TFG DAM - CIFP LaLaboral

Eldoria - Alejandro Orviz Recalde

Índice

1. Agradecimientos

Este proyecto no seria posible sin la ayuda de varias personas involucradas en este desarrollo. Carles por ayudarme con sus reviews de código, Laura por darme parte de sus conocimientos en el desarrollo de videojuegos para poder desarrollar bien, Marta la tutora por resolverme dudas y otorgarme documentación, y por ultimo Miguel por haberme inspirado a realizar este videojuego, el cual está dedicado hacia él.

Las funciones de estas personas me han ayudado en mayor o menor medida a terminar este TFG, el cual no podría haber acabado sin ayuda, es por ello que reservo esta parte del docuemnto para agradecer a estar personas su colaboración conmigo, sobre todo a Carles y Laura, los cuales me han apoyado durante todo el desarollo.

2. Introducción

El presente proyecto trata sobre un Juego llamado **Eldoria**, se encuentra desarrollado en *Java* y es una aventura en 2 dimensiones con graficos *pixel art* el cual tendra una forma de jugar similar a videojuegos antiguos tales como:

- The legend of Zelda
- The legend of Zelda Minish cap
- Final Fantasy I, II, III
- Super Mario RPG

Algunos datos del proyecto adicionales son:

Titulo de la aplicacion	Eldoria
Autor	Alejandro Orviz
Profesor	Alberto
Tutora	Marta
Code reviews	Carles
Desarrollo de videojuegos	Laura

El videojuego se inspira fuertemente en los primeros juegos creados para la Super Nintendo, por lo que se veran bastantes similitudes tanto en estructura de niveles como en linealidad e historia del mismo. Ademas todas las images que se han usado son libres de derechos de autor o en su defecto creadas con inteligencia artificial para evitar problemas de copyright.

2.1. Motivación

A dia de hoy son muchas las personas que juegan a videojuegos, pero no muchos recuerdan las bases u origenes de los mismos, en este caso, en mi juego llamado **Eldoria** tenemos una aventura en 2 dimensiones en estetica *pixel* que nos lleva a un mundo de fantasia el cual nos puede provocar la nostalgia de revivir videojuegos como *The legend of Zelda 1*. Ademas este videojuego aprovecha conceptos de programacion sobre lectura, apertura y creacion de archivos asi como conexiones con bases de datos. La duracion del mismo aun esta por determinar y depende mas bien de la persona y del tipo de jugador. Con este proyecto se busca es proveer al usuario de un tiempo de ocio en el que pueda disfrutar pasando el tiempo frente a retos y acertijos propuestos por el videojuego.

2.2. Inspiraciones

Este juego además tiene varias inspiraciones en el mundo moderno y *retro*, pasaremos a explicar en profundidad sus inspiraciones, las cuales incluyen tambien documentos realizados por otra gente como veremos a contiuación:

1. **The legend of Zelda**

   - *The legend of Zelda* es un videojuego japonés de los años 80, el cual fue desarrollado por Nintendo. Se trata de un videojuego de aventuras, puzzles y acción, fue un referente en su época y a día de hoy se considera la base de los juegos de aventuras.
   - El objetivo dentro del juego es simple, nuestro protagonista *Link*, debe rescatar a la princesa *Zelda* de las garras del *Rey Demonio*, para ello tendrá que recorrer *Hyrule* en busca de objetos, armas, conocimiento, etc ...
   - Hablando de temas tecnicos, el juego se considera un *RPG (Role playing game)*, es decir, un juego de rol, de perspectiva aérea que cuenta con varios puzzles para superar algún nivel. En tema de gráficos, encontramos una estética *pixel art* la cual, no requiere un desarrollo super extenso.
   - Aunque a día de hoy existen varios videojuegos de *The legend of Zelda*, nosotros nos fijaremos en el primero ya que es el que dejó las bases para el resto de los juegos de la saga, es por eso que el videojuego, en estética y jugabilidad, bebe mucho de esta obra.

2. **Final fantasy**

   - *Final fantasy* es un videojuego de rol japonés desarrollado por *Square.Co(Más tarde conocida como "Square Enix")*, el cual fue un gran pilar en la industria de los videojuegos de rol de la época y actuales, salió en el año 1987, y fue el inicio de una saga de videojuegos que a día de hoy continua su desarrollo(*Actualmente existe el "Final Fantasy XVI"*)
   - La historia de este juego, trata acerca de 4 personajes que van en busca de unos cristales mágicos para vencer el mal que acecha el mundo, esta historia se repetirá en mayor o menor medidad en los próximos juegos de la saga. Su estética es similar al juego previo hablado, *The legend of Zelda*, pues ambos juegos salieron casi en el mismo año y en una época donde el hardware, no era tan potente como ahora.

3. **TFG de la Escuela de ingenieria de Segovia**

   - Para este proyecto primero, hemos investigado las opciones ya creadas por la gente, en caso de que existieran, en este caso hemos encontrado un TFG (*Estará adjuntado en la parte de Bibliografia*), el cual nos habla acerca del desarrollo de un videojuego móvil con unity, este TFG, nos ha servido de ayuda ya que nos ha provisto de una estructura de desarrollo, enlaces de interés e ideas para el desarrollo.

2.3. Herramientas usadas

En esta sección, explicaremos brevemente las herramientas usadas, y el por qué de las mismas, asi como su coste en el caso de que sean de pago.

