

Entrega Final - RetroRace

Proyecto DAM 2017-18

Andreu Juan

Ramón Moreno

Andrés Rodríguez

Índice

Índice	1
Introducción	6
Motivación	6
Descripción del juego	7
Mecánica del juego	8
Descripción y diseño de las pantallas	9
Menú Principal	9
Elección de mapa	9
Pantalla de Juego	9
Ranking	9
Flujo de pantallas	9
Entorno tecnológico	10
Definición del proyecto	10
Módulos del Proyecto	10
Diagrama UML	11
Reparto de tareas	12
Explicación de las clases del cliente	13
Juego	13
Sesión	13
Mapa	13
Casilla	13
Partida	13
Personaje	13
Ranking	13
GUI	13
Cámara de Juego / Game Screen	14
Partes y planificación del proyecto	14
Parte 1	14
Módulos del Proyecto	14
Diagrama UML	15
Reparto de tareas	16
Explicación de las clases del cliente	17
Juego	17

Sesión	17
Mapa	17
Casilla	17
Partida	17
Personaje	17
Ranking	17
GUI	17
Cámara de Juego / Game Screen	17
Parte 2	18
Diagrama UML	18
Parte Servidor	18
Explicación de las clases del Servidor	18
Servidor	18
Out Server	18
Server Multithread	18
Cliente	18
Diagrama Entidad-Relación Base de Datos	19
Diseño y arquitectura	19
Andreu	19
Diagrama	19
Explicación de las clases	20
Partida	20
Personaje	20
Server	20
Out Server	21
ServerMT	21
Cliente	21
Ramon	22
Diagrama	22
Diseño final de las clases	23
Explicación de las clases	24
Juego	24
Sesión	24
GUI	24
DB Manager	24
Ranking GUI	24
Gamescreen	24
Diseño de Bases de Datos	25
Explicación estructura Base de Datos	25

Users	25
Ranking	25
Andrés	26
Diagrama	26
Explicación de las clases	27
Mapa	27
Casilla	27
Otros	27
Implementación y código	27
Andreu	27
Llistat amb breu descripció de cadascun dels arxius de codi font.	27
Partida (Thread)	27
Personaje (Thread)	28
Timer (Thread)	29
Server	29
ClientProject	30
OutServer (Thread)	30
ServerProject	30
ServerMT (Thread)	30
Client (Thread)	30
ServerFrame	31
Ramon	31
Archivos del código fuente	31
Juego	32
Sesión	32
GUI	32
DB Manager	34
Ranking GUI	34
Gamescreen	34
Script de creación de Bases de Datos	35
Base de Datos en Java: Entidades	35
Otros archivos	36
Clase: Hash	36
Archivo: img/gui/loader.gif	36
Problemas surgidos durante la implementación.	36
Andrés	37
Listado y descripción de archivos	37
Detalles de implementación	38
Casillas	38

Propiedades	38
Cielo	38
Tierra	39
Hierba	39
Antorcha	39
Fin	40
Pinchos	40
Nube	40
Pasarela	41
Plataforma	41
Agua	41
Trampolin	42
Seta	42
Planta	42
Cactus	42
Roca	43
Mapas	43
Instalación y configuración	44
Instalación del entorno de desarrollo	44
Instrucciones para la compilación del proyecto	44
Instalación de la Base de Datos	44
Instrucciones de configuración	44
Manual de usuario	45
Objetivo del juego	45
Objetos	45
Antorcha	45
Pinchos	45
Trampolín	46
Agua	46
Meta	46
Mecánicas	46
Un jugador	46
Dos Jugadores	46
Multijugador en LAN	46
Controles	47
Ranking	47
Personal	47
Global	47
Cliente	48

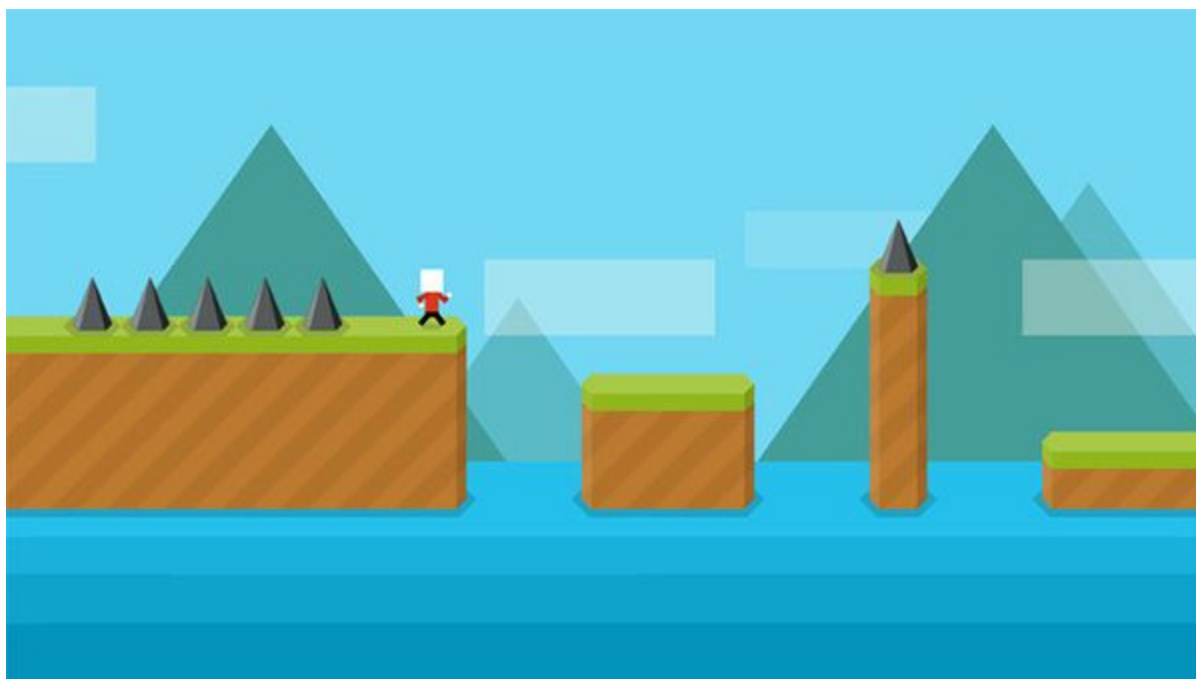


Inicialización	48
Hacer login	48
Registrar un usuario	49
Menú del juego	50
Un jugador	50
Dos jugadores	50
Multijugador en LAN	51
Ranking	51
Servidor	52

Introducción

Durante los dos años de estudio del FP de Desarrollo de Aplicaciones Multiplataforma hemos adquirido una gran cantidad de diferentes conocimientos que esperamos poner en práctica con el desarrollo de este proyecto.

La idea del proyecto consiste en un juego de plataformas individual y multijugador, en el que los jugadores deberán recorrer un mapa a través de una serie de obstáculos para llegar a la meta en el menor tiempo posible.



Motivación

En el desarrollo de este proyecto nuestro objetivo principal sería conseguir desarrollar un juego a la vez que demostramos los conocimientos adquiridos a lo largo de estos dos años, como los threads (que se utilizarán lo largo de todo el juego), sockets (en la conexión de varios jugadores desde diferentes ordenadores) o conexiones a bases de datos.

Además, nos enfrentaremos a un reto bastante complicado que será la animación del juego, por lo que deberemos buscar información para complementar nuestros conocimientos. Pero no solo tendremos retos de programación, ya que otro factor determinante será el trabajo en equipo, por lo que necesitaremos estar comunicados y coordinados ya que intentaremos dividir el trabajo en diferentes partes de manera que, al juntarlas todas, encajen y funcionen correctamente entre sí.

