

Contenido

1	Introducción	2
2	Objetivos.....	2
2.1	Objetivos funcionales	2
2.2	Objetivos no funcionales:.....	4
3	Tecnologías usadas.....	4
3.1	IDE.....	4
3.2	Editor de mapas.....	5
3.3	Herramienta de diseño.....	5
4	Diseño.....	6
4.1	Diseño técnico	6
4.2	Diseño GUI.....	11
4.2.1	Diseño layout.....	11
4.2.2	Diseño visual y contenidos	14
5	Desarrollo	15
5.1	API JSON	15
6	Testing	17
7	Seguimiento y control en la ejecución del proyecto	23
7.1	Trello.....	23
7.2	GITHUB	24
7.3	Incidencias	24
8	Costes y planificación	25
8.1	RoadMap	25
8.2	Coste final	27
9	Conclusiones.....	34
10	Mejoras y líneas futuras	35
11	ANEXOS.....	36
11.1	SKETCH.....	36
11.2	Guía del usuario.....	38
11.3	Bibliografía/webgrafía consultada	43

1 Introducción

Este proyecto de trata de un videojuego en 2D programado en Java, en que la gran parte del proyecto se programa el motor grafico, sin uso de Frameworks u otras aplicaciones externas.

El videojuego consta de varios géneros, el roguelike, shooter y acción-RPG. En resumen a lo que se refiere a estos géneros, es que trata de un juego donde empiezas con un personaje en una mazmorra, donde se generan mapas aleatorios, los ítems, etc. En concreto este juego, al ser shooter, se trata que el sistema de combate se dé uso de armas de fuego, que podrás ir consiguiendo a medida que avanzas en el juego.

La mazmorra está llena de salas y pisos con gran variedad de enemigos y un enemigo final que es difícil de derrotar para poder avanzar. El objetivo principal es sobrevivir, ya que una vez mueres terminaría la partida y conseguir los máximos puntos posibles para clasificarte en el ranking de los 5 mejores.

2 Objetivos

2.1 Objetivos funcionales

En esta parte se trata de objetivos al que se han conseguido en el desarrollo de la aplicación que en un inicio se habían planteado y se han logrado hacer.

- Programar y diseñar el motor grafico del videojuego.

Básicamente se trata de programar y diseñar toda la carga de imágenes y animaciones del videojuego. Como en el caso de mostrar una ventana con imágenes, la creación de un menú al que poder navegar, las diferentes pantallas que mostrar en cada opción del menú seleccionado, etc.

Dentro de la parte del juego, entraría la carga de los mapas, los personajes, animaciones de cada uno de los elementos o seres vivos que pueda haber, objetos, acciones, las interfaces, etc.

- Poder observar el top de las mejores puntuaciones de los 5 mejores partidas.
Este objetivo, profundiza una parte del menú y un factor importante del juego, en el que se trataría de guardar la información de las diferentes partidas, en este caso puntos, por lo que habría que tener en cuenta un sistema de puntuaciones y mostrarlo en una pantalla.
- Diseño visual.
Una parte muy importante de un videojuego, el diseño. En este caso elegido el estilo pixel Art.
- Enemigos
Toda relación a ser vivo al que derrotar con sus animaciones, su propia vida, fuerza y movilidad.

De los objetivos funcionales actualmente se han implementado más durante el proceso de desarrollo.

- El sistema de combate a distancia
Un requisito al que sin él no tendría gracia el juego. Se trata toda la parte de acción, sistema de armas y las animaciones al disparar hacia un enemigo, con el objetivo de matarlo.
- Cofres aleatorios
Un requisito al que contiene unas probabilidades de conseguir algo, ya sea un arma o un objeto valioso al que nos pueda permitir avanzar o empeorar la situación.
- Respawn de los enemigos en las salas de forma fija.
Este requisito ha sido elegido a cambio del respawn constante, es decir, que los enemigos ya permanecen en el mapa, a diferencia del otro que la idea era que tu personaje estuviera en un mapa y los enemigos fueran apareciendo.
- Pantalla completa con re-escalado de imágenes.
Este requisito también es nuevo al que se trata de implementar una forma multi-resolución. Se trata de aumentar las imágenes X veces de tamaño respecto la resolución de la pantalla, además de mantener una pantalla completa.

2.2 Objetivos no funcionales:

En esta parte describo los diferentes requisitos que en un inicio se habían planteado para el desarrollo de la aplicación y no se han podido realizar.

- Mapas aleatorios.

Este requisito ha sido retirado por falta de tiempo, pero no queda descartado del todo para futuras actualizaciones. Es necesario una cantidad grande de mapas y diferentes niveles para complementar este requisito.

Se trataría de dentro de una gran cantidad de mapas creados, al empezar la partida solo iniciarías a uno aleatoriamente, clasificándolo por niveles.

- Respawn aleatorio.

Este requisito complementa al anterior.

- Enemigos respawn constante.

3 Tecnologías usadas

3.1 IDE

NetBeans es una app que da entorno principalmente a la programación en Java, aunque también permite otros lenguajes. Es un producto libre y gratuito sin restricciones de uso.

El paquete de NetBeans IDE para Java SE contiene lo que se necesita para empezar a desarrollar plugins y aplicaciones basadas en la plataforma NetBeans; no se requiere un SDK adicional.

La plataforma ofrece servicios reusables comunes para las aplicaciones de escritorio, permitiendo a los desarrolladores centrarse en la lógica de sus aplicaciones.

Podría compararse perfectamente con eclipse ya que contiene las mismas características. Pero elegí NetBeans porque le doy uso diario y por tener cierta experiencia con él no me hace falta necesario otro. También como mi proyecto se trata de un videojuego en vez de recurrir a un app que incluya un motor gráfico, preferí programar el motor gráfico personalmente.

3.2 Editor de mapas

Tiled es un editor de mapas sencillo, pero potente se adapta a cualquier tipo de juego 2D tanto si tu juego es un RPG, un plataformas como un clon del Breakout. Con Tiled tienes todo lo que necesitas.

Es un editor de propósito general que se adapta a todo tipo de proyectos, cuenta con un formato basado en XML llamado **TMX**.

Soporta mapas tanto Ortogonales como isométricos, soporte para objetos con precisión a nivel de pixel que hace que no sea necesario que tu mapa sea basado solamente en tiles. Tiene soporte para copiar y pegar trozos de mapa de manera natural ideal para partes repetitivas y cuenta con deshacer y rehacer fácilmente.

Recarga automáticamente los tileset usados, ideal para cuando se está diseñando o modificando un tileset en un programa de edición de imágenes como Photoshop o GIMP al guardarlo automáticamente se actualiza en el Tiled.

En resumen, es una aplicación gratuita, que facilita a la hora de diseñar los mapas con sus respectivos sprites, devolviéndome un fichero json o otros formatos, pero en mi caso uso el json. Por temas económicos y facilidad en su uso, elegí esta aplicación.

3.3 Herramienta de diseño.

Photoshop es otra de las app que he usado para el entorno grafico del videojuego como mapas, personajes, etc.

Se pueden editar y componer imágenes de mapa de bits en múltiples capas y soporta máscaras, composición alfa y varios modelos de color incluyendo RGB. Soporta infinidad de formatos de imagen, incluso diseños de imagen 3D.

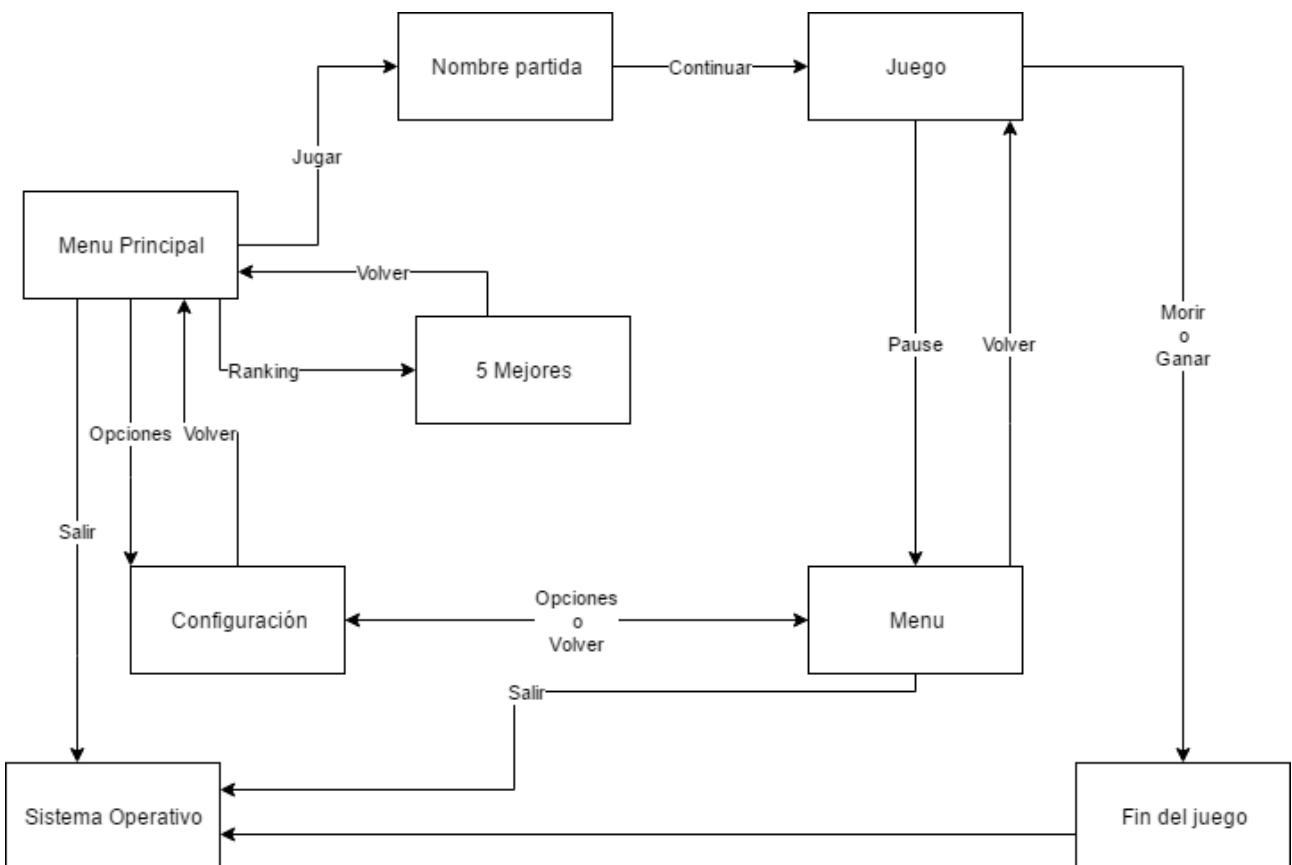
Es comparable con Gimp, que su objetivo es el mismo pero con menos herramientas y de uso gratuito.