Para el desarrollo de este TFG, se han usado las siguientes herramientas:

1. **IntelliJ Idea Community y Ultimate**
    - *IntelliJ* es nuestro IDE de preferencia, el cual nos provee de varias herramientas utiles y necesarias para el desarrollo del proyecto, se trata de un entorno de desarrollo, lanzado en 2001, el cual cuenta con dos versiones, una gratuita y otra de pago, en este proyecto usaremos principalmente la gratuita, aunque necesitaremos una funcionalidad de la versión de pago.
    - Este entorno de desarrollo cuenta con una tecnología de sugerencias de código real, bastante avanzada la cual nos provee de código por lo general bastante acertado según lo que estamos desarrollando, ya que mientras programamos, el IDE, analiza lo que estamos escribiendo y lo que hemos programado para darnos mejores sugerencias de autocompletado.
    - **El precio de este IDE, actualmente(2024) es de: 160 euros al mes**
2. **Docker**
    - **Docker** se trata de un proyecto de código abierto, que nos permite automatizar y facilitar el despliegue de aplicaciones dentro de *contenedores de software*, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.
    - **Docker** ademas está construido sobre las facilidades proporcionadas por el kernel Linux (principalmente cgroups y namespaces), un contenedor Docker, a diferencia de una máquina virtual, no requiere incluir un sistema operativo independiente, por lo que nos permite tener procesos aislados del sistema *"host"* y tener contendores ligeros.
3. **Jooq**
    - **Jooq** tambien conocido como Java Object Oriented Querying, es una biblioteca ligera de mapeo de bases de datos que implementa el patrón de registro activo, su proposito es ser tanto relacional como orientada a objetos.
    - Hemos usado esta biblioteca debido a su similitud con *Linq* el cual es un componente de .NET, que nos permite hacer consultas de una manera muy cómoda.
4. **Visual studio Code**
    - **Visual studio Code** es un IDE, que nos provee de una cantidad de configuraciones infinita, y también al tener esta versatilidad es perfecto como IDE, para trabajar en un proyecto que involucre usar varios lenguajes a la vez, aunque en nuestro caso, lo hemos usado para desarrollar el documento LaTeX, de la memoria del proyecto.
    - Para desarrollar el documento, hemos tenido que descargarnos *Perl*, *LaTeX Workshop* y por ultimo *MikTex*, esto es necesario si queremos que Visual Studio nos deje desarrollar cualquier docuemnto LaTeX, además de un plugin para poder visualizar el documento pdf compilado desde Visual Studio.
5. **LaTeX**
    - **LaTeX** es un sistema de composición de textos orientado a la creación de documentos escritos que presenten una alta calidad tipográfica. Por sus características y posibilidades, se usa de forma especialmente intensa en la generación de artículos y libros científicos que incluyen, entre otros elementos, expresiones matemáticas

- **LaTeX** está formado por un gran conjunto de macros de **TeX**, escrito por Leslie Lamport en 1984 con la intención de facilitar el uso del lenguaje de composición tipográfica LaTeX, creado por Donald Knuth. Se utiliza mucho para la composición de artículos académicos, tesis y libros técnicos, dado que la calidad tipográfica de los documentos realizados en LaTeX se considera adecuada a las necesidades de una editorial científica de primera línea, muchas de las cuales ya lo emplean

- Para poder desarrollar el documento de la memoria en LaTeX, hemos hecho uso de la docuemntación que existe en internet, asi como documentos ya hechos, por ejemplo repositorios de gente que ya ha realizado un documento LaTeX.

6. **MySql**

   - **MySql** es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por *Oracle Corporation* y está considerada como la base de datos de código abierto más popular del mundo, y una de las más populares en general junto a *Oracle* y *Microsoft SQL Server*, todo para entornos de desarrollo web.

   - **MySql** fue inicialmente desarrollado por *MySQL AB* (empresa fundada por David Axmark, Allan Larsson y Michael Widenius). *MySQL AB* fue adquirida por *Sun Microsystems* en 2008, y ésta a su vez fue comprada por *Oracle Corporation* en 2010, la cual ya era dueña desde 2005 de Innobase Oy, empresa finlandesa desarrolladora del motor *InnoDB* para *MySQL*.

7. **JDK 21**

   - **JDK o *Java Development Kit*** es una seria de herramientas de desarrollo que nos permite crear software en *Java*, ademas de poder ejecutar archivos *"Jar"* desarrollados con esta *JDK*. Hemos decidido usar la última versión para estar libres de problemas de que nos hagan falta paquetes o de vulnerabilidades.

8. **Github Repositories**

   - Para poder tener un sistema de control de versiones hemos optado por usar los repositorios de *Github*, los cuales de forma gratuita nos dejan tener un sistema de control de versiones que nos permite mantener de forma sincronizada en varios equipos el mismo proyecto, en nuestro caso el repositorio es público, en caso de quererlo privado, tendríamos o bien que pagar o bien disponer de un servidor local con el que poder tener subido el repositorio en él mediante *Git*.

9. **Excalidraw**

   - **Excalidraw** es una web que nos ofrece una pizarra virtual en la que poder dibujar, hacer notas o generar diagramas incluso, como los diagramas UML de software, ademas cuenta con bastantes opciones a la hora de exportar lo dibujado, y es una herramienta muy útil para hacer esquemas.

2.4. Primeras metas

1. **Realizar un personaje que se mueva en pantalla**
    - El principal punto sobre el que podemos avanzar es mostrar el personaje en pantalla y ser capaces de moverlo con las teclas del teclado, que el programa sea capaz de saber si queremos ir a las 4 direcciones posibles y que se mueva con cierta fluidez.
2. **60 Frames por segundo**
    - Una de las metas tambien principales es limitar el rendimiento del juego ya que queremos que se dibuje lo que hay en pantalla 60 veces por segundo, si no ponemos un limitador, esta cifra sería mucho más alta, lo que nos podria provocar problemas, llegados a un número muy alto de interacciones.
3. **Colisiones**
    - Dentro de un videojuego queremos que se produzcan colisiones con el personaje o personajes del juego, no queremos que se atraviesen muros, o cosas de ese estilo, por lo que esta es una de las principales metas a superar.
4. **Menús de juego**
    - También queremos realizar pantallas de estado del personaje, menú principal, dialogos, menú de pausa, etc ...
5. **Sistema de combate**
    - Al ser un juego de rol y aventuras, queremos incluir algún enemigo y algun sistema de combate en el que se pueda dar y recibir daño asi como barras de vida para los enemigos, o para el jugador.
6. **Pantalla Completa**
    - Otra meta seria poder ejecutar el juego en pantalla completa sin bordes, es un pequeño punto de estética.
7. **Empaquetar el juego**
    - La ultima meta propuesta es empaquetar el juego y poder tener algun ejecutable para poder llevarlo a *"todas partes"*.