Por ese motivo sabemos que una parte vital en el desarrollo de un proyecto es la planificación. Queremos planificarlo todo bien desde un principio, desde un diagrama UML de todo el programa hasta la base de datos que guardará los tiempos de los jugadores. Al igual que tener listas todas las imágenes o sprites que vayamos a usar para el diseño de los niveles y personajes de los jugadores. Esperamos que con una correcta planificación el desarrollo del proyecto sea más sencillo y estructurado, de manera que no debamos hacer cambios en el diseño, o en el caso de tener que hacerlos, que estos cambios sean mínimos.



En cuanto al proyecto en sí, la idea de poder realizar nuestro propio juego es algo que nos motiva y emociona, por lo que esperamos que en el resultado del proyecto se vean reflejadas todas las ganas y el esfuerzo que depositaremos en el proyecto.

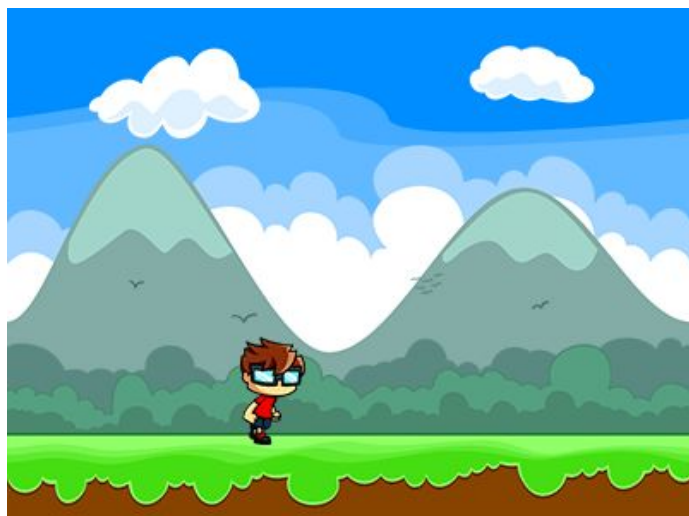
Descripción del juego

Como ya hemos dicho, el objetivo del juego es que el jugador supere diferentes elementos como pinchos que te puedan matar, caídas al vacío y otros elementos que dificulten el recorrido con el objetivo de llegar a la meta en el menor tiempo posible.

Nuestra idea es que puedan jugar varios jugadores el mismo mapa a la vez desde diferentes ordenadores, y que cada jugador pueda elegir si juega con teclado o con un mando conectado a su ordenador. Sin embargo, un jugador también podría elegir jugar solo para intentar



mejorar sus tiempos en un mapa.



Si un jugador decide jugar de manera individual, su objetivo será completar los mapas en el menor tiempo para aparecer en el ranking global, mientras que si opta por jugar en el modo multijugador además de competir por entrar en el ranking competirá en tiempo real contra otros jugadores, lo que creará

una experiencia de juego más desafiante y divertida.

Para ello contaremos con un sistema de registro de jugadores y de sus mejores tiempos en cada mapa de manera que cualquier jugador pueda acceder a los rankings para conocer los mejores tiempos e intentar batirlos.

Además, el juego contará con animaciones en los personajes y sonidos para que la experiencia de juego sea más entretenida. En cuanto a la realización del mapa nos basaremos en los juegos “tile games” con scroll lateral, de manera que el jugador siempre pueda ver a su personaje en el centro de la pantalla mientras se mueve a lo largo del mapa.

Ver: <https://goo.gl/uRT7by>

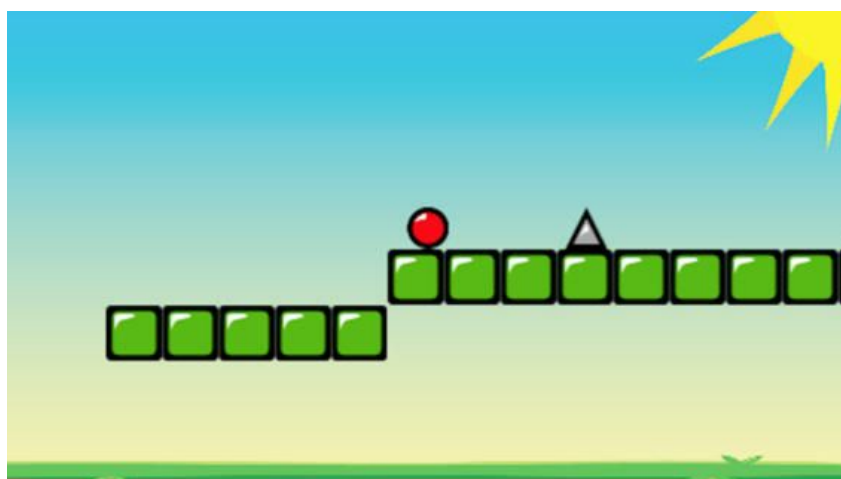
Mecánica del juego

Una vez iniciado el juego y que el jugador se haya conectado con su usuario podrá elegir el modo de juego: individual o multijugador. El jugador podrá elegir entonces en qué mapa desea jugar y entonces comenzará el juego.

En el juego, el jugador controlará a un personaje mediante el teclado o un mando de juego. De esta manera podrá controlar los movimientos del personaje entre los que se encuentra el desplazamiento lateral y el salto. (Mando no realizado)



Una vez empezada la partida, el jugador se encontrará en el mapa que haya seleccionado y deberá controlar a su personaje para llegar a la meta esquivando diferentes obstáculos en el menor tiempo posible.



Descripción y diseño de las pantallas

Menú Principal

Después de que el jugador se haya conectado con su usuario accederá a un menú sencillo en el que podrá elegir el modo de juego: Individual o Multijugador, o ver los Rankings de los mejores tiempos.

Elección de mapa

Si el jugador decide empezar una partida podrá elegir en qué mapa desea jugar, en esta pantalla aparecerán los mapas y el jugador podrá seleccionar en el que quiere empezar una partida.

Pantalla de Juego

La pantalla de juego se visualizará durante el desarrollo de la partida. Lo más importante será el mapa y el personaje al que controla el jugador, además tenemos pensado que se vea el tiempo en una esquina o en algún lugar de la pantalla, para ver cual es el tiempo actual del mapa.

Ranking

Como hemos especificado anteriormente, tenemos pensado un ranking que permita a los jugadores ver sus mejores tiempos, así como los mejores tiempos globales. Intentado mejorar la competitividad del juego, incentivando al jugador a estar en lo más alto de las clasificaciones.

Flujo de pantallas

El juego inicia con una pantalla para que el jugador entre con su usuario. Si no tiene cuenta podrá acceder a una pantalla de registro. Una vez que el usuario haya accedido se encontrará con la pantalla del menú principal, desde la cual podrá acceder a las otras pantallas en función de la opción que seleccione.

Si elige jugar el modo individual le llevará a la pantalla de elección de mapa y, tras elegir un mapa, el jugador se encontrará en la pantalla de juego. Si elige la opción de jugar multijugador el flujo de pantallas será similar al modo individual con la excepción de que entrará en una sala para esperar que se conecten otros jugadores y elegir el mapa.

Desde el menú principal también podrá acceder a las puntuaciones del ranking, explicadas en el apartado anterior.

Del mismo modo, mientras el jugador esté en la pantalla de juego también podrá acceder a un menú de pausa desde el cual podrá volver al menú principal.

Entorno tecnológico

Usaremos Java para llevar a cabo el juego, ya que nuestro objetivo es intentar hacer un juego lo más flexible a las diferentes plataformas, y como es obvio Java es un buen candidato para conseguir tal objetivo.

Adicionalmente usaremos MySQL para la base de datos. Dónde tendremos los registros de los usuarios del juego.

Además queremos dar compatibilidad al mando de la Xbox propio de microsoft para intentar cautivar a aquellos jugadores que prefieran el mando por si así se sienten más confiados.