Realmente en mi proyecto no le doy un gran uso de todas sus herramientas para mí diseño grafico, pero por experiencia de uso, prefiero el photoshop.

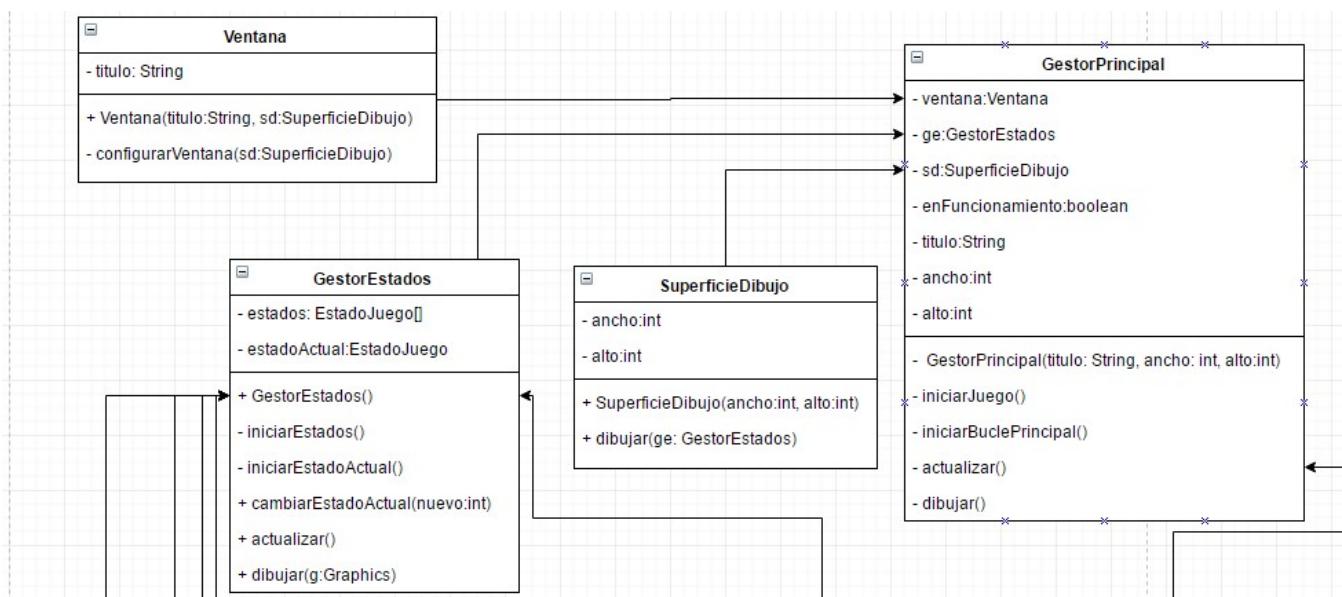
4 Diseño

4.1 Diseño técnico

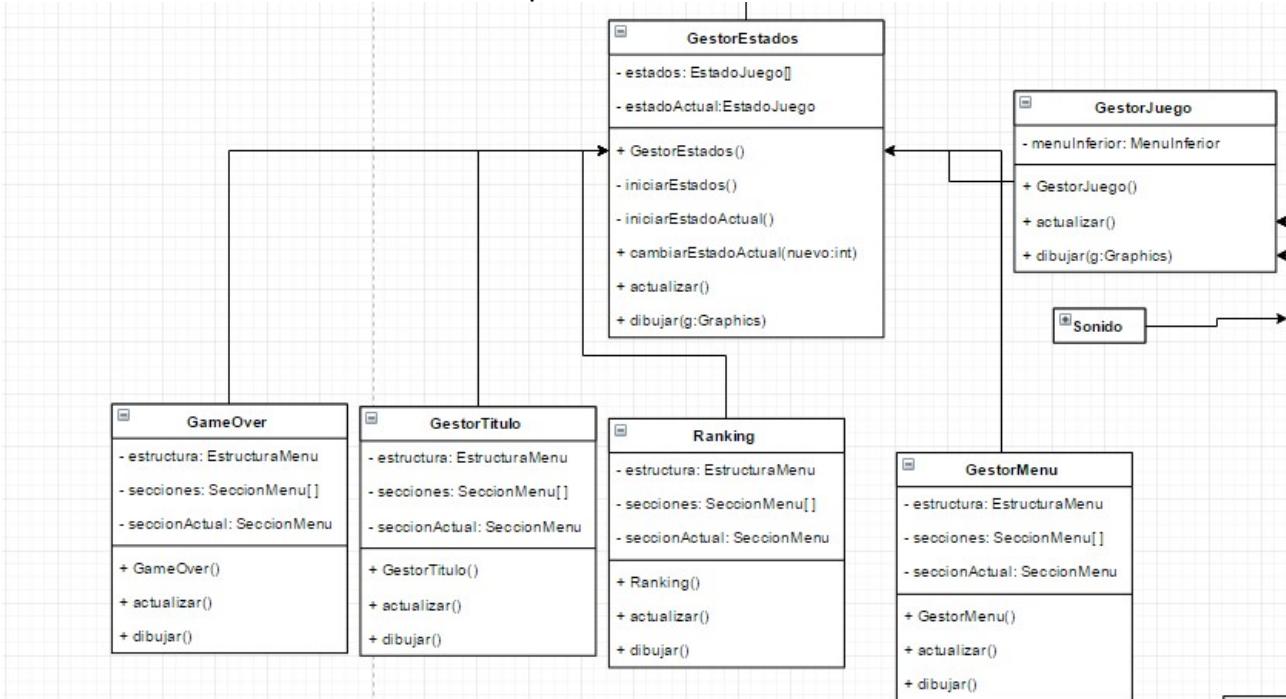
Aquí se puede ver el diagrama de flujo donde se gestión de los estados del juego según las acciones que haga el usuario o sucedan.



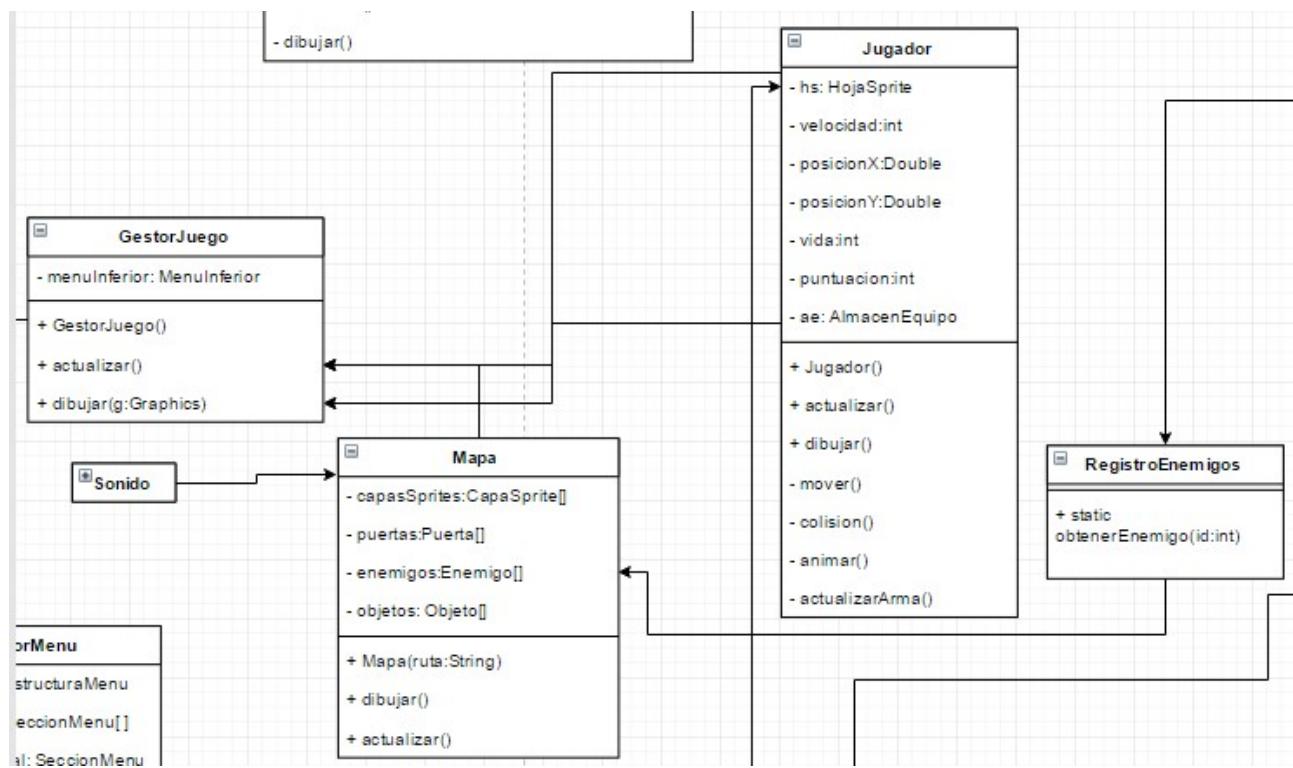
Seguidamente se representa un diagrama de clases donde se muestran las clases más importantes y el funcionamiento de la aplicación.