3. Planificacion y presupuesto

3.1. Planificacion

Este proyecto se ha desarrollado en varias fases como un desarrollo real en un entorno empresarial, empezando en *Octubre de 2023*(los totales de horas son aproximados).

3.2. Septiembre - 2023

En esta fase comenzó el descarte de ideas y la lluvia de las misma para saber más o menos que podríamos hacer en cuanto a TFG se refiere

Total de horas: 10 horas

3.3. Octubre - 2023

En la primera fase del mismo se ha pensado la idea y hecho una planificacion inicial del mismo, asi como varios apuntes sobre la tecnología a usar, la cual ha recibido cambios en el desarrollo.

Total de horas: 20 horas

3.4. Noviembre - 2023

En esta fase empezamos los primeros pasos del desarrollo y encontramos las primeras dificultades del mismo, como por ejemplo:

- Assets
- Falta de conocimiento
- Desconocimiento de la BDD
- Diseño de la BDD

Teniendo estas dificultades claras continuamos un desarrollo temprano de una pequeña version bastante primitiva del proyecto que nos servirá de base para poder continuar con él. En esta versión declaramos las bases del programa final que tendremos por el final del proyecto.

Total de horas: 30 horas

3.5. Diciembre - 2023

En esta fase continuamos un poco mas el desarrollo y adelantamos un poco del mismo de cara a la siguiente fase, ya que en el momento que hemos acabado la fase de primeros pasos el resto que se realice antes del anteproyecto es trabajo adelantado, dentro de esta fase el principal problema que nos hemos encontrado ha sido **el tamaño del heap de Java**, siendo este el principal limitante a la hora de realizar un diseño de niveles grande como se tenía pensado principalmente. Este problema nos produjo un cambio de planes ya que el mapa que inicialmente se habia pensado en un mapa de 100x100 casillas se tuvo que reducir a 50x50, es decir la mitad, esto nos deja para las pruebas del desarrollo un mapa de 25x25 que más tarde será de 50x50. Este cambio viene de la famosa excepcion *"java.lang.OutOfMemoryError: Java heap space"*.
Este problema se produce al dibujar el mapa del videojuego, ya que el heap no es capaz de almacenarlo y entonces nos devuelve esa excepción, para corregirlo, lo que hemos hecho ha sido disminuir el tamaño del mapa a uno más pequeño, si bien es cierto que se puede aumentar la memoria del heap de java mediante comandos, hemos preferido dejarlo como está y disminuir el mapa.
Otros datos interesantes de esta fase han sido los progresos en el desarrollo y la nueva replanificación del mismo, ya que al tener el error del heap de java, hemos tenido que reestructurar partes del desarrollo ya planificadas.

Total de horas: 50 horas

3.6. Enero/Febrero/Marzo - 2024

Dentro de esta fase tendriamos la parte referente al anteproyecto. El desarrollo del documento del anteproyecto se realizó usando LaTeX, y *Overleaf (Una pagina dedicada a escribir documentos LaTeX de forma online)*. La plantilla usada en ese documento es una plantilla creada desde cero con los conocimientos de ese momento acerca del lenguaje, y tambien cogiendo cosas de documentación, foros y proyectos ya hechos de otra gente, es el caso de foros como *StackExchange* en su parte referente al lenguaje, repositorios de github con determinadas funciones que he necesitado poner en el documento, etc...
Por ultimo dentro del anteproyecto se han escrito cosas acerca de funcionalidad finales que no se han podido cumplir por falta de tiempo, estos comentarios se tendrán en cuenta como *propuestas de mejora*, las cuales son pequeñas o grandes funcionalidades que pueden mejorar o hacer más divertido el juego.

Total de horas: 40 horas

3.7. Abril/Mayo - 2024

La penultima fase del proyecto, en esta se encontraria la recta final de desarrollo en la que ya nos ponemos en serio con el mismo y ultimamos los detalles del programa, ademas de añadir todo lo que podamos hasta el dia 3 de junio, el cual es la fecha limite que hemos planificado como *deadline*. Durante esta fase hemos tenido ayuda, bien sea de docuemntación, foros, videos o incluso personas, en esta fase hemos recibido en concreto ayuda de 3 personas, la primera sería la tutora encargada del mismo, que nos ha proporcionado, docuemntación y links de interés para resolver las dudas surgidas, la segunda en este caso sería Laura, una desarrolladora de videojuegos que nos ha dotado de consejos a lo largo del desarrollo y pautas que hemos podido seguir asi como webs donde buscar assets gratis, y por ultimo, Carles, que se trata de un desarollador titulado en DAM, el cual se ha encargado de hacer *code reviews* sobre el codigo del proyecto y el cual nos ha dicho pequeños aspectos que se nos han pasado a la hora de desarrollar, como los comentarios o la indentación correcta.

Total de horas: 120 horas

3.8. Junio - 2024

Ultima fase del proyecto en la que nos hemos centrado en la documentacion del mismo aunque se ha hecho de manera transversal, pero se han dado los ultimos retoques en esta fase ya que el desarrollo acabó el dia 3. Dentro de esta fase hemos buscado muchas funciones y curiosidades para poder hacer un documento parecido a un TFG universitario pero a menor escala, para ello hemos visualizado y leido varios TFGs, de varias universidades de España y hemos tenido la suerte de poder tenerlos descargados para poder ver la estructura de los mismos, como se han escrito y tambien como se han desarollado las personsa que los han realizado.