Definición del proyecto

Módulos del Proyecto

Para realizar la planificación del proyecto hemos dividido el proyecto en dos “grandes” partes:

- **Creación básica del juego:** Crear la base del proyecto para jugar de manera individual, con algunos mapas y que el juego funcione y se vea bien.
- **Creación del servidor, base de datos y multijugador:** Crear una base de datos que almacene los usuarios y el ranking además de crear un servidor en el que se puedan unir varios jugadores para jugar simultáneamente una partida.

Hemos decidido dividir el proyecto en estas dos partes para no mezclar tareas y asegurarnos que por lo menos conseguiremos tener el juego individual. Una vez que nos aseguremos que el juego funciona bien entonces implementaremos el segundo módulo con las bases de datos y el servidor. De esta manera nos aseguramos que si el proyecto resulta ser muy grande y no darnos tiempo a realizarlo todo, por lo menos la parte individual sí que funcionará.

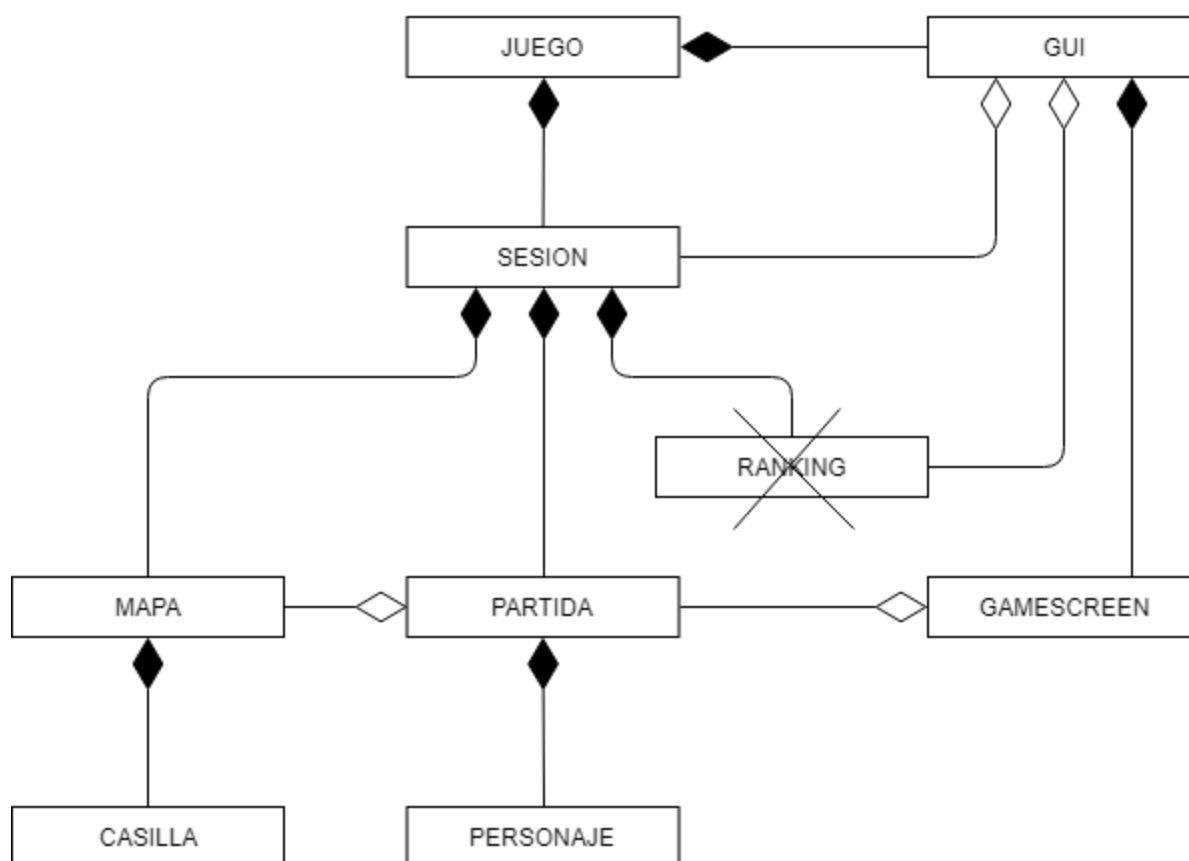
Además, cada una de estas grandes partes será subdividida en pequeños módulos, de manera que nos podamos repartir el trabajo de forma equitativa en la realización de cada módulo. Para hacer este reparto, primero hemos planteado un diagrama UML inicial para realizar el juego de manera básica.

Por ese motivo, este documento corresponde a la planificación de la primera parte: **Creación básica del juego**, y una vez que funcione correctamente pasaremos a la planificación de la segunda parte.

Diagrama UML

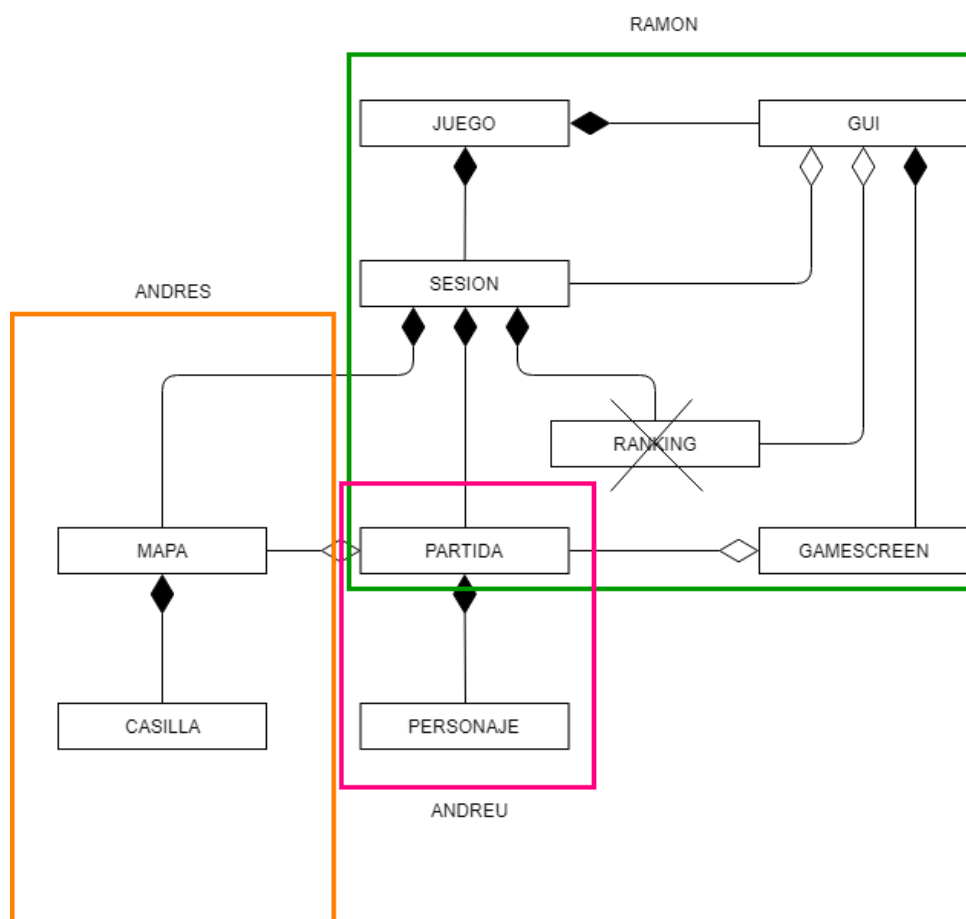
Para realizar nuestro proyecto de un juego en el que los jugadores deberán recorrer un mapa a través de una serie de obstáculos para llegar a la meta en el menor tiempo posible hemos planteado el siguiente diagrama UML.

Recordemos que este diagrama contiene lo básico para que el juego funcione para un jugador y, posteriormente, incorporaremos las clases necesarias para que funcione en modo multijugador.