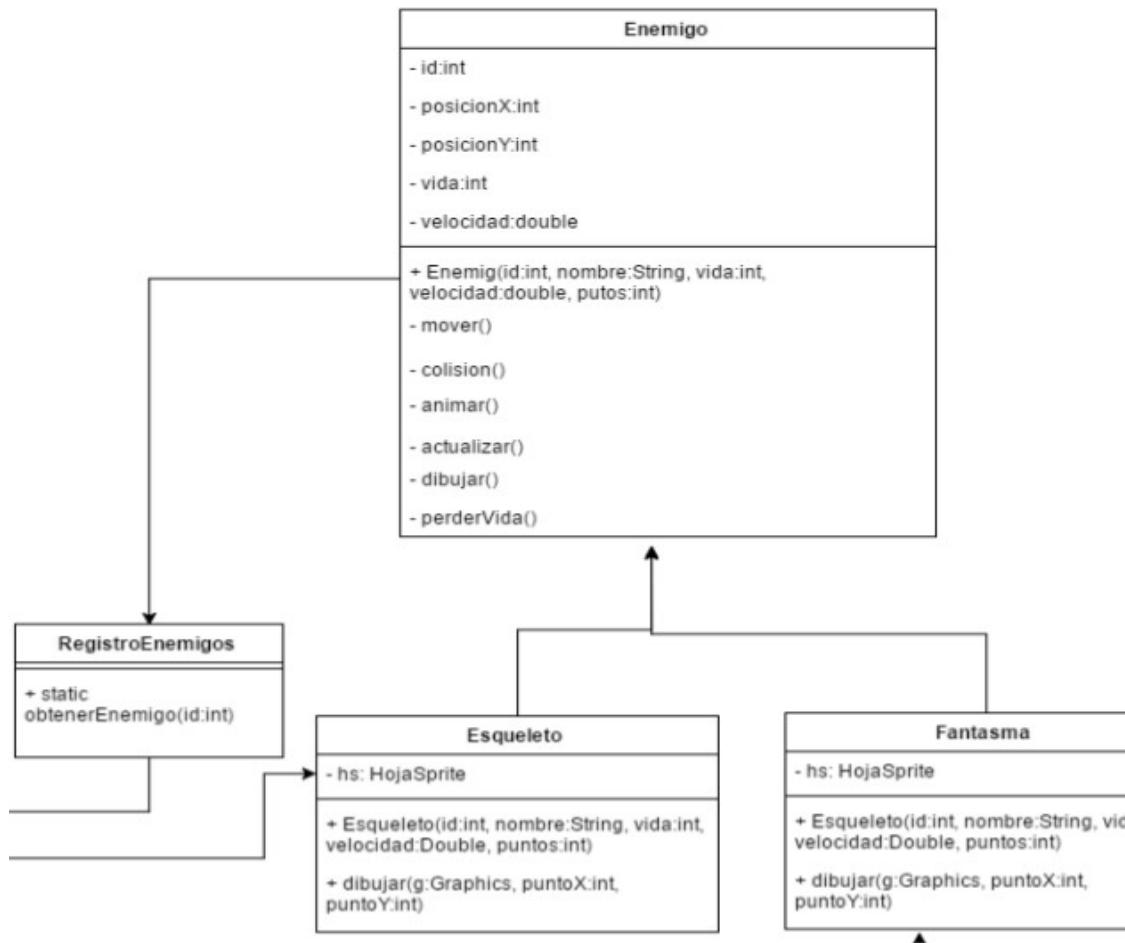
Esto es la continuidad de las clases que usa GestorEstados.



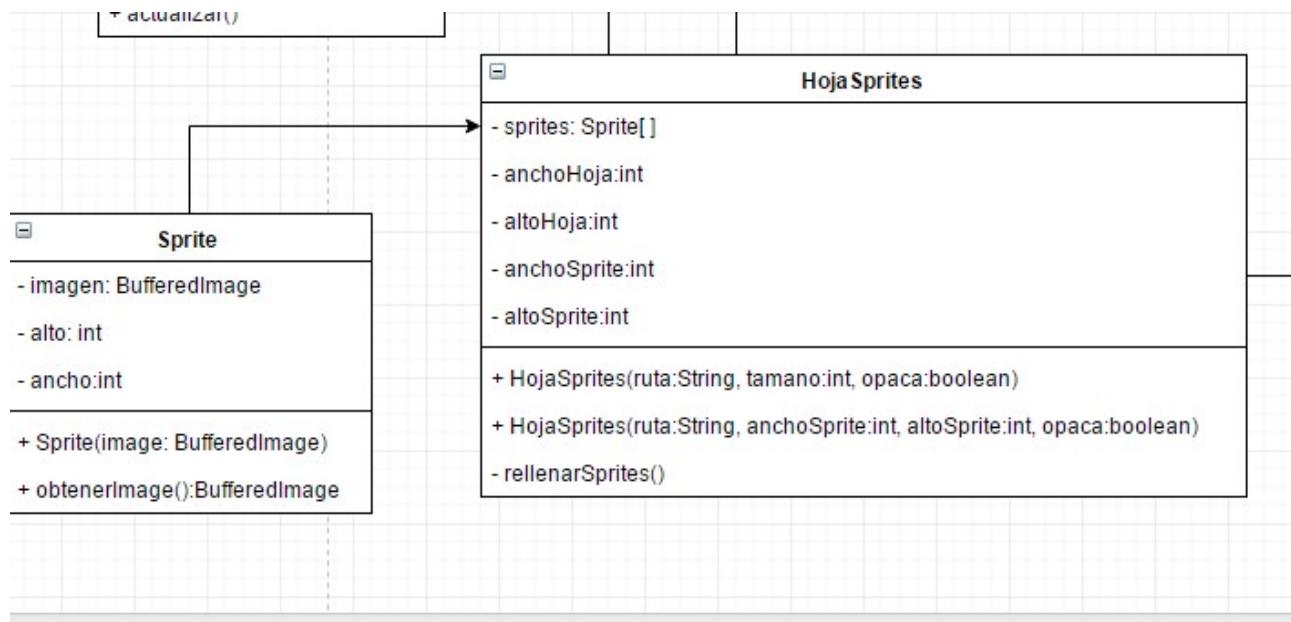
Dentro de los gestores, tenemos el gestorJuego que se encarga de conectar todo lo relacionado con los personajes y mapas, y los actualice y dibuje.



El registro de enemigos donde contiene la superclase enemigo y las subclases de cada uno de los enemigos con sus propiedades particulares.