Total de horas: 50 horas

3.9. Diario de horas del proyecto

Tarea	Fecha de Inicio	Fecha de Fin	Horas Planificadas
Diseño de Juego	20/08/2023	07/02/2024	20
Desarrollo del juego	08/01/2024	03/06/2024	180
Pruebas y Depuración	03/06/2024	10/06/2024	30
Diseño de Niveles	10/01/2024	10/01/2024	15
Apartado artistico de personajes	04/10/2024	10/04/2024	25
Documentación	10/09/2023	10/06/2024	50
Total de Horas Planificadas			**320**

3.10. Presupuesto

Dentro de esta parte del proyecto hemos intentado abaratar en algunas partes costes y en otras no nos ha quedado más remedio que tener que pagar, ya que o bien nos ofrecen un servicio/funcionalidad que necesitamos o bien es un programa con el que estamos muy familiarizados. Por lo que en este apartado explicaremos parte de las herramientas usadas, ya que algunas son de pago.

3.10.1. IntelliJ Idea

Empezamos esta sección con nuestro IDE de preferencia, el cual nos provee de bastantes funcionalidades en su versión community, pero en este caso hemos hecho uso de la versión Ultimate, ya que nos realiza el diagrama de clases el mismo IDE, aunque tenemos la licencia educativa, la cual nos deja tener este producto durante nuestros estudios de forma gratuita, hemos optado por incluirlo en el presupuesto ya que sigue siendo una herramienta de pago.

Su precio es de: 160 euros al año

3.10.2. Horas de trabajo

Teniendo en cuenta la media ofrecida por *Glassdor* acerca de lo que puede cobrar un desarrollador de software Java, hemos visto que su sueldo debería ser cercano a 2000 euros, este sueldo si tenemos en cuenta los meses de desarrollo y horas nos sale lo siguiente.
Haciendo calculos, poniendo como sueldo 1700 euros mensuales y viendo las horas del proyecto y el tiempo, deberiamos cobrar **15300 euros netos**. Esta cifra sería si estuvieramos en una empresa la cual nos cobra por mes, pero al presentar el supuesto de ser un autonomo, deberiamos cobrar **1960 euros netos** si tenemos en cuenta las horas totales del proyecto, por lo que tomaremos esta cifra mejor.

Coste de horas: 7.08 euros la hora

3.10.3. Equipo usado

El equipo usado para el desarrollo ha sido un ordenador portatil HP, con un intel core i7 de undecima generacion, 16gb de RAM, y un ssd de 512GB, el precio del equipo sin ninguna oferta seria de 820 euros.

Coste del equipo de desarrollo: 820 euros

3.10.4. Tabla de presupuesto

Sección	Coste
IntelliJ	160 euros
Horas de trabajo	7.08 euros/hora
Equipo usado	820 euros
Total del presupuesto	**2940 euros**

4. Estudio de mercado y viabilidad de la propuesta

4.1. Aplicaciones ya existentes

Este tipo de videojuego ya existe, siendo claros ejemplos, **Final Fantasy 1**, **The legend of Zelda** o incluso **Pokemon rojo**, este tipo de programas que han sido desarrollados hace años tienen en común que siguen la estética *pixel art* y en ellos se entiende ya que por circunstancias temporales, no existía nada mejor. En este caso son ejemplos válidos *Sea of stars* el cual fue lanzado en el año 2023, siguiendo tambien una estetica *retro*, y tambien siguiendo un genero parecido a los juegos ya mencionados.

4.2. Viabilidad

Dentro de este proyecto nos enfrentamos a varios obstaculos y decisiones que nos van a hacer tardar mas o tardar menos en desarrollar el proyecto. Los principales obstaculos que nos hemos encontrado han sido:

- **Desconocimiento de la lógica de un videojuego:**
  La principal dificultad de este punto ha sido la poca familiaridad de la creacion de un programa que funcione como un videojuego, ya que se requiere una logica distinta a la que estoy familiarizado.
- **Desconocimiento en LaTeX:**
  A pesar de haber escogido LaTex para realizar el documento, el principal obstaculo ha sido tener en cuenta casi todos los aspectos de formato bonito para el documento, ya que la falta de experiencia documentando en LaTeX ha hecho complicado poder seguir de forma fluida el proyecto
- **Desconocimiento en la gestión de memoria:**
  La gestión de memoria es crucial en el desarrollo de software. La falta de entendimiento de este aspecto nos ha provocado un problema de rendimiento en las etapas tempranas del desarrollo.
- **Diseño de niveles:**
  En el caso de este punto ya que carecemos de teoria acerca de niveles se ha hecho complicado poder realizar un mapa competente ya que carece de una tecnicidad o complejidad excesiva.
- **Diseño de personajes:**
  En cuanto a los personajes se ha optado por usar galerias gratuitas, el principal obstaculo ha sido cuadrar los sprites ya que cada uno viene de autores diferentes y no puede quedar todo diferente.
- **Implementación de base de datos:**
  El principal problema de la Implementación de la base de datos, ha sido el que guardamos en la base de datos, ya que no podemos guardar algo que se tenga que consultar cada poco, por que provocaria en este caso problemas de rendimiento.
- **Desconocimiento de herramientas:**
  Uno de los principales problemas y lastres a la hora del desarrollo es el hecho de no contar con las suficientes herramientas, ya sea para diseñar niveles, personajes, o automatizar procesos
- **Desarrollo parcialmente solitario:**
  Si bien es cierto que contamos con ayuda de la tutora correspondiente para realizar el TFG, no contamos con un desarrollador al lado que pueda ver quizás optimizaciones o refactorizaciones que nosotros no.
- **Envergadura del proyecto:**
  En fases finales del proyecto, el tiempo ha sido un factor importante el cual ha podido incluso decidir por nosotros que funciones metemos y cuales no.

