] Disponible en:
<https://pixabay.com/es/photos/castillo-colina-fortaleza-5745011/>
JPG, 1280 px. By 854 px., 187 kB
2. Unity 3D [en línea] [fecha de consulta: 24 de marzo de 2024]. Disponible en:
<https://unity.com/es>
3. AUTORES COMUNITARIOS. *Ultima* [en línea]. Wikipedia. [fecha de consulta: 6 de abril de 2024]. Disponible en: <https://es.wikipedia.org/wiki/Ultima>
4. AUTORES COMUNITARIOS. *Eye of the Beholder* [en línea]. Wikipedia. [fecha de consulta: 6 de abril de 2024]. Disponible en:
[https://es.wikipedia.org/wiki/Eye_of_the_Beholder_\(videojuego\)](https://es.wikipedia.org/wiki/Eye_of_the_Beholder_(videojuego))
5. AUTORES COMUNITARIOS. *Role-playing video game* [en línea]. Wikipedia. [fecha de consulta: 6 de abril de 2024]. Disponible en:
https://en.wikipedia.org/wiki/Role-playing_video_game
6. Aseprite [en línea] [fecha de consulta: 24 de marzo de 2024]. Disponible en:
<https://aseprite.org/>
7. Audacity [en línea] [fecha de consulta: 24 de marzo de 2024]. Disponible en:
<https://www.audacityteam.org/>
8. Linktr de Irene. Redes [en línea]. [fecha de consulta: 26 de marzo de 2024]. Disponible en: <https://linktr.ee/TuturuArt>
9. Web oficial de Kenney. Assets gratuitos [en línea]. [fecha de consulta: 26 de marzo de 2024]. Disponible en: <https://kenney.nl/>
10. Carpeta pública con datos del proyecto [en línea]. Fernando Tarriño del Pozo.
Disponible en:
https://drive.google.com/drive/folders/1hIA3n_Y_TiKCLjam2W7bnFvy-PNVHsRH
11. SQLite Home Page. SGDB [en línea]. [fecha de consulta: 26 de marzo de 2024]. Disponible en: <https://sqlite.org/>

10.- Anexos

A continuación se detalla una lista de recursos empleados para una correcta documentación e investigación por parte del alumno. Los primeros recursos, en especial, son de lectura recomendada si se quiere conocer más sobre el mundo de los CRPG.

1. FELIPE PEPE, 166 AUTORES COMUNITARIOS MÁS. *The CRPG Book Project* [en línea]. Editado por Felipe Pepe. Libro gratuito, Octubre de 2023 [fecha de consulta: 6 de abril de 2024]. Disponible en: <https://crpgbook.wordpress.com/>
2. MATT BARTON y SHANE STACKS. *Dungeons and desktops, the history of computer role-playing games* [en línea y físico]. CRC Press, segunda edición 2019 [fecha de consulta: 6 de abril de 2024]. Disponible en: <https://www.amazon.com/Dungeons-Desktops-History-Computer-Role-Playing/dp/1138574643>
3. BITMAP BOOKS. *A guide to japanese role-playing games* [disponible en pdf y físico]. Bitmap books 2021 [fecha de consulta: 6 de abril de 2024]. Disponible en: <https://www.bitmapbooks.com/products/a-guide-to-japanese-role-playing-games>
4. AARON A. REED. *50 years of text games: From Oregon Trail to A.I. Dungeon and everything* [proyecto de kickstarter, en línea pdf]. AARON A. REED primera edición marzo 2023 [fecha de consulta: 6 de abril de 2024]. Disponible en: <https://if50.textories.com/>
5. ROBERTO HUERTAS. Plugin SQLite4Unity3D [en línea]. Publicado de forma gratuita en GitHub, 2019 [fecha de consulta: 6 de abril de 2024]. Disponible en: <https://github.com/robertohuertasm/SQLite4Unity3d>