Reparto de tareas

Basándonos en el UML anterior, nos hemos dividido el trabajo en función de trabajar en clases que estén relacionadas y que tengan un tiempo de trabajo similar. En cualquier caso, siempre estaremos dispuestos a ayudarnos si algún miembro del equipo necesita ayuda.



Andrés → Realizará la generación de los mapas con sus respectivas casillas, definiendo sus propiedades entre otras cosas.

Ramón → Encargado de la interfaz gráfica de la gestión del juego, la sesión e interfaz gráfica donde se desarrollará el juego.

Andreu → Llevará a cabo la lógica de la partida así como la del personaje, interacciones, movimientos, etc.

Explicación de las clases del cliente

Juego

Esta clase se encarga de iniciar el programa y crear la interfaz gráfica del juego.

Sesión

La clase sesión se crea al introducir un nombre de usuario y contraseña registrados en la base de datos (aunque esto no se implementará hasta la segunda fase del proyecto). Esta clase gestionará las partidas y puntuaciones que obtenga el jugador en dicha sesión.

Mapa

La sesión contiene una lista de mapas que el jugador puede elegir cuál jugar de manera individual. Cada mapa está compuesto por una serie de casillas que unidas en conjunto forman todo el mapa.

Casilla

Al basarnos en “tile-game”, nuestro mapa estará formado por diferentes casillas que contendrán los atributos de dicha casilla, como por ejemplo, si es una casilla que hace daño o es una de inicio o fin del nivel.

Partida

La partida se crea una vez que el jugador a elegido un mapa en el que jugar. Es la clase más compleja ya que es la que se encarga de coordinar el mapa con los movimientos que hace el personaje para luego mostrarse en la pantalla de juego. De esta manera, cada vez que el personaje haga un movimiento la partida determinará hacia donde tiene que ir y cómo interactúa el personaje al entrar en contacto con determinadas casillas del juego, como por ejemplo las casillas de daño.

Personaje

Es la clase que se encarga de gestionar los movimientos que hace el jugador y como debe responder el personaje a dichos movimientos.

Ranking

(Esta clase no se implementará en esta fase) Accede a la base de datos y muestra el ranking con las mejores puntuaciones en una tabla.

GUI

Esta clase es la que se encarga de coordinar todos los elementos gráficos del juego.

Cámara de Juego / Game Screen

Esta clase gestiona lo que ve el jugador durante el desarrollo de una partida.

NOTA: Para la segunda fase del proyecto, tenemos pensado añadir clases como Servidor y Out Server, además de un diagrama UML del propio servidor, sin embargo, y como hemos dicho al principio de este documento, no nos centraremos en la segunda parte del proyecto hasta que la primera parte funcione correctamente.

Partes y planificación del proyecto

Parte 1

Módulos del Proyecto

Para realizar la planificación del proyecto hemos dividido el proyecto en dos “grandes” partes:

- **Creación básica del juego:** Crear la base del proyecto para jugar de manera individual, con algunos mapas y que el juego funcione y se vea bien.
- **Creación del servidor, base de datos y multijugador:** Crear una base de datos que almacene los usuarios y el ranking además de crear un servidor en el que se puedan unir varios jugadores para jugar simultáneamente una partida.

Hemos decidido dividir el proyecto en estas dos partes para no mezclar tareas y asegurarnos que por lo menos conseguiremos tener el juego individual. Una vez que nos aseguremos que el juego funciona bien entonces implementaremos el segundo módulo con las bases de datos y el servidor. De esta manera nos aseguramos que si el proyecto resulta ser muy grande y no darnos tiempo a realizarlo todo, por lo menos la parte individual sí que funcionará.

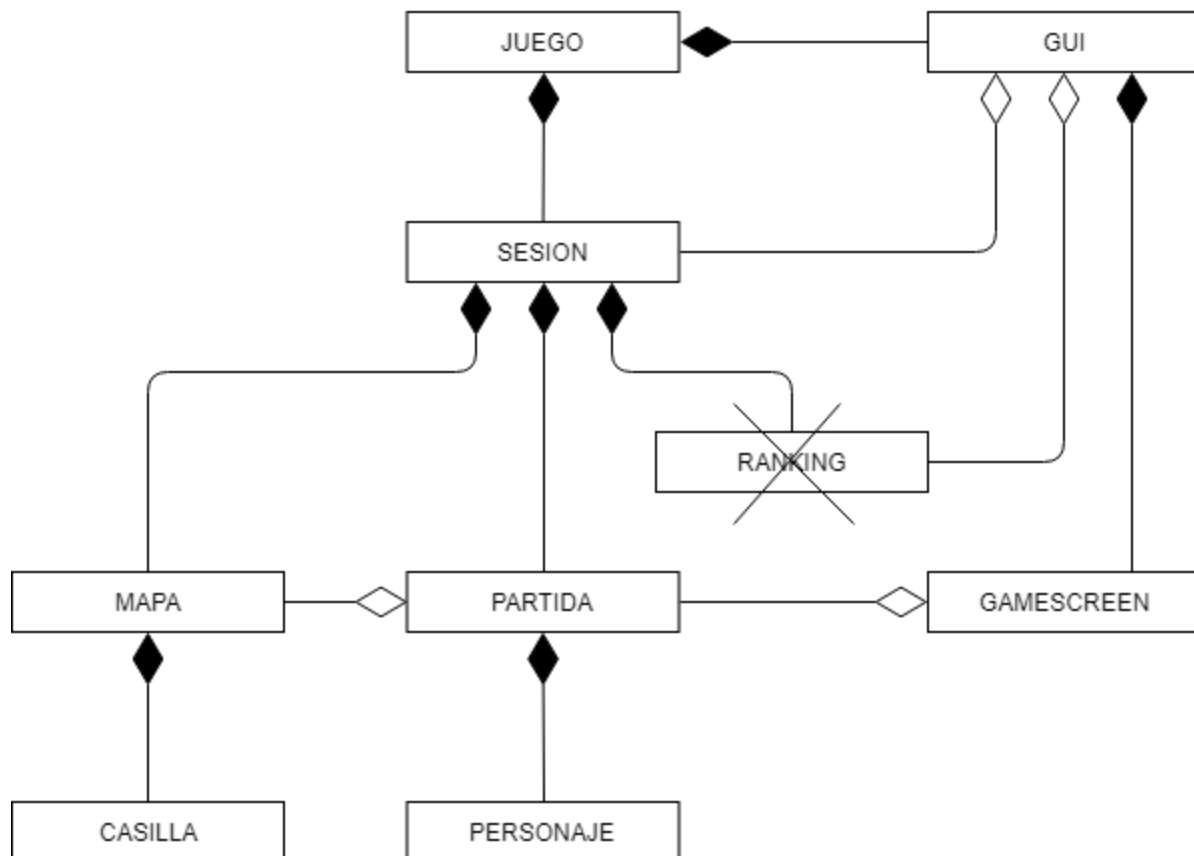
Además, cada una de estas grandes partes será subdividida en pequeños módulos, de manera que nos podamos repartir el trabajo de forma equitativa en la realización de cada módulo. Para hacer este reparto, primero hemos planteado un diagrama UML inicial para realizar el juego de manera básica.

Por ese motivo, este documento corresponde a la planificación de la primera parte: **Creación básica del juego**, y una vez que funcione correctamente pasaremos a la planificación de la segunda parte.