5. Requisitos de la aplicacion

Dentro de esta aplicacion no necesitamos tampoco unos requisitos muy avanzados, pero aun asi dejaremos anotados ciertos requisitos minimos y recomendados

5.1. Requisitos Mínimos

- **Procesador (CPU)**: Procesador de al menos 1.0 GHz.
- **Memoria RAM**: 1 GB de RAM.
- **Tarjeta Gráfica (GPU)**: Tarjeta gráfica integrada o dedicada capaz de manejar gráficos 2D simples.
- **Almacenamiento**: 100 MB de espacio libre en disco.
- **Sistema Operativo**: Windows 7/8/10, macOS 10.10 o superior, Linux con kernel 2.6 o superior.

5.2. Requisitos Recomendados

- **Procesador (CPU)**: Procesador de doble núcleo de al menos 2.0 GHz.
- **Memoria RAM**: 2 GB de RAM.
- **Tarjeta Gráfica (GPU)**: Tarjeta gráfica integrada o dedicada para gráficos 2D mejorados.
- **Almacenamiento**: 500 MB de espacio libre en disco.
- **Sistema Operativo**: Windows 10, macOS 10.14 o superior, Linux con kernel 4.0 o superior.

5.3. Requisitos del usuario

Los requisitos que debe cumplir el usuario, son tener simplemente un teclado, bien sea conectado por USB, o por Bluetooth al ordenador.

5.4. Alcance

Esta sección explica a quién va dirigido el desarrollo de este videojuego, en este caso a cualquier persona que tenga un ordenador y que quiera jugar un videojuego de aventuras. La dificultad de este videojuego, es estática, es decir, de momento no existe un selector de dificultad que permita seleccionarla.

- **Juego RPG:** El juego presenta una mecánica de juego de rol que para algunas personas puede resultar complicado.
- **Juego de aventuras:** El juego presenta algún puzzle por el cual se debe idear o pensar una estrategia para avanzar.

6. Diseño técnico

En esta parte del proyecto detallaremos la arquitectura física y lógica del videojuego.

6.1. Arquitectura física

Para la arquitectura fisica, solamente detallaremos el teclado, ya que es lo único físico que usara el usuario para poder jugar al videojuego. Para controlar el personaje, el usuario dará unos *ïnputs"* con el teclado que harán ciertas acciones dentro del juego, esto se detallará más adelante en el manual de usuario, donde diremos como se controla el juego con las teclas.

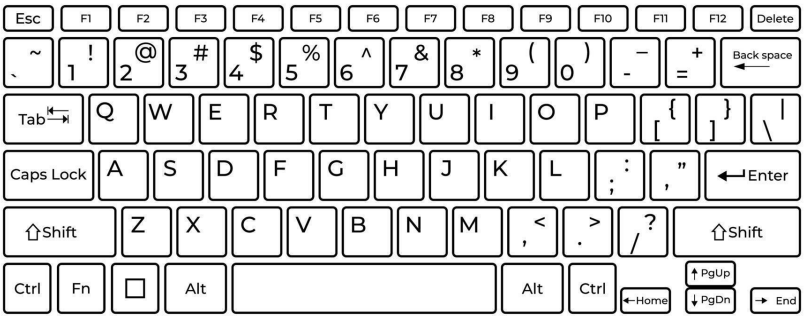

Figura 1: Imagen de un teclado

6.2. Plataforma de despliegue

Para el despliegue de este programa usaremos un ordenador, en el cual tengamos el contendor de *Docker* instalado y encendido, y tambien por ultimo, que tenga la *JDK 21*, instalada, además de que este despliegue ocurrirá en un sistema Windows primero, y después se barajará la posibilidad de desplegarlo en sistemas *Linux*, basados en *Debian*.

6.3. Arquitectura logica

Estructura del proyecto en el diagrama de clases:

6.3.1. Diagrama de clases

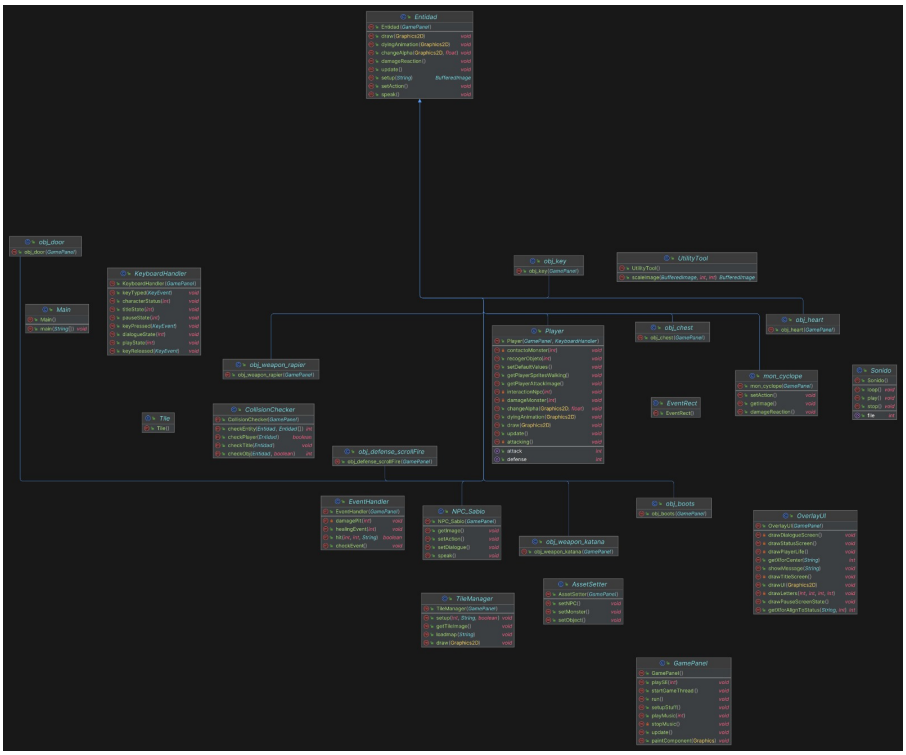

Figura 2: Diagrama de clases

6.3.2. Explicaciones del diagrama

Como podemos ver en el diagrama, hay varias clases importantes a lo largo de la estructura del mismo, que son, **Player, Gamepanel y Entidad**, estas clases son las más importantes ya que nos ayudan a definir muchas cosas.
En el caso de *Entidad*, nos permite desarrollar enemigos, objetos, npc y el propio jugador, ya que estos objetos, extienden de *Entidad*, lo que deja que esta clase sea una clase llena de atributos.