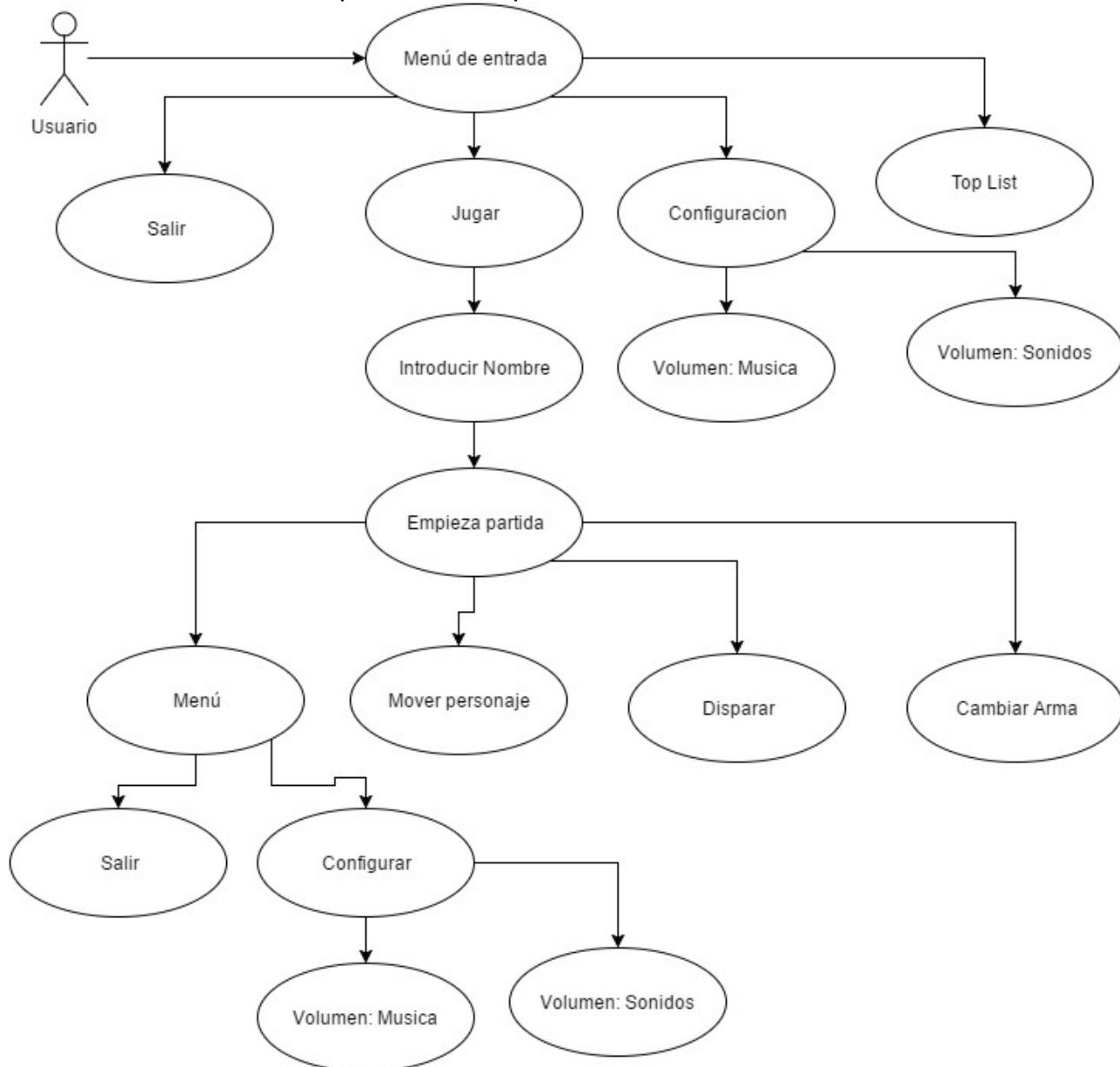


Tanto como los enemigos, como el jugador y el mapa, requiere de una clase encargada de cargar las imágenes para dibujarlas.



Casos de uso

Estos son los casos en el que el usuario podrá realizar acciones.



Uso del Usuario

- Acceder al Menú principal y elegir entre las diferentes opciones.
- Configurar el volumen del sonido y de la música.
- Acceder a las partidas Top 5.
- Jugar al juego.
- Dentro del juego mover el personaje, atacar, coger objetos y equiparlos, interactuar con los enemigos.
- Acceder a un menú y pausar la partida o salir.
- En este menú/pause aparte de salir la partida, tendrá acceso a la configuración para cambiar volumen.

4.2 Diseño GUI

4.2.1 Diseño layout

El videojuego realmente está diseñado en una misma ventana/pantalla.

Es decir, que realmente el juego cuando ejecuta la primera pantalla como sería el caso del menú, si seleccionásemos alguna opción del menú que nos condujese a otra pantalla, como por ejemplo a “Jugar”, borraría todas las imágenes que se estaban mostrando y redibujaría las nuevas imágenes sobre la pantalla.

Cuando ejecutamos el Programa el usuario lo primero que se encontrara es un menú con un título superior y diferentes opciones a las que puede moverse con el teclado, Jugar, Ranking, opciones y salir. Si el usuario seleccionase cada una de las diferentes opciones con el teclado, se encontraría con lo siguiente.

Un ejemplo de la pantalla de menú:

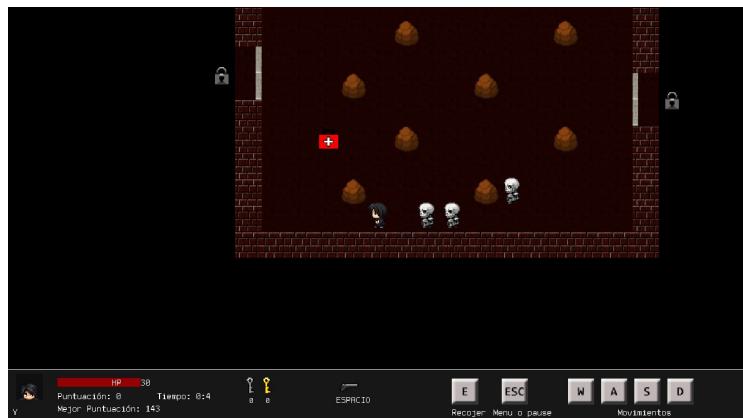


- Jugar:

Seleccionando esta opción nos enviaría a una pantalla donde nos saldría el título del juego y un pequeño campo de texto en el que tiene que escribir el nombre del jugador para poder continuar.

Seguidamente al continuar empezara la partida y apareceremos directamente en el mapa. Los diferentes enemigos que han aparecido, tu personaje en el centro de la pantalla y en la parte inferior de la pantalla una interfaz para el usuario que nos informa sobre la partida.

Un ejemplo:



Dentro de esa interfaz disponemos de información, un avatar del personaje y al lado su vida que dispone actualmente, también se puede ver las puntuaciones que tienes acumuladas y la mejor puntuación que existe en el ranking con lo que nos dará como desafío de superarlo comparando con nuestra puntuación actual.

En la interfaz en la parte más central se pode ver lo que nuestro personaje llevaría equipado, ya sea 2 tipos de llaves, una dorada y otra plateada y la cantidad que tiene y también el arma equipada. Seguidamente tendríamos en la parte izquierda de la interfaz una pequeña guía de los controles que puede hacer el usuario muy visualmente se identificaría de que se trata. Están marcados con imágenes de las teclas del teclado que puede usar y para qué sirven.

Un ejemplo:



En el juego podríamos desplazarnos con el personaje por todo el mapa sin pasar por las rocas ni paredes y mientras el personaje se moviera se visualizaría su animación al andar.

En cualquier momento el usuario desde la partida podría acceder a un menú en el que se encontraría con 2 opciones, "Opciones" con lo que nos llevaría la misma pantalla que la de opciones que está en el menú principal y "Salir" con los que nos permitiría salir y cerrar el juego.

- Ranking

Esta opción nos permite ver una pantalla nueva en el que visualizariámos una lista de partidas que se hayan hecho pero con la excepción de que solamente se mostrarían las 5 mejores con respecto a la puntuación final.

Las partidas en la pantalla están distribuidas en diferentes columnas, por nombre, puntuación y tiempo. El usuario en cualquier momento podría seleccionar volver al menú principal que se indica en la parte inferior de la pantalla.

Un ejemplo:

	Nombre	Puntuación	Tiempo
1	ORI	143	4:25
2	ORI	59	2:21
3	ORI	55	2:19
4	ORIOL	30	1:18
5	OR	0	0:11

Volver

- Opciones

En esta opción seleccionada nos llevaría a otra pantalla en la que nos mostraría 2 posibles opciones de configuración de la aplicación, que nos permite con el teclado A o D, subir o bajar el volumen de las 2 opciones que hay indicadas. Siempre el usuario podría volver al menú principal desde esta pantalla.

Un ejemplo:



- Salir

En esta opción directamente saldríamos de la aplicación.

4.2.2 Diseño visual y contenidos

El diseño del videojuego está ambientado en los juegos antiguos árcades, que se producían para consolas y PC de esa época. El pixel art es un arte digital que se caracteriza por diseñar imágenes a tamaño pixel, es decir todos los tonos, colores, sombras, etc. Es decir, que todas las imágenes están hechas pixel a pixel. Un ejemplo sería el siguiente:



El tamaño de esta imagen aunque se vea más grande y no parece que se vea bien los cuadraditos que representarían cada pixel. Su tamaño original es de 32x32 pixeles. Cada una de las imágenes hechas están diseñadas a través de la herramienta Photoshop que sirve básicamente para todo tipo de diseño de imágenes.

La ambientación del juego como representa estar dentro de una mazmorra donde hay poca luz, ya que es un lugar abandonado donde solo habitan monstruos, se destacan mucho los colores oscuros como, rojo oscuro, negros, marrones, grises. Y para destacar y ver bien los enemigos en respecto la ambientación, se usan tonos de colores como el blanco para fantasmas o esqueletos, etc.

Las animaciones como las de los enemigos o el jugador, están representadas por una serie de imágenes diseñadas previamente que según el movimiento y dirección que haga se ejecutaran una serie de imágenes. Un ejemplo del diseño de un enemigo sería esta hoja llena de Sprites con la que representaría cada una de las situaciones del enemigo.



El texto del juego también se usa una fuente en la que las letras están hechas de pixel para concordar con el estilo del juego en general.

¿Porque la elección de este diseño pixel art?

Me parece más divertido y fácil de realizar. Ya que el juego primeramente esta hecho en 2D, eso significa que su diseño no tiene profundidad, es totalmente plano. Segundo porque un juego nunca podrá llegar a ser real, por lo que me parece más atractivo que sea un dibujo ya que destaca que es pura ficción.