Diagrama UML

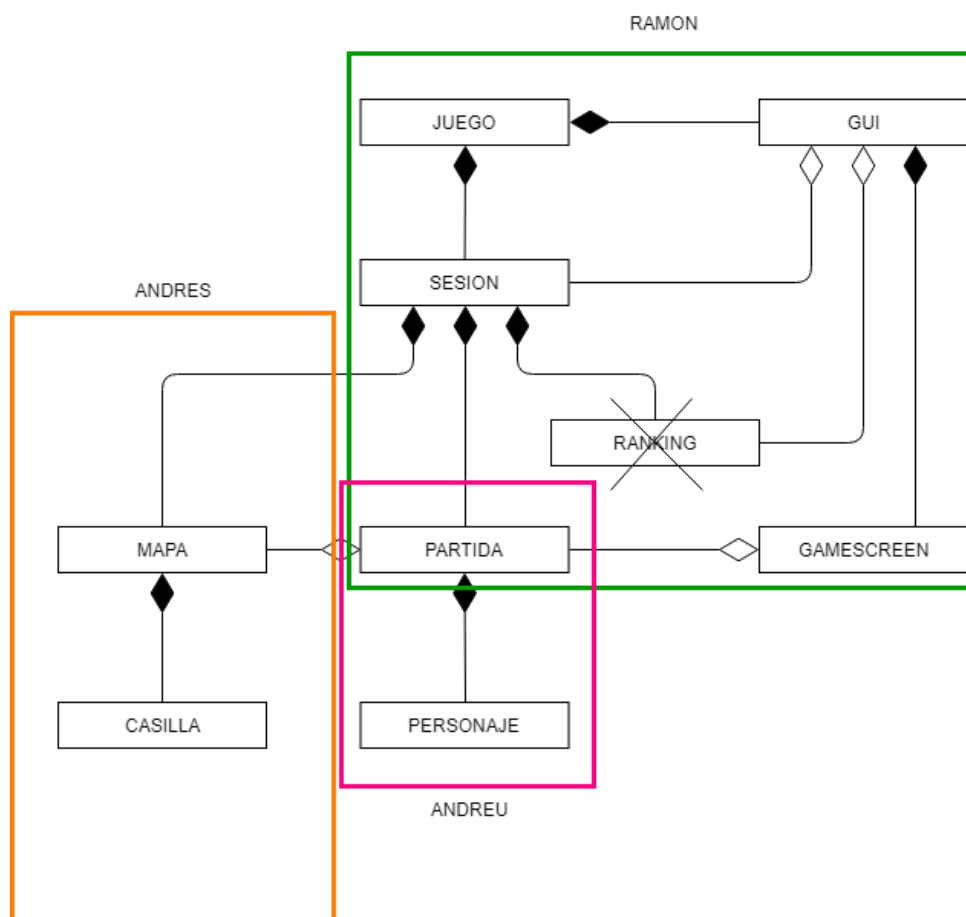
Para realizar nuestro proyecto de un juego en el que los jugadores deberán recorrer un mapa a través de una serie de obstáculos para llegar a la meta en el menor tiempo posible hemos planteado el siguiente diagrama UML.

Recordemos que este diagrama contiene lo básico para que el juego funcione para un jugador y, posteriormente, incorporaremos las clases necesarias para que funcione en modo multijugador.



Reparto de tareas

Basándonos en el UML anterior, nos hemos dividido el trabajo en función de trabajar en clases que estén relacionadas y que tengan un tiempo de trabajo similar. En cualquier caso, siempre estaremos dispuestos a ayudarnos si algún miembro del equipo necesita ayuda.



Andrés → Realizará la generación de los mapas con sus respectivas casillas, definiendo sus propiedades entre otras cosas.

Ramón → Encargado de la interfaz gráfica de la gestión del juego, la sesión e interfaz gráfica donde se desarrollará el juego.

Andreu → Llevará a cabo la lógica de la partida así como la del personaje, interacciones, movimientos, etc.

Explicación de las clases del cliente

Juego

Esta clase se encarga de iniciar el programa y crear la interfaz gráfica del juego.

Sesión

La clase sesión se crea al introducir un nombre de usuario y contraseña registrados en la base de datos (aunque esto no se implementará hasta la segunda fase del proyecto). Esta clase gestionará las partidas y puntuaciones que obtenga el jugador en dicha sesión.

Mapa

La sesión contiene una lista de mapas que el jugador puede elegir cuál jugar de manera individual. Cada mapa está compuesto por una serie de casillas que unidas en conjunto forman todo el mapa.

Casilla

Al basarnos en "tile-game", nuestro mapa estará formado por diferentes casillas que contendrán los atributos de dicha casilla, como por ejemplo, si es una casilla que hace daño o es una de inicio o fin del nivel.

Partida

La partida se crea una vez que el jugador a elegido un mapa en el que jugar. Es la clase más compleja ya que es la que se encarga de coordinar el mapa con los movimientos que hace el personaje para luego mostrarse en la pantalla de juego. De esta manera, cada vez que el personaje haga un movimiento la partida determinará hacia donde tiene que ir y cómo interactúa el personaje al entrar en contacto con determinadas casillas del juego, como por ejemplo las casillas de daño.

Personaje

Es la clase que se encarga de gestionar los movimientos que hace el jugador y como debe responder el personaje a dichos movimientos.

Ranking

(Esta clase no se implementará en esta fase) Accede a la base de datos y muestra el ranking con las mejores puntuaciones en una tabla.

GUI

Esta clase es la que se encarga de coordinar todos los elementos gráficos del juego.

Cámara de Juego / Game Screen

Esta clase gestiona lo que ve el jugador durante el desarrollo de una partida.

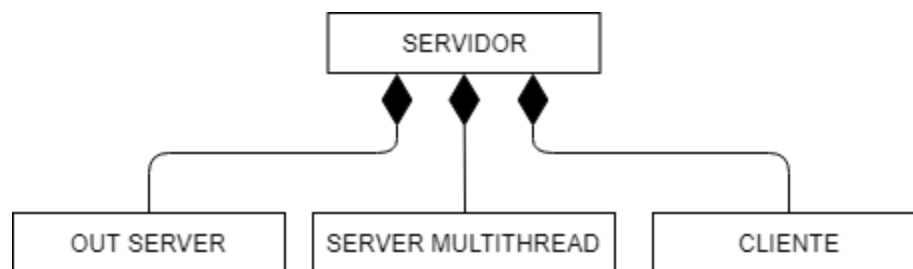
NOTA: Para la segunda fase del proyecto, tenemos pensado añadir clases como Servidor y Out Server, además de un diagrama UML del propio servidor, sin embargo, y como hemos dicho al principio de este documento, no nos centraremos en la segunda parte del proyecto hasta que la primera parte funcione correctamente.

Parte 2

Diagrama UML

Parte Servidor

Para la parte del servidor creemos que el diagrama UML que se corresponde a nuestra idea sería el siguiente:



Explicación de las clases del Servidor

Servidor

Esta clase inicia el servidor y sus componentes.

Out Server

Es la clase que envía peticiones al exterior

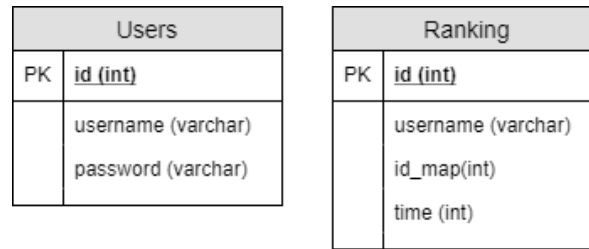
Server Multithread

Es el servidor que está escuchando y esperando que se conecten nuevos clientes.

Cliente

Es una lista de los clientes conectados al servidor, para poder gestionar la información que recibe de ellos y la información que les envía.