6.3.3. Metodo run

En cuanto a *Gamepanel*, esta clase sería la encargada de tener el panel de juego y conectarse con todas las clases del proyecto, ya que, por ejemplo define el tamaño de los assetts, el tamaño del texto, y varios aspectos más de dentro del juego, esta clase además es de las más importantes ya que contiene parte de los metodos más criticos del proyecto como por ejemplo el siguiente.

Este método empieza inicializando 6 variables que le servirán para calcular un *delta*, para ello hace uso de esta ecuación:

$$\text{IntervalDraw} = \frac{1}{\text{FPS}}$$

Como vemos esta ecuación sería la inversa de los *FPS o Frames per Second*, dicho esto explicaremos cada variable.

- *IntervalDraw:* esta variable es la encargada de calcular en nanosegundos el tiempo necesario para dibujar un fotograma a la velocidad deseada
- *delta:* Lleva un seguimiento del tiempo transcurrido desde el último fotograma.

```java
@Override
public void run() {

    double IntervalDraw =  1000000000/FPS;
    double delta = 0;
    long LastTime = System.nanoTime();
    long CurrenTime;
    long Timer = 0;
    int DrawCount = 0;
        while (gameThread != null){
            CurrenTime = System.nanoTime();
            delta += (CurrenTime - LastTime) / IntervalDraw;
            Timer+=(CurrenTime-LastTime);
            LastTime = CurrenTime;
            if (delta >= 1){
                update();
                repaint();
                delta--;
                DrawCount++;
            }
            if (Timer >=1000000000){
                System.out.println("FPS: " + DrawCount);
                DrawCount = 0;
                Timer = 0;
            }
        }
}
```

Figura 3: Método del hilo principal

- *LastTime:* Almacena el tiempo en nanosegundos en el que se procesó el último fotograma.
- *CurrenTime:* Almacena el tiempo actual en nanosegundos.
- *Timer:* Lleva el seguimiento del tiempo transcurrido.
- *DrawCount:* Es el contador de FPS del juego.

Bucle principal

Esta parte del codigo, ejecuta un bucle (El bucle se ejecuta mientras gameThread no sea nulo (es decir, mientras el juego esté funcionando)) en el cual ocurre lo siguiente:

- Se actualiza el valor de CurrenTime.
- Se calcula el tiempo transcurrido desde el último fotograma y se agrega a delta.
- Se actualiza el valor de LastTime.
- Si delta supera 1, se llama a los métodos update() y repaint() para actualizar y dibujar el siguiente fotograma.
- Se decrementa delta en 1 y se incrementa DrawCount.
- Si Timer supera 1 segundo (1,000,000,000 nanosegundos), se muestra el número de fotogramas dibujados en la consola y se reinician los contadores.

Es decir, este metodo controla la lógica del dibujado o render del juego para que sea siempre de 60 fotogramas por segundo, es decir, dibuja 60 veces por segundo lo que tenemos en pantalla de esta manera no tenemos valores extremadamente altos que no sirve de nada tener en un juego de este estilo.
Acabando con las clases principales, tenemos la clase *Player* la cual tiene todos los metodos relacionados al jugador, es decir, el metodo de atacar, el metodo de moverse, colisiones con objetos, etc ...

6.3.4. Método loadmap

Aun asi, vale la pena destacar varios metodos interesantes como el de *"loadmap"*, que encontramos en la clase *"TileMananger"*, la clase encargada de leer el mapa del juego.

```
public void loadmap(String filepath){

    try{
        InputStream is = getClass().getResourceAsStream(filepath);
        BufferedReader br = new BufferedReader(new InputStreamReader(is));

        int col = 0;
        int row = 0;

        while (col < gp.maxWorldCol && row < gp.maxWorldRow){

            String line = br.readLine();

            while (col < gp.maxWorldCol){
                String numbers [] = line.split(" ");

                int num = Integer.parseInt(numbers[col]);

                mapTileNum[col][row] = num;
                col++;
            }
            if (col == gp.maxWorldCol){
                col = 0;
                row++;
            }
        }
        br.close();

    }catch (Exception e){
        e.printStackTrace();
    }

}
```

Figura 4: Método de carga de mapas

Explicacion del metodo

Este metodo funciona de la siguiente manera, a modo de resumen, lee el archivo .txt del mapa que esta compuesto solo de numeros y a cada numero le asigna una casilla, de esta manera, según el numero la casilla tiene una imagen u otra.