Una de las partes a destacar también de un videojuego es la banda sonora y los efectos de sonido que lo componen. El juego dispone de 2 músicas de fondo en las que una daría entrada el juego en el menú y luego dentro del juego en la partida otra música sonara infinitamente. Es decir, que una vez termina vuelve empezar. Por otra parte están los efectos de sonido como los sonidos al disparar el arma, cuando un enemigo te golpea el personaje se queja, el boss cuando lanza bolas de fuego, o un lanzallamas, etc.

5 Desarrollo

5.1 API JSON

En mi proyecto para generar los mapas con sus enemigos, objetos, colisiones, etc. Como explique en herramientas de uso, utilizo una herramienta editor de mapas, que me permite generar mapas con mis hojas de Sprites y con ello me devuelve un fichero json. Para facilitar su uso y traducir ese json en java a ser un mapa o extraer los datos del json de forma más fácil, doy uso de una librería json-simple-1.1.1.

Con ella divido las partes del código del json en arrays que luego me permitirán generar el mapa.

Para importar esta librería a Netbeans, se accede desde la IDE nuestro proyecto y en la parte de libraries con el botón derecho Add jar/folder y buscamos nuestra librería descargada y descomprimida que se tratará de un .jar y se importará a nuestro proyecto.

Como el json trata con objetos, esta api se encarga de recoger cada parte del código obtener su información y parsearla a objetos de datos para java. Y luego de obtener este objeto con todos los datos de nuestro fichero, se podrían tratar buscando por identificaciones de string o dobles, etc.

En mi caso para generar el mapa primeramente busco mi fichero pasándole la ruta del json. Seguidamente guardo en un JSONObject todo el contenido del fichero. Luego ya es cuestión de ir dividiendo las partes del contenido y extraer lo que nos interesa. Como estoy generando un mapa, necesito obtener el nombre y los tamaños. Para ello utilizando una función de la api globalJSON.get("nameMap").toString() esta función lo que nos hace es desde el objeto que contiene todos los datos del mapa json que es globalJSON con un get y que contenga de nombre "nameMap" nos devolverá el valor de ese identificador en formato string. En el caso de obtener los tamaños del mapa que son necesarios para luego ir colocando las imágenes por la pantalla, para ello usamos la misma función pero buscando por heigh y width, obteniendo en formato string, parsear-lo a int. Un ejemplo seria este (int) Double.parseDouble(objetoJSON.get(clave).toString()). A partir del objetoJSON que sería nuestro contenido de todo el json, buscamos la clave que podría ser heigh y nos devolvería un string. Seguidamente se parsea a doublé ya que la api solo permite uso de doublés, pero como era necesario un int, desde Java se parsea el doublé a int. Pero esto no es todo, una de las mejores cosas de JSON, esque se guardan arrays de objetos, por lo que en esta API también contiene una clase JSONArray en la que nos permite guardar arrays de objetos. Un ejemplo de una función que uso para extraer y llenar un array de objetos json es:

```
private JSONArray obtenerJSONArray(final String codigoJSON) {
    JSONParser lector = new JSONParser();
    JSONArray arrayJSON = null;

    try {
        Object recuperado = lector.parse(codigoJSON);
        arrayJSON = (JSONArray) recuperado;
    } catch (ParseException e) {
        System.out.println("Posicion: " + e.getPosition() + " " + e);

    }
    return arrayJSON;
}
```

Esta función básicamente se encarga de a partir de un codigoJSON que se trataría de un string del nombre del campo a buscar que contenga un array de objetos y parsear todo el array.

En este ejemplo se puede ver donde llamo el método anterior y le paso por parámetros una clave que en este caso sería un identificador “layers” que se buscaría en el fichero json y se traería todo el array que contenga ese objeto.

```
JSONArray capas = obtenerJSONArray(globalJSON.get("layers").toString());
```

En mi caso, layers, contiene un array de todas las id de sprites colocadas en un array de ints, también otro array de otros sprites que contendría una 2n capa del mapa y finalmente un array de objetos recuadrados, que son areas rectangulares que representarían las colisiones del mapa.

6 Testing

Descripción de las pruebas realizadas: tipo de prueba, objetivo de la prueba y resultado.

Tipo de prueba:	Ejecución y muestra de la ventana.
Objetivo:	Que se muestre la ventana JFrame.
Acciones:	Ejecutar el programa.
Resultado:	Correcto.

Tipo de prueba:	Gráficos y velocidad de ejecución.
Objetivo:	Mostrar imágenes en la ventana, los aps y fps.
Acciones:	Ejecutar el programa.
Resultado:	Correcto.

Tipo de prueba:	Creación de mapas.
Objetivo:	Mostrar un mapa con sus sprites correctamente.
Acciones:	Ejecutar el programa.
Resultado:	Correcto.

Tipo de prueba:	Graphicos y velocidad de ejecución.
Objetivo:	Mostrar imágenes en la ventana, los aps y fps.
Acciones:	Ejecutar el programa.
Resultado:	Correcto.

Tipo de prueba:	Movimientos en el mapa.
Objetivo:	Mover una imagen central, que en el futuro será el personaje.
Acciones:	Pulsar las teclas WASD.
Resultado:	Correcto.

Tipo de prueba:	Diseño del personaje y animaciones.
Objetivo:	Mostrar el personaje y animar cuando se mueve por el mapa.
Acciones:	Pulsar las teclas WASD.
Resultado:	Correcto.

Tipo de prueba:	Comprobar que el personaje colisione con las paredes y rocas.
Objetivo:	Las paredes y rocas impiden el paso del personaje y no pueda avanzar.
Acciones:	Pulsar las teclas WASD y chocar contra una pared del mapa.
Resultado:	Correcto.

Tipo de prueba:	Acceso de menús del juego.
Objetivo:	Moverse por menus y acceder a las diferentes opciones.
Acciones:	Pulsar ENTER sobre la opción “Jugar”. Pulsar WS para subir y bajar las opciones del menú. Pulsar ENTER sobre “opciones”. Dentro de opciones, pulsar AD y modificar valor de “volumen musica”. Dentro de opciones, pulsar WS para subir y bajar las opciones. Dentro de opciones, pulsar ENTER en “Volver”. Pulsar ENTER a “Salir”.
Resultado:	Correcto.

Tipo de prueba:	Pantalla completa y rendimiento.
Objetivo:	Mostrar el juego en pantalla completa con las imágenes re-escaladas sin perder mucho rendimiento.
Acciones:	Ejecutar el programa. Pulsar ENTER en “Jugar”.
Resultado:	Correcto.

Tipo de prueba:	Interfaz de usuario.
Objetivo:	Mostrar en el juego una interfaz en la parte inferior de la pantalla con información de la partida, ya sea: vida, puntuaciones, etc.
Acciones:	Ejecutar el programa. Pulsar ENTER en “Jugar”.
Resultado:	Correcto.

Tipo de prueba:	Pantalla completa y rendimiento.
Objetivo:	Mostrar el juego en pantalla completa con las imágenes re-escaladas sin perder mucho rendimiento.
Acciones:	Ejecutar el programa. Pulsar ENTER en “Jugar”.
Resultado:	Correcto.

Tipo de prueba:	Nombre del jugador/partida y guardado de un .save
Objetivo:	Al iniciar una partida antes de entrar a jugar, pedir un nombre al jugador al que cuando termine la partida se guarde en un fichero.save
Acciones:	Ejecutar el programa. Pulsar ENTER en “Jugar”. Escribir “ORIOL”. Pulsar ENTER para continuar. Pulsar ESC. Pulsar ENTER sobre el menú, en la opción “Salir”.
Resultado:	Correcto.

Tipo de prueba:	Nombre del jugador con carácter español ñ.
Objetivo:	Al pedir escribir un nombre, que acepte el carácter Ñ.
Acciones:	Ejecutar el programa. Pulsar ENTER en “Jugar”. Escribir “ÑÑÑÑ” Pulsar ENTER para continuar.
Resultado:	Correcto.

Tipo de prueba:	Nombre del jugador en blanco.
Objetivo:	No permitir empezar la partida si el nombre está en blanco.
Acciones:	Ejecutar el programa. Pulsar ENTER en “Jugar”. Pulsar ENTER para continuar.
Resultado:	Correcto.

Tipo de prueba:	Enemigos
Objetivo:	Mostrar en el juego los enemigos posicionados correctamente en el mapa.
Acciones:	Ejecutar el programa. Pulsar ENTER en “Jugar”. Escribir “KJSHF”. Pulsar ENTER para continuar.
Resultado:	Se muestran los enemigos correctamente.

Tipo de prueba:	Persecución de los enemigos.
Objetivo:	Que los enemigos persigan al jugador.
Acciones:	Ejecutar el programa. Pulsar ENTER en “Jugar”. Escribir “ORIOL”. Pulsar ENTER para continuar. Pulsar WASD.
Resultado:	Correcto.

Tipo de prueba:	Animaciones y movimientos de los enemigos.
Objetivo:	Mientras lo persiguen, cuando se mueven se ejecute la animación al andar y sus respectivos cambios de dirección.
Acciones:	Ejecutar el programa. Pulsar ENTER en “Jugar”. Escribir “ORIOL”. Pulsar ENTER para continuar. Pulsar WASD.
Resultado:	Correcto.