Diagrama Entidad-Relación Base de Datos



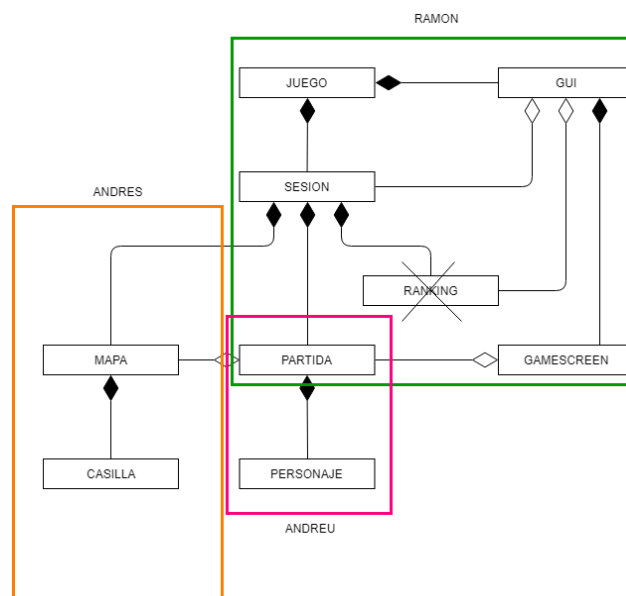
Diseño y arquitectura

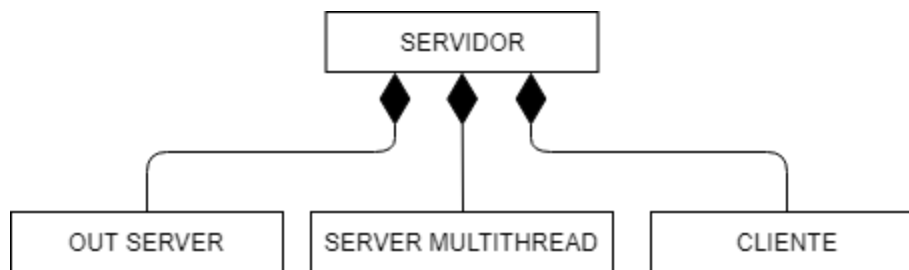
Andreu

Diagrama

Basándonos en el UML que realizamos en grupo, y el reparto de tareas. Mi parte es la relacionada con las clases PARTIDA y PERSONAJE (Rosa) y la parte del SERVIDOR. Relacionando la lógica del juego.

En la próxima entrega (U4) se explicará todo con más detalle, ahora solo es una explicación básica de la funcionalidad de la clase.





Explicación de las clases

Partida

Es una clase que será un Thread. La partida gestiona la lógica del juego y la de los personajes que hay en ella.

Se encarga de proveer los datos necesarios a los personajes para que puedan moverse por el mapa de una manera correcta, es decir, la lógica intermediaria entre los personajes y las casillas del mapa.

La partida está preparada para un máximo de 4 jugadores, donde se le asignan unos colores


Personaje

El personaje será un Thread también (cada personaje). Los personajes tendrán unas características propias como un color identificativo.

El Thread del personaje tiene la funcionalidad de para cada iteración moverse, siempre pidiendo la información necesaria a la partida, para cerciorarse de que el jugador puede hacer tal movimiento.

Server

En el proyecto retrorace hemos desarrollado un multijugador. Que puede ser jugador con la IP y el PUERTO, si el "RetroraceServer" está iniciado (El retroraceServer es un proyecto que simula un servidor en Java, este servidor se encarga de manejar a los jugadores online



y reenviarlos a los demás personajes unidos al servidor. Pero el RetroRaceServer será explicado más adelante.

Out Server

El Out Server es un Thread que lo que hace es enviar peticiones de conexión todo el rato a una IP y PUERTO (el servidor al que nos queremos conectar). Esto solo lo hace cuando no tenemos ningún servidor ya conectado en nuestro proyecto, por si se ha perdido la conexión o similar. Así se auto recupera la conexión.

ServerMT

Es la parte del servidor que se encarga de ir interpretando los mensajes que nos llegan por primera vez al servidor. El mensaje es interpretado. Si el código de llegada es válido, se acepta el mensaje y se crea un nuevo cliente en nuestro servidor. El cual es un hilo de ejecución, que sirve como medio de comunicación directo entre ese cliente en concreto y nuestro servidor.

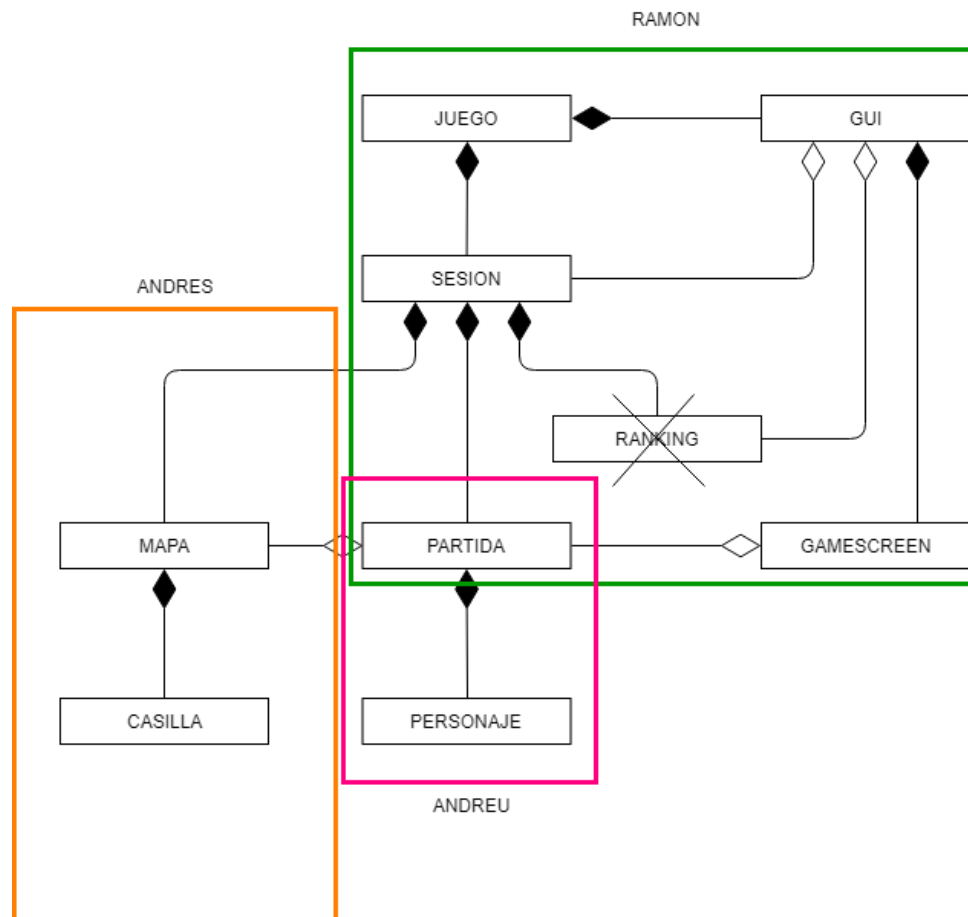
Cliente

Es un Thread, como hemos dicho anteriormente. Y gestiona las comunicaciones entre los clientes y nuestro servidor, para poder comunicarse de forma directa debidamente.