Ahora de forma extensa, explicaremos paso por paso que hace este metodo:

1. El método toma una cadena, *"filepath"*, esta cadena se refiere a la ubicación del mapa en el sistema de archivos del ordenador, es decir, la ruta del archivo
2. ÏnputStream is = getClass().getResourceAsStream(filepath); BufferedReader br = new BufferedReader(new InputStreamReader(is));.^{es}ta parte del codigo lo que hace es crear un BufferReader que lee el archivo del mapa.
3. "getClass().getResourceAsStream(filepath)"lo que hace es localizar el archivo en el sistema.
4. Las variables *row* y *col* nos sirven para saber en que posición del mapa se encuentra.
5. "while (col <gp.maxWorldCol && row <gp.maxWorldRow).^{el} bucle continua mientras no se llegue al final del mapa
6. "String line = br.readLine();"simplemente lee una linea del archivo

7. "while (col <gp.maxWorldCol).[el] bucle se ejecuta para cada columna actual
8. "String numbers [] = line.split(); int num = Integer.parseInt(numbers[col]); mapTileNum[col][row] = num; col++;.[a]quí se divide la línea en números, se convierte cada número a un entero y se almacena en el array *mapTileNum*
9. ïf (col == gp.maxWorldCol) col = 0; row++; Çuando se ha llegado al final de una fila, se incrementa la variable *row* y se reinicia la variable *col*
10. "br.close();çierra el BufferReader
11. Y por ultimo el catch captura la excepcion en caso de que se produzca

Este es un metodo muy curioso ya que puede producir otra excepción más que no he controlado, estamos hablando de la famosa excepción *java.lang.OutOfMemoryError*, debido a que si tenemos un mapa muy grande la máquina virtual de java, *JVM* se queda sin memoria en el heap y detiene el programa porque ha roto".

6.3.5. Método draw

Otro método interesante sería el método *"draw"* de nuevo de la clase *"TileMananger"*, el cual se encarga de dibujar las çasillas"de juego.
De forma resumida, va dibujando las casillas en la posición determinada en el mapa del juego.

Explicación del método

Este método es bastante simple de entender si hemos entendido los anteriores, por lo que procederemos a su desglose:

1. "public void draw(Graphics2D g2)"primero recibe un objeto del tipo Graphics2D como argumento que usará para dibujar en pantalla.
2. "worlCol y worlRow"son las variables que usaremos para rastrear la posición en la que nos encontramos
3. "while(Wordlcol <gp.maxWorldCol && WorldRow <gp.maxWorldRow).ésteste bucle continua mientras no lleguemos al final del mapa
4. ïnt tileNum = mapTileNum[Wordlcol][WorldRow]ż aqui obtenemos el número de la casilla que se va a dibujar
5. `int worldX = Wordlcol * gp.tileSize; int worldy = WorldRow * gp.tileSize` aquí calculamos las coordenadas `x` e `y` del juego
6. ïnt screenX = worldX - gp.player.wordlx + gp.player.ScreenX; int screenY = worldy - gp.player.wordly + gp.player.ScreenY.ªqui se calculan las coordenadas de la pantalla
7. "g2.drawImage(tile[tileNum].image,screenX,screenY, null);.ªqui dibujamos la casilla
8. Incrementamos las columnas de la variable que rastrea las columnas
9. ïf (Wordlcol == gp.maxWorldCol) Wordlcol = 0; WorldRow++; çuando se ha llegado al final de una fila, se incrementa la variable WorldRow y se reinicia la variable Wordlcol

```
public void draw(Graphics2D g2){

    int Wordlcol = 0;
    int WorldRow = 0;


    while(Wordlcol < gp.maxWorldCol && WorldRow < gp.maxWorldRow){

        int tileNum = mapTileNum[Wordlcol][WorldRow];

        int worldX = Wordlcol * gp.tileSize;
        int worldy = WorldRow * gp.tileSize;
        int screenX = worldX - gp.player.wordlx + gp.player.ScreenX;
        int screenY = worldy - gp.player.wordly + gp.player.ScreenY;

        g2.drawImage(tile[tileNum].image,screenX,screenY, null);

        Wordlcol++;


        if (Wordlcol == gp.maxWorldCol){
            Wordlcol = 0;

            WorldRow++;
        }
    }

}
```

Figura 5: Método de dibujado de mapas

6.4. Diagrama de casos de uso

En el diagrama de casos de uso, tenemos un actor que seria el jugador. Este actor realizaría varias funciones:

- Crear una partida
- Cargar una partida
- Empezar el juego
- Mover el personaje
- Pausar el juego
- Ver el estado del personaje
- Salir del juego
- Activar eventos
- Combatir
- Interactuar con NPCs
- Obtener objetos, experiencia, dinero.
- Comprar y Vender objetos
- Interactuar con el entorno

Insertar imagen del diagrama UML

7. Desarrollo realizado

Para el desarrollo del proyecto hemos usado las siguientes herramientas:

- IntelliJ
- Docker
- Jooq
- JDK 21
- Visual studio Code
- LaTeX
- Excalidraw
- Github Repositories
- MySQL

Ahora pasaremos a los detalles del codigo.

7.1. Detalles del codigo desarrollado

Empezaremos describiendo lo que realiza la clase principal, la clase *"Main"*, cuyo codigo es:

```
    package main;

import javax.swing.*;
import java.awt.*;

public class Main {
    public static void main(String args[]){
        JFrame win = new JFrame();
        win.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        win.setResizable(false);
        //win.setBackground(Color.black);
        win.setTitle("Eldoria");

        GamePanel gpanel = new GamePanel();
        win.add(gpanel);

        win.pack();

        win.setLocationRelativeTo(null);
        win.setVisible(true);
        gpanel.setupStuff();
        gpanel.startGameThread();

    }
}
```

Dentro de este codigo encontramos varios aspectos a destacar, el primero es que estamos creando un JFrame en primera instancia y luego le estamos diciendo tambien que al darle a la "X", nos cierre el programa automaticamente. Continuamos diciendole que no queremos un resize de la ventana actual del juego, por eso el false,

seguimos con el titulo, y a continuacion la instancia de GamePanel, la cual nos servirá para instanciar más cosas. Centramos la ventana con el setLocationRelativeTo, la hacemos visible y le decimos a GamePanel que vaya dibujando ciertas cosas del videojuego e inicie el hilo principal del juego.