Tipo de prueba:	Colisiones de los enemigos.
Objetivo:	Los enemigos si se encuentran con una roca o pared, les impida avanzar.
Acciones:	Ejecutar el programa. Pulsar ENTER en “Jugar”. Escribir “ORIOL”. Pulsar ENTER para continuar. Pulsar WASD. Moverse detrás de una roca.
Resultado:	Correcto.

Tipo de prueba:	Cambios entre mapas.
Objetivo:	Poder cambiar un mapa a otro a través de puertas.
Acciones:	Ejecutar el programa. Pulsar ENTER en “Jugar”. Escribir “ORIOL”. Pulsar ENTER para continuar. Pulsar WASD. Moverse hacia una puerta.
Resultado:	Correcto.

Tipo de prueba:	Inventario
Objetivo:	Coger un objeto, arma o consumible del suelo.
Acciones:	Ejecutar el programa. Pulsar ENTER en “Jugar”. Escribir “ORIOL”. Pulsar ENTER para continuar. Pulsar WASD. Moverse hacia un objeto del mapa. Pulsar E.
Resultado:	Correcto.

Tipo de prueba:	Sistema de combate
Objetivo:	Disparar a una dirección e impactar contra un enemigo.
Acciones:	<p>Ejecutar el programa.</p> <p>Pulsar ENTER en “Jugar”.</p> <p>Escribir “ORIOL”.</p> <p>Pulsar ENTER para continuar.</p> <p>Pulsar WASD.</p> <p>Pulsar ESPACIO mirando hacia un enemigo.</p>
Resultado:	Correcto.

Tipo de prueba:	GameOver
Objetivo:	Perder vida hasta llegar a 0 y mostrar la pantalla de fin de juego.
Acciones:	<p>Ejecutar el programa.</p> <p>Pulsar ENTER en “Jugar”.</p> <p>Escribir “ORIOL”.</p> <p>Pulsar ENTER para continuar.</p> <p>Pulsar WASD.</p> <p>Esperar a chocar con un enemigo.</p>
Resultado:	Correcto.

Tipo de prueba:	Sonidos
Objetivo:	Cuando ejecutamos el juego suene una música de fondo. Cuando disparamos ejecute un sonido de disparo y cuando nos golpea un enemigo suene un ruido de queja.
Acciones:	<p>Ejecutar el programa.</p> <p>Pulsar ENTER en “Jugar”.</p> <p>Escribir “ORIOL”.</p> <p>Pulsar ENTER para continuar.</p> <p>Pulsar WASD.</p> <p>Moverse hacia un enemigo.</p>
Resultado:	Correcto.

Tipo de prueba:	Boss
Objetivo:	Al llegar a una sala donde está el boss. En el combate se muestren sus habilidades y sus fases.
Acciones:	Ejecutar el programa. Pulsar ENTER en “Jugar”. Escribir “ORIOL”. Pulsar ENTER para continuar. Pulsar WASD. Moverse hacia la sala del boss. Pulsar WASD. Pulsar ESPACIO para disparar al boss.
Resultado:	Correcto.

7 Seguimiento y control en la ejecución del proyecto

En las evaluaciones o testeos, no solamente he probado toda la aplicación únicamente yo. Siempre se agradece tener algún que otro compañero voluntario a testear la app y opinar sobre su usabilidad y posibles errores o rendimiento en diferentes tipos de ordenadores. En ello he tenido 1 compañero, con el que he podido compartir mi app y me ha comentado sus expectativas y sus opiniones como, el tema de la velocidad del personaje que era muy lenta, la selección del menú que no se podía ver exactamente que opción se estaba marcando, la fuente del texto, que anteriormente no era pixel, la creación del boss, si era muy difícil o demasiado fácil, el rendimiento en un portátil que no contiene tarjeta grafica, etc.

7.1 Trello

Es una web en la que me ha permitido organizar el tiempo y hacer un seguimiento de todos los sprints y tareas a realizar en un plazo determinador. Incluyendo una serie de evaluaciones.

URL del trello: <https://trello.com/b/H2oalWQS/oriol-garcia-storm-bullets>

7.2 GITHUB

Esta es una de las herramientas en la que nos permite tener un control de versiones de nuestra aplicación y tener siempre una copia de seguridad. Cada vez que se actualizaba la aplicación se subía en un servidor de git con un cometario describiendo los cambios de esa versión en la aplicación. En este control de versiones, aun por evitar problemas, siempre tenía otra versión de la app llamada beta, en la que probaba lo nuevo programado, y así en caso de no funcionar no afectaba a la app original.

URL de github: https://github.com/OriolGarciaFores/Projecte_Storm_Bullets

7.3 Incidencias

Una de las incidencias ocurridas en la app, es cuando estaba testeando la muestra de imágenes del videojuego, en el virtualizador IDE netBeans, funcionaba correctamente. En cambio, cuando se creaba el ejecutable compilado a .jar. La aplicación dejaba de funcionar. El problema resultó que, la carpeta donde guardaba las imágenes/Sprites de los personajes o mapa, el nombre contenía una mayúscula. En la IDE eso no importaba, ya que la aplicación al contener la carpeta origen con la mayúscula funcionaba correctamente, pero en el caso del ejecutable. La aplicación cuando se compila, todas las carpetas pasan a ser automáticamente minúsculos sus nombres. Por eso, la aplicación fallaba. La solución fue modificar el nombre que no tuviera mayúsculas.

Otro de los errores más graves que tuve en el momento que me dedicaba a programar los cambios de mapas. En un inicio, no hubo ningún problema, en cuanto llegabas a una puerta con un personaje y colisionaba, mientras los enemigos estuvieran todos muertos pasabas a la sala correspondiente. Pero, como las salas están hechas entre 1-4 puertas. Una de las salas había 2 puertas al mismo nivel de X, Y. Es decir, que la comprobación de si la posición de la puerta con la del jugador fuera \geq o $=$ a una de las posiciones, simultáneamente esas 2 puertas cumplían el mismo requisito. Por lo que accedía a una sala cualquiera de las 2 aleatoriamente. Su solución fue, en los objetos de tipo puerta, en vez de comparar posiciones, cree un rectángulo a cada puerta para saber con qué puerta esta colisionando. Seguidamente obtener el nombre de qué mapa estoy y al destino que lleva esa puerta en concreto. A si, se podía diferenciar de una puerta a otra.

En el momento de crear el tiempo de juego, me dio bastantes problemas, por no tener en cuenta las actualizaciones del juego. Quería crear el tiempo que estabas dentro del juego y calcularlo. Uno de las ideas que me propuse era creando un contador a partir de un bucle, pero no resultó funcional, ya que dependía de la velocidad de ejecución del ordenador. Otro fue, usando un thread con un sleep, pero me relantizaba la aplicación ya que necesitaba un tiempo de respuesta muy rápido para que mostrase en pantalla los segundos. Hasta que me di cuenta de que mi juego si es Frame by Frame, es decir, que se actualiza 60 actualizaciones por segundo. Solamente debía crear un contador para las actualizaciones, luego uno para los segundos y otro para los minutos, quedando de la siguiente manera. Las actualizaciones, cuando llegasen a 60, sumasen 1 segundo y cuando los segundos llegasen a 60 sumar 1 minuto, y así sucesivamente.

8 Costes y planificación

8.1 RoadMap

Sprint	Tareas	Fecha Inic	Fecha fin
Primeras clases de pantalla	Investigar e implementar la ventana, contador fps, hojas de sprites y teclado	01/07/2016	15/07/2016
Gráficos en pantalla	Investigar e implementar imágenes en pantalla, extraer sprites, introducción de mapa, implementación de cuadrados(imágenes 32x32)	15/07/2016	29/07/2016
Gráficos en pantalla v2	Investigar formato para implementar mapas. Implementar mapa dibujado, implementar movimiento.	29/07/2016	05/08/2016
Personaje	Dibujar e implementar animaciones del personaje y movimiento.	05/08/2016	10/08/2016
Colisiones del mapa	Investigación de las colisiones e implementación en el mapa	12/08/2016	22/08/2016
Interfaz de usuario HUD	Implementación de una interfaz donde muestra información de la partida, puntos, vida, tiempo, etc.	02/09/2016	09/09/2016

Menú principal	Investigación de estados de juego e implementación de menús, visualmente.	09/09/2016	23/09/2016
Menú principal mejorado	Investigación a implementar sonidos y música. Añadido opción de configurar volumen. Añadido música de menú inicial, el código.	23/09/2016	13/10/2016
Gráficos en pantalla v3	Implementación de re-escalado a pantalla completa, modo sin bordes, mejorar rendimiento, arreglar errores y testing	04/11/2016	25/11/2016
Interfaz de usuario HUD Mejorado	Arreglar posibles errores de la HUD, re-escalar la HUD, añadir datos.	25/11/2016	02/12/2016
Datos de jugador guardados y mejora de menú principal	Investigación de uso creación, escritura, borrar ficheros. Clases de leer y guardar información del jugador. Implementar nombre del jugador. Mejorar diseño menú principal.	02/12/2016	23/12/2016
Creación Enemigo	Diseño del enemigo. Clase enemigo y sus características. Implementar en el mapa.	23/12/2016	13/01/2017
Inteligencia Artificial	Investigación de IA. Comportamiento del enemigo. Movimiento o animación del enemigo. Interacción enemigo a personaje. Detección a colisiones.	13/01/2017	03/02/2017
Batalla y puntuación	Mejorar la IA y provocar daño al personaje. Estado morir. Añadir animación muerto. Arma al personaje y daño. Vida al enemigo y al personaje. Puntuaciones por muerte.	03/02/2017	17/02/2017
Variedad de enemigos	Diseño de nuevos enemigos. Nuevas características.	17/02/2017	24/02/2017
Sonidos y música	Implementación de banda sonora. Sonidos de combate.	24/02/2017	10/03/2017
Mejora grafica y transición de mapas	Mejorar el diseño grafico, nuevos dibujos, mapas y poder cambiar de mapas.	10/03/2017	31/03/2017

Boss de Sala	Añadido un mapa específico. Un enemigo especial, con características y habilidades especiales. Una IA más inteligente. Diferentes fases.	31/03/2017	14/04/2017
Icono, arreglo de errores, extras y documentación final	Añadir un ícono identificativo de la app. Arreglar posibles errores o optimizar el código. Añadir posibles extras. Y la presentación e informe del proyecto finalizado	28/04/2017	12/05/2017

Respecto al RoadMap final el único cambió en respecto al inicial, es que se ha descartado la creación de mapas aleatorios ya que por falta de tiempo no podía dedicarme hacer más mapas y sin tener gran variedad de mapas, no tendría sentido hacer aleatoriedad en los mapas ya que no habrían. Pero no se descarta como líneas futuras.