Ramón

Diagrama

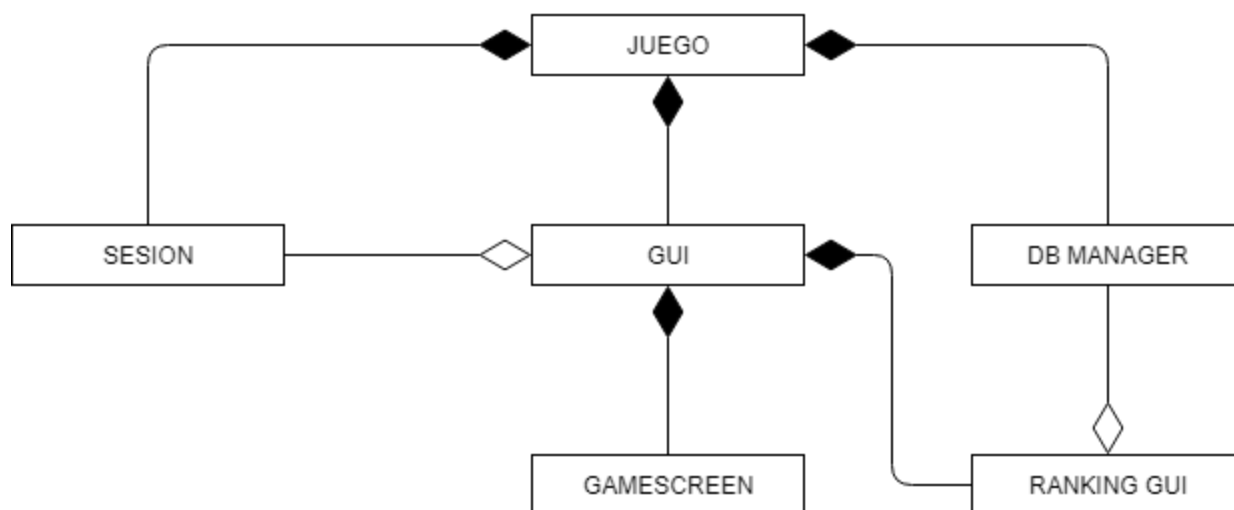
Basándonos en el UML que realizamos en grupo, y el reparto de tareas. Mi parte es la relacionada con las clases JUEGO, GUI, SESION y GAMESCREEN. Además de incorporar posteriormente RANKING en el caso de tener tiempo.



Diseño final de las clases

Tal y como se dijo en la planificación mi parte consistía en la interfaz gráfica de la gestión del juego, la sesión e interfaz gráfica donde se desarrollaría el juego.

Posteriormente y durante el desarrollo del proyecto, me vi en la necesidad de ampliar esas clases tal y como se muestra en el siguiente diagrama para poder incluir correctamente el Ranking que dudabamos sobre cómo implementar:



Explicación de las clases

Juego

Esta clase se encarga de iniciar el programa y crear la interfaz gráfica del juego.

Sesión

La clase sesión se crea al introducir un nombre de usuario y contraseña registrados en la base de datos. Esta clase gestionará las partidas.

GUI

Esta clase es la que se encarga de coordinar todos los principales elementos gráficos del juego, entre los que se incluyen los cambios de pantalla y todos los botones y campos de texto con los que puede interactuar el jugador.

Extiende de JFrame. Implementa ActionListener y KeyListener.

DB Manager

Esta clase ha sido creada para establecer una comunicación entre el juego y las entidades de bases de datos. De esta manera se pueden gestionar las acciones que hace nuestra aplicación con la base de datos.

Ranking GUI

Accede a la base de datos y muestra el ranking con las mejores puntuaciones.

Extiende de JPanel.

Gamescreen

Esta clase gestiona el repintado del canvas durante el desarrollo de una partida.

Extiende de Canvas. Implementa Runnable y KeyListener.

Diseño de Bases de Datos

Nuestro programa cuenta con un registro de usuarios y un ranking. Por ese motivo necesitábamos una base de datos para almacenar esta información.

Este es el diseño de nuestra base de datos:

Users	
PK	<u>id (int)</u>
	username (varchar)
	password (varchar)

Ranking	
PK	<u>id (int)</u>
	username (varchar)
	id_map(int)
	time (int)

Explicación estructura Base de Datos

Users

Esta tabla almacena el nombre de usuario y el password (encriptado) de los usuarios registrados.

username	password	id
admin	d033e22ae348aeb5660fc2140aec35850c4da997	1
test	a94a8fe5ccb19ba61c4c0873d391e987982fbbd3	3
andres	7110eda4d09e062aa5e4a390b0a572ac0d2c0220	4
Andreu	7c4a8d09ca3762af61e59520943dc26494f8941b	5

Ranking

Esta tabla almacena el nombre de usuario, el mapa y el tiempo que ha obtenido un jugador en una partida.

username	id_map	time	id
test	0	300	1
admin	0	200	2
admin	1	200	3
admin	0	68	4
Andreu	0	49	7

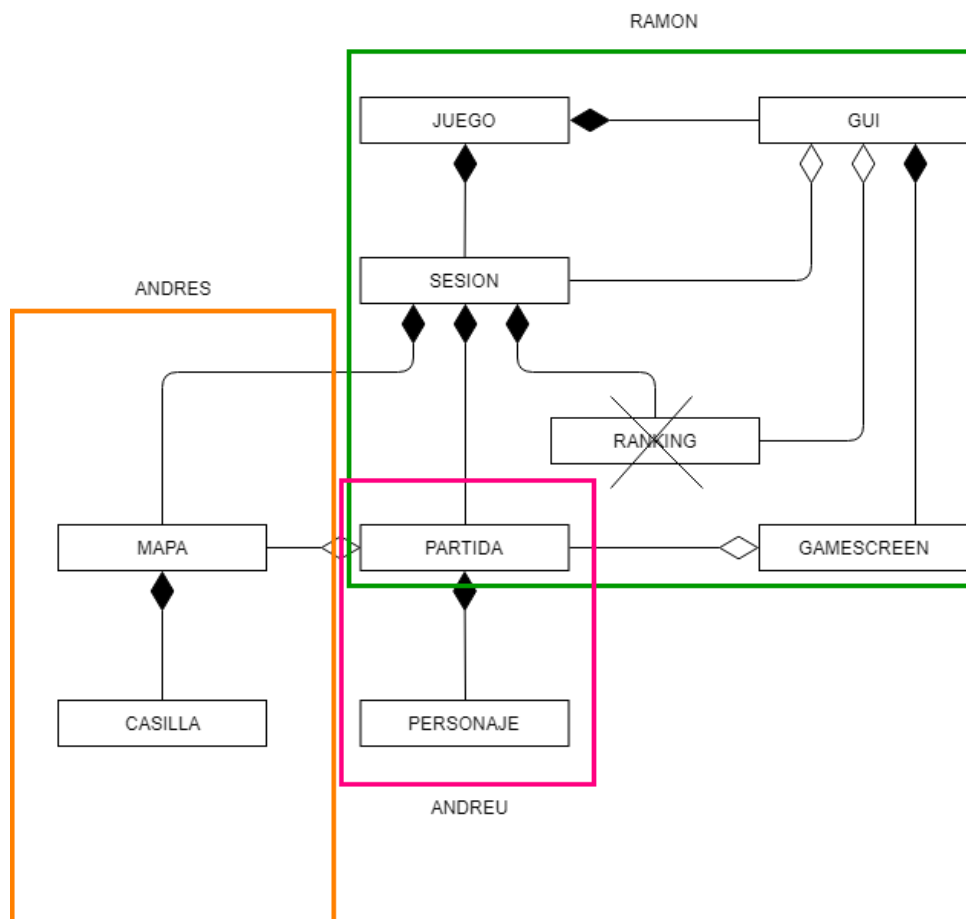
En este punto cabe destacar que no hemos utilizado claves foráneas para que al hacer la consulta a la tabla Ranking obtuvieramos directamente el nombre del usuario y evitar

tener que hacer un subquery para obtener el nombre de usuario a partir de una id. Además, el diseño de nuestro programa nos permitía realizar la base de datos de esta manera sin demasiados problemas.

Andrés

Diagrama

Basándonos en el UML que realizamos en grupo, y el reparto de tareas. Mi parte es la relacionada con las clases MAPA y CASILLA.



Explicación de las clases

Mapa

La sesión contiene una lista de mapas que el jugador puede elegir cuál jugar de manera individual, estos mapas serán cargados de un JSON. Cada mapa está compuesto por una serie de casillas que unidas en conjunto forman todo el mapa, estas casillas son cargadas de un JSON en el mapa.

Casilla

Al basarnos en “tile-game”, nuestro mapa estará formado por diferentes casillas que contendrán los atributos de dicha casilla, como por ejemplo, si es una casilla que hace daño o es una de inicio o fin del nivel.

Otros

También se realizará la búsqueda de imágenes, redimensionado de ellas, idear los mapas y las casillas, creación del JSON con cada una de las casillas existentes (con sus atributos, nombre e imagen), creación del JSON con cada uno de los mapas (con su nombre y distribución de cada una de las casillas).