8. Bibliografia

- **Repositorio del juego** - https://github.com/Pisa-17/TFG-DAM-Eldoria/tree/main
- **Excalidraw** - https://excalidraw.com/
- **Manual LaTeX** - https://manualdelatex.com/
- **Mermaid** - https://mermaid.js.org/intro/getting-started.html
- **IntelliJ** - https://www.jetbrains.com/es-es/idea/
- **Editor Pixelart** - https://www.piskelapp.com/
- **Como hacer Assets** - https://tips.clip-studio.com/es-es/articles/2484
- **TFG de la Escuela de Segovia** - https://uvadoc.uva.es/bitstream/handle/10324/24495/TFG-B%201042.pdf?sequence=1
- **Docker** - https://www.docker.com/
- **Imagen Docker MySQL** - https://hub.docker.com/_/mysql/
- **Out Of Memory Error** - https://stackoverflow.com/questions/1596009/java-lang-outofmemoryerror-java-
- **Assets** - https://pixel-boy.itch.io/ninja-adventure-asset-pack
- **Autor de los Assets** - https://twitter.com/2Pblog1
- **JooQ** - https://www.jooq.org/
- **Visual Studio Code** - https://code.visualstudio.com/Download
- **Diagrama de clases IntelliJ** - https://www.jetbrains.com/help/idea/class-diagram.html
- **JDK 21** - https://www.oracle.com/java/technologies/javase/jdk21-archive-downloads.html
- **Retro Art Article** - https://medium.com/@dq_irfandi/the-nostalgia-effect-how-retro-games-influence-mo
- **Why retro games are so Loved?** - https://www.wired.com/story/why-retro-looking-games-get-so-much-lov
- **How to list code in LaTeX** - https://es.overleaf.com/learn/latex/Code_listing
- **Libro - The Legend of Zelda Hyrule historia** - https://en.wikipedia.org/wiki/The_Legend_of_Zelda:_Hyrule_Historia - ISBN 978-1-61655-041-7

. Anexo I - Manual de Usuario

Los controles del juego son bastante simples, tendriamos las teclas *W,A,S y D*, para mover el personaje y las teclas *C, P, y ENTER* para otras acciones.

- W - Mover el personaje hacia arriba
- A - Mover el personaje a la izquierda
- D - Mover el personaje a la derecha
- S - Mover el personaje hacia abajo
- C - Ver las estadisticas del personaje
- P - Abrir el menu de Pausa
- ENTER - Interactuar con NPCs, atacar, aceptar o continuar.

. Anexo II - Assets

Aquí va el contenido del segundo anexo...

. Anexo III - Arte e inspiraciones detalladas

Dentro del apartado artistico debemos remontarnos a una época pasada, ya que tomamos muchos aspectos de juegos de décadas anteriores que nos han ayudado en cuanto a diseño, estética y cómo se ha diseñado el juego.

.1. The legend of Zelda

Empezamos por la primera y clara inspiración que es *The legend Of Zelda*, este videojuego bastante importante en la industria, sentó las bases para los próximos videojuegos de los siguientes años y es día de hoy que su fórmula sigue siendo un exito. La fórmula de este juego nos presenta una aventura de un héroe que debe rescatar a una princesa de un rey malvado, para ello durante la aventura se ganará el favor de las diosas de su mundo así como una espada sagrada para poder derrotar a este rey malvado. Esta fórmula gana mucho cuando nos metemos de lleno al juego, y dentro del juego lo que encontramos son puzzles, puzzles que nos proponen desafíos para superar las mazmorras y conseguir objetos. En los años 80 fue uno de los juegos más importantes.

.2. Final fantasy

La segunda inspiración tomada es otro juego de los años 80, el cual se llama *Final Fantasy*, este juego se trata de un juego en dos dimensiones RPG por turnos que nos presenta una historia, acerca de 4 muchachos que se embarcan en la búsqueda de unos cristales mágicos que les servirán para disipar el mal de su mundo, durante la aventura se enfrentan a enemigos formidables y el juego tiene una estética de fantasía medieval. A dia de hoy esta saga sigue en desarollo, actualmente han sacado 16 juegos principales y varios secundarios.

.3. Blasphemous

La tercera inspiración ha sido un juego que se sale de los generos de RPG, se trata de un juego metroidvania, desarrollado en España, se trata de *Blasphemous*, un juego en dos dimensiones que nos pone en la piel del penitente, el cual entrará en un camino de penitencia y tendrá que realizar humillaciones para devolver el orden a su mundo, Cvstodia. Este juego presenta inspiración en mi proyecto ya que se trata de un juego con estética *pixel art*, que es relativamente reciente.

.4. Arte del juego

El arte del juego sería una mezcla de todos los juegos ya mencionados excepto Blasphemous, ya que este último cuenta con sprites muy bien detallados y de mayor resolución. Se supone que el juego se desarrolla en una época medieval en la que existen magias, caballeros, dragones, ciclopes, etc ...

. Anexo IV - Vocablos usados

Durante este documento se han hecho uso de varios vocablos de los que seguramente se desconozca el significado, tales como "FPS"por ejemplo, por ello vamos a recopilarlos aqui junto a su definicion.

- FPS Frames per second - Imagenes que dibuja el programa por segundo, un frame es toda la imagen que dibuja el programa entera, con el personaje, el mapa, los objetos, etc ...
- RPG Role playing game - Acrónimo asociado a los juegos de Rol
- NPC Non Playable Character - Personaje no jugable, suelen ser personajes con los que podemos hablar o Interactuar
- Inputs - Se refiere a las entradas del usuario al programa
- Pixel art - Estética basada en dibujar con pixeles solamente sin ningun arco o elemento circular
- Retro - Se refiere a una estética antigua, por ejemplo la de los juegos de los años 80
- Sprites - Son un conjunto de imágenes que representan un objeto o personaje