Mapas aleatorio	Diseñar y añadir nuevos mapas. Implementación de aparición de mapas aleatorios.	14/04/2017	28/04/2017
-----------------	--	------------	------------

8.2 Coste final

Primeras Clases de pantalla

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	70	€ 175,00
Diseñador grafico	€ 2,70	10	€ 27,00
Scrum Master	€ 3,00	70	€ 210,00
Tester	€ 2,00	10	€ 20,00
		TOTAL Sprint	€ 432,00

Gráficos en Pantalla

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	70	€ 175,00
Diseñador grafico	€ 2,70	0	€ -
Scrum Master	€ 3,00	70	€ 210,00
Tester	€ 2,00	10	€ 20,00
			TOTAL Sprint € 405,00

Gráficos Pantalla v2

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	30	€ 75,00
Diseñador grafico	€ 2,70	0	€ -
Scrum Master	€ 3,00	30	€ 90,00
Tester	€ 2,00	10	€ 20,00
			TOTAL Sprint € 185,00

Personaje

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	10	€ 25,00
Diseñador grafico	€ 2,70	35	€ 94,50
Scrum Master	€ 3,00	40	€ 120,00
Tester	€ 2,00	1	€ 2,00
			TOTAL Sprint € 241,50

Colisiones

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	45	€ 112,50
Diseñador grafico	€ 2,70	0	€ -
Scrum Master	€ 3,00	48	€ 144,00
Tester	€ 2,00	3	€ 6,00
		TOTAL Sprint	€ 262,50

HUD

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	39	€ 97,50
Diseñador grafico	€ 2,70	0	€ -
Scrum Master	€ 3,00	40	€ 120,00
Tester	€ 2,00	1	€ 2,00
		TOTAL Sprint	€ 219,50

Menú principal

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	70	€ 175,00
Diseñador grafico	€ 2,70	8	€ 21,60
Scrum Master	€ 3,00	80	€ 240,00
Tester	€ 2,00	2	€ 4,00
		TOTAL Sprint	€ 440,60

Menu principal mejorado

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	90	€ 225,00
Diseñador grafico	€ 2,70	0	€ -
Scrum Master	€ 3,00	96	€ 288,00
Tester	€ 2,00	6	€ 12,00
			TOTAL Sprint € 525,00

Gráficos en pantalla v3

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	110	€ 275,00
Diseñador grafico	€ 2,70	0	€ -
Scrum Master	€ 3,00	120	€ 360,00
Tester	€ 2,00	10	€ 20,00
			TOTAL Sprint € 655,00

HUD mejorado

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	39	€ 97,50
Diseñador grafico	€ 2,70	0	€ -
Scrum Master	€ 3,00	40	€ 120,00
Tester	€ 2,00	1	€ 2,00
			TOTAL Sprint € 219,50

Datos guardados

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	115	€ 287,50
Diseñador grafico	€ 2,70	0	€ -
Scrum Master	€ 3,00	120	€ 360,00
Tester	€ 2,00	5	€ 10,00
			€ 657,50
		TOTAL Sprint	

Crear Enemigo

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	115	€ 287,50
Diseñador grafico	€ 2,70	20	€ 54,00
Scrum Master	€ 3,00	120	€ 360,00
Tester	€ 2,00	5	€ 10,00
			€ 711,50
		TOTAL Sprint	

IA

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	115	€ 287,50
Diseñador grafico	€ 2,70	20	€ 54,00
Scrum Master	€ 3,00	120	€ 360,00
Tester	€ 2,00	5	€ 10,00
			€ 711,50
		TOTAL Sprint	

Batalla y puntuación

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	70	€ 175,00
Diseñador grafico	€ 2,70	20	€ 54,00
Scrum Master	€ 3,00	80	€ 240,00
Tester	€ 2,00	10	€ 20,00
			€ 489,00
			TOTAL Sprint

Variedad de enemigos

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	10	€ 25,00
Diseñador grafico	€ 2,70	30	€ 81,00
Scrum Master	€ 3,00	40	€ 120,00
Tester	€ 2,00	10	€ 20,00
			€ 246,00
			TOTAL Sprint

Sonidos y música

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	20	€ 50,00
Diseñador grafico	€ 2,70	0	€ -
Scrum Master	€ 3,00	80	€ 240,00
Diseñador de Audio	€ 2,80	70	€ 196,00
Tester	€ 2,00	10	€ 20,00
			€ 506,00
			TOTAL Sprint

Mejora grafica y transición de mapas

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	110	€ 275,00
Diseñador grafico	€ 2,70	100	€ 270,00
Scrum Master	€ 3,00	120	€ 360,00
Diseñador de Audio	€ 2,80	0	€ -
Tester	€ 2,00	20	€ 40,00
			TOTAL Sprint € 945,00

Boss de sala

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	70	€ 175,00
Diseñador grafico	€ 2,70	50	€ 135,00
Scrum Master	€ 3,00	70	€ 210,00
Diseñador de Audio	€ 2,80	0	€ -
Tester	€ 2,00	5	€ 10,00
			TOTAL Sprint € 530,00

Icono, arreglo de errores extras

ROL	PRECIO/HORA	Horas trabajo	TOTAL
Programador	€ 2,50	70	€ 175,00
Diseñador grafico	€ 2,70	70	€ 189,00
Scrum Master	€ 3,00	70	€ 210,00
Diseñador de Audio	€ 2,80	0	€ -
Tester	€	40	€

	2,00		80,00
		TOTAL Sprint	€ 654,00

Costes de app

App	PRECIO	PRECIO/Mes	Can. Meses
NetBeans	€ -	€ -	Todos
Photoshop		€ 29,99	7
MAGIX Music Maker	€ 59,99		
Total App		€ 269,92	

TOTAL: 9.306,02€ sin IVA

TOTAL + IVA: 11.167,22€.

La única variación con respecto el precio inicial se ha descontado **395€** del total ya que no se ha implementado la generación de mapas aleatorios que para ello también implicaba muchas horas de diseño.

9 Conclusiones

La conclusión que he sacado en este proyecto, es que nunca trabajes solo. Siempre necesitas alguien que opine, pruebe o te ayude en cómo mejorar. En un inicio decidí hacer este juego como hobby pero teniendo en cuenta que me gustan los videojuegos y por el poco tiempo que hubiera dedicado como hobby, la única forma era obligándome hacerlo como trabajo final. Y con eso concluyo que si haces un trabajo que te gusta, saldrá mucho mejor y intentas hacer lo mejor para que sea bueno.

Otra conclusión que he sacado es que nunca se puede hacer las cosas en 2 días. Y que las cosas tampoco se solucionan al momento, habido veces que podido solucionar incidencias en el mismo día, y otras veces al cabo de 1 semana en un momento que no me esperaba poder conseguirlo. Por eso siempre hay que estar relajado y centrado en el

momento que te surge un problema y quieres solucionarlo, si estas relajado acabas sacando la solución fácilmente.

10 Mejoras y líneas futuras

¿Qué mejorarías en la aplicación que has desarrollado?

Toda aplicación tiene posibilidad de optimización así que una de las mejoras futuras es optimizar el código. Como por ejemplo añadir una pantalla de carga antes de iniciar la partida, ya que actualmente cuando inicias el juego, se carga al momento las imágenes y podría llegar a hacer algún bloqueo si hubiera una gran cantidad de imágenes. Otra mejora seria, en vez de cargar un mapa cada vez que accedo a una puerta y cambio de mapa, como el juego está desarrollado de forma en niveles/pisos, que está compuesto de una serie de mapas. Se podría optimizar de forma que antes de iniciar el juego, cargue todo el nivel, es decir los 5-6 mapas que tenga que desplazarse el personaje, para si no tener que estar constantemente cargando los mapas, cada vez que traviese una puerta.

¿Líneas futuras de desarrollo?

A parte de optimizar el código. Cosas nuevas a implementar, una de ellas es el sprint de mapas aleatorios que ha sido descartado, me gustaría implementarlo. Dentro del tema de mapas aleatorios, los mapas que tenga que diseñar, tengan diferentes biomas. Es decir, que ahora mismo el juego se inicia en una cueva oscura, si bajásemos a otros niveles/pisos, adentrándonos más a la mazmorra, otro tipo de biomas serían, un ambiente de hielo, uno húmedo con aguas subterráneas, otro con fuego o magma, etc. Entre otras cosas nuevos enemigos con respecto al bioma y bosses.

Una de las cosas futuras a desarrollar, es la capacidad de multi-lenguaje. Es decir, que desde el menú, poder cambiar diferentes idiomas, catalán, castellano, inglés, etc.

11 ANEXOS

11.1 SKETCH

Menú principal



Opciones



Ranking

Nombre	Puntuacion	Tiempo
<input type="button" value="Volver"/>		

Jugar

Titulo	
Nombre Usuario: <input type="text"/>	

En partida

Mapa de juego				
<input type="button" value="Avatar"/>	<input type="button" value="HP"/> Puntos Mejores Puntos	Llaves	Arma	<input type="button" value="Controles"/>

11.2 Guía del usuario

Guía de instalación

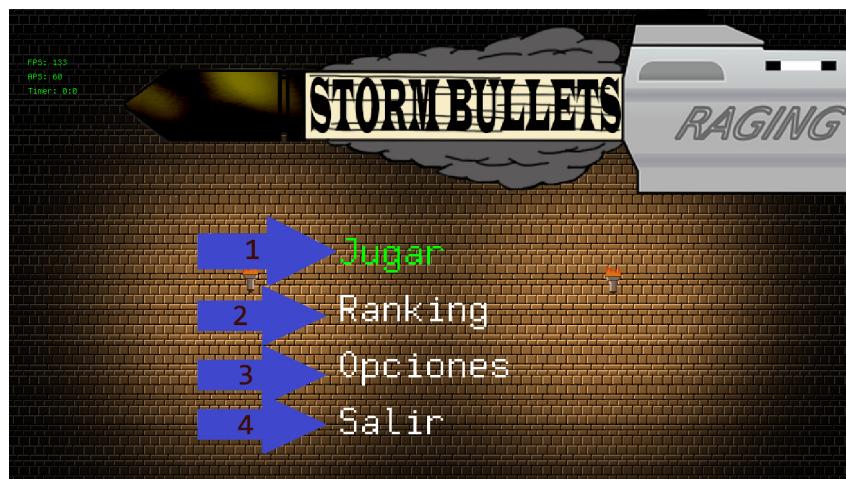
La aplicación no contiene ninguna instalación. Cuando te lo descargas, estará comprimido en un .rar al que necesitaras descomprimir y te devolverá los siguientes archivos.

NOMBRE	RECNA DE MODIRICA...	TIPO	RAMANO
lib	03/05/2017 15:12	Carpeta de archivos	
README.TXT	31/03/2017 17:27	Documento de tex...	2 KB
Storm_Bullets_Beta.jar	31/03/2017 17:27	Executable Jar File	18,808 KB

Es una aplicación portable, es decir, que una vez ejecutamos la aplicación indicada en la anterior imagen, nos ejecutara el programa directamente, sin necesidad de instalar nada.

Guía tutorial del funcionamiento

Menú principal



Al ejecutar la aplicación, la primera pantalla se nos mostrara un menú al que nos podemos mover por las siguientes opciones con el teclado, pulsado W para subir o bien S para bajar.

Para acceder a la opción seleccionada pulsamos ENTER.

Opciones:

- 1- **Jugar:** Empezaría el juego accediendo antes una ventana para nombrar el jugador.
- 2- **Ranking:** Permite ver una lista de las 5 mejores partidas que se han hecho, en base las puntuaciones conseguidas.
- 3- **Opciones:** Permite configurar las diferentes opciones, en este caso el volumen para subir y bajar los sonidos o música.
- 4- **Salir:** Permite cerrar la aplicación.

Jugar

Desde aquí accederíamos a la ventana de Nombrar al jugador.



Aquí podemos escribir nuestro nombre directamente y finalmente pulsar ENTER para continuar y empezar a jugar.



Aquí se ve claramente la pantalla de juego donde tenemos diferentes elementos.

1. Una interfaz de usuario en el que contenemos una pequeña guía de los controles.
 - a. E: para recoger los objetos.
 - b. ESC: para acceder a un menú y pausar la partida.



- c. WASD: las teclas de movimientos.
- d. ESPACIO: para disparar el arma hacia los enemigos.

Dentro de la interfaz, también tenemos el arma con la que llevamos en el inventario equipada. Las llaves disponibles para poder abrir puertas cerradas.

La vida del personaje y puntuaciones.

2. Un botiquín: Se trata de un objeto que nos permite recogerlo y curarnos una cierta cantidad de vida.
3. Aquí se verían el personaje principal y los enemigos a los que hay que derrotar.
4. Esto sería una de las diferentes puertas a las que puedes acceder otras salas para así, poder avanzar en el juego.

También nos podemos fijar en unos candados. Eso significa que todas las puertas de la sala están cerradas. Con lo que no se abrirán hasta que los enemigos hayan sido derrotados.

Cofre



En algunas salas nos podemos encontrar con cofres en los que nos pueden dar aleatoriamente algún arma y hay un 20% de probabilidades que pueda darte 1 consumible extra.

Puertas



Otra de las situaciones que nos podemos encontrar es, con las distintas puertas.

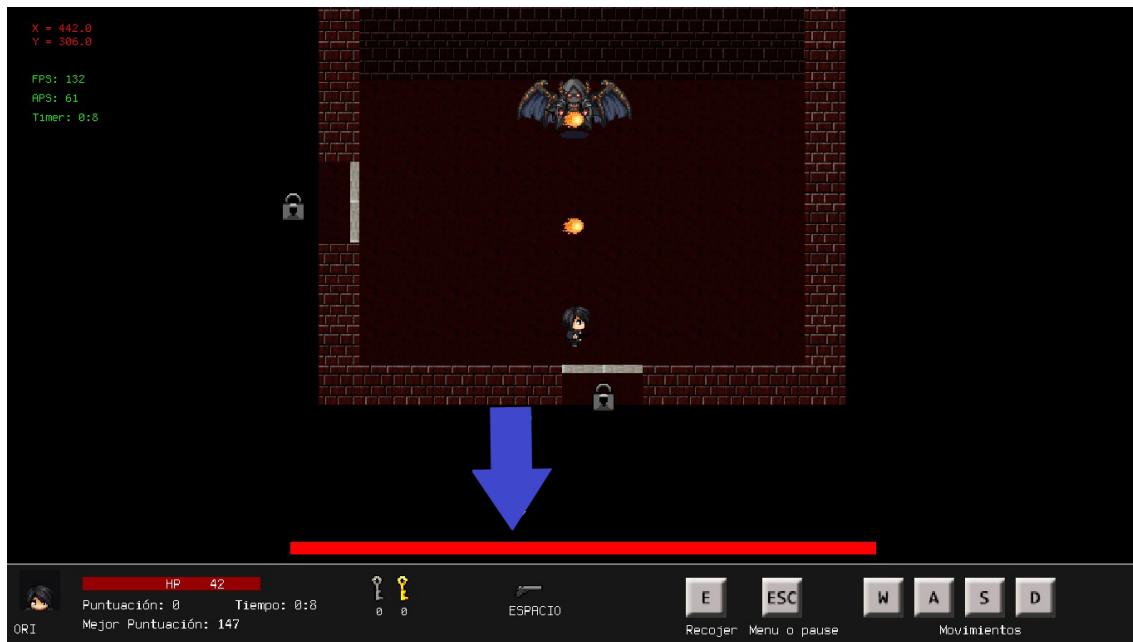
Hay 3 tipos:

Las normales: que podemos acceder a la siguiente sala sin necesidad de nada.

Las con candado: que requiere de 1 llave plateada.

Las con una calavera: que requiere una llave dorada y esta puerta en particular nos llevaría a un enemigo final.

Sala boss



Esta ventana podemos ver la sala de un boss. En lo que se puede ver la barra de vida del boss y hay que vigilar en sus diferentes ataques que pueda tener, como por ejemplo la bola de fuego.

Ranking

	Nombre	Puntuación	Tiempo
1	ORI	147	2:41
2	ORI	65	5:20
3	Y	50	1:3
4	Y 	8	0:12
5	Y	0	0:24

[Volver](#)

En el ranking se podrían ver las diferentes mejores partidas clasificándolas por puntuaciones.

Opciones

En esta sección podemos subir y bajar el volumen de la música y los sonidos del juego, pulsando AD.



11.3 Bibliografía/webgrafía consultada

Portal oficial de Photoshop <<http://www.adobe.com/es/products/photoshop.html>> [20 octubre del 2017]

Pixels Mil. Configurar photoshop para pixel Art

<<http://www.pixelsmil.com/2011/08/configurar-photoshop-para-pixel-art.html>> [3 Noviembre del 2017]

Portal oficial de netbeans <<https://netbeans.org/features/index.html>> [20 Septiembre del 2015]

Java Dev One “Java- Juego de Rol 2D”. Youtube:

<<https://www.youtube.com/playlist?list=PLN9W6BC54TJJr3erMptodGOQFX7gWfKTM>>

Alrededor de unos 100 videos. [Consultado entre Junio-Agosto de 2016].