Todos estos JSON serán cargados con la librería GSON permitiendo así tener un modelo de casilla y mapa, posteriormente con esta librería cargar todos los elementos y volcarlos en esos modelos.

Implementación y código

Andreu

Llistat amb breu descripció de cadascun dels arxius de codi font.

(PROYECTO RETRORACE)

- Partida (Thread)
 - La partida es la encargada de la lógica entre los personajes que están jugando en esa partida y el mapa. Es decir, el vínculo de como los personajes interaccionan con el mapa.
 - Tiene una ArrayList de Personajes, que en principio lo hemos dejado preparado para un máximo de 4 jugadores, cada uno de ellos con su color.
 - Al iniciarse la partida se iniciará el timer y todos los personajes que estén en ella, así como una música arcade.

- El Thread de la partida es un bucle infinito que solo mantiene la partida viva, y reproduciendo el loop la música.
- En caso de que sea Online, el bucle run() se encarga de hacer algo muy importante: **Enviar la información del jugador local al servidor (en caso de que la partida esté conectado al servidor)**
- Una vez finaliza la partida, se informa a la sesión de guardar el tiempo conseguido. Esto solo ocurre cuando se juega en modo Individual. Y se destruyen todos los thread asociados para optimizar recursos.
- Consta de setters y getters Básicos
- Y métodos con los que se los personajes se relacionarán para el movimiento con el mapa.
- Personaje (Thread)
 - El personaje es aquel objeto e hilo de ejecución el cual se encuentra en la partida y tiene que superar el mapa.
 - Consta de unos atributos (Mencionaré los básicos)
 - X e Y: Coordenadas
 - VelX y VelY: Velocidad de movimiento
 - Jumping, falling, muerto: Booleanos de estado
 - Fuerza de salto
 - Esta sobre suelo, esta moviendo: Boolean
 - Último checkpoint
 - Última dirección
 - Luego tiene otras propiedades adicionales como el sprite de imágenes que simulan su movilidad (gracias al Timer), el color, si es online, una ID, un sonido.. Que son recursos que mejoran la experiencia de cara al jugador.
 - El run() del personaje se basa en:
 - Si el jugador NO es online, moveremos al personaje. El moverPersonaje se comunica con la partida para aspectos básicos como:
 - ¿Puedo moverme a la derecha / izquierda?
 - ¿Que tengo a la derecha / izquierda / pies / centro de mi cuerpo?
 - Todo esto interacciona con el personaje y limita sus movimientos, así como se le aplica una gravedad si está cayendo, etc.

- Y sus métodos principales son:
 - Mover personaje: Lo mueve
 - Saltar / matar: Hace la acción correspondiente
 - Pintar: El personaje se repinta a él mismo con los gráficos que le da el Gamescreen
- Timer (Thread)
 - Esta clase fue creada más adelante al darme cuenta que para llevar a cabo las animaciones de mover los pies y las manos podía crearme mi propio Timer, que me proporcione lo que desee.
 - Es un contador, que puede ser inicializado, pausado, reseteado ,y acabado.
 - Se le puede pedir en qué momento del contador se encuentra.
 - Y se le puede decir cada cuando se desea incrementar el valor (en ms)
- Server
 - Gestiona la comunicación a nivel del protocolo de aplicación
 - Mensajes de salida:
 - "\$PLAYER-STATUS\$ + información del personaje
 - ID, ,X, Y, saltando, muerto, color, última dirección
 - Mensajes de entrada:
 - \$ID\$ → Es la ID que te devuelve el servidor cuando te conectas a él. Así cada jugador tiene su ID propia
 - \$COLOR\$ → Es el color que te devuelve el servidor cuando te conectas a él. Así cada jugador tiene su ID propia
 - \$ALL-PLAYERS\$ → Cuando una persona se conecta al servidor, se le devuelve un conjunto de IDs. Que son todas las IDs de los personajes que tiene el servidor. De tal manera que el juego local, añadirá a los jugadores si no existían antes en su partida.
 - \$PLAYER-STATUS\$ → Da la información necesaria a la partida local para que el jugador (especificado en la ID del mensaje) se pinte en el lugar correcto y de la forma correcta.

- ClientProject
 - El client project es el iniciador de la gestión del multiplayer. Inicia el out server en un determinado puerto e IP. Para que se intente conectar al servidor externo
- OutServer (Thread)
 - Intenta conectarse al servidor especificado anteriormente todo el rato. En caso de conseguir realizar tal conexión. Genera un server con su correspondiente hilo de ejecución. Y deja de conectarse.
 - Está preparado para recuperarse en caso de pérdida de conexión. Por lo que en caso de que se pierda la conexión, el servidor será eliminado y volverá a entrar en bucle de intentar conectarse.

(PROYECTO RETRORACESERVER)

- ServerProject
 - Es la parte encargada de gestionar el servidor, en el puerto especificado.
 - Tiene un array de clientes (players) y un array de colores el cual se los irá asociando
 - Una vez lo iniciemos, creará un serverMT, que explico justo a continuación
- ServerMT (Thread)
 - Es el encargado de ponerse en escucha del puerto especificado.
 - Al llegarle un mensaje se cerciora de que sea un mensaje válido de entrada de un cliente nuevo. Si lo es, crea un Thread de cliente y lo lanza para poder establecer una relación directa entre nuestro servidor y el cliente. Sin hacer esto, sólo podríamos comunicarnos con un único cliente.
 - Al detectar que es un cliente válido, le asociará un color a ese cliente.
- Client (Thread)
 - Es la clase que permite una comunicación directa entre el servidor y el cliente con el que hemos establecido la conexión.
 - Nada más se cree este thread y se inicie, se le reenviará al cliente una ID única y un color, para que la partida online pueda ser llevada a cabo con normalidad. Así como todos los personajes del juego actuales en el servidor.
 - A continuación se quedará recibiendo y procesando los mensajes de:
 - \$PLAYER-STATUS\$

- Estos mensajes son transmitidos al ServerProject para que haga un broadcast a los clientes y que todo el mundo reciba la información de los otros jugadores.
- ServerFrame
 - Ventana gráfica del servidor con su propio log

Hay muchos aspectos que me han costado y las he tenido que pensar y reflexionar varias veces. Pero hay que destacar 2

- Lógica de colisiones / objetos
 - El cómo el jugador se comunica con la partida via X e Y y la partida le informa y hace las acciones necesarias sobre el personaje según donde se encuentre en el mapa.
 - Ya que no es una x e y. Hay que comprobar lo que tiene en su cabeza, a su derecha, a su izquierda, en sus pies, en la casilla del suelo..... Etc.
 - Por lo que fué un momento largo y costoso
- Servidor
 - Bien la estrategia de montaje del servidor era clara. Había el problema de cómo cada personaje era independiente a los demás y cómo identificarlo en nuestra partida local sin necesitar de abrir constantes comunicaciones. Así que la opción fue que el servidor nos de una ID única al conectarnos a él, así como un color. De forma que la partida local recibe las id de los personajes online y sabe que hacer con cada uno localmente.

Aunque también ha habido otros problemas, estos son los principales

Ramon

NOTA: Durante la explicación de los archivos también intento explicar las partes más relevantes de cada parte del código. Para mayor profundidad, el código de las clases está debidamente comentado en el proyecto.

Archivos del código fuente

A continuación se muestran los archivos en los que he intervenido principalmente, así como las implementaciones más relevantes en estas clases.

Juego

Es la clase principal, a partir de la cual implementamos las otras. Concretamente se encarga de la creación de la interfaz gráfica y de crear la clase que conecta con la base de datos.

Sus métodos más destacables son aquellos que nos sirven para conectar con la base de datos para que otras clases que tengan acceso a juego puedan enviar sus datos para que se almacenen en la base de datos.

Sesión

Esta clase también es muy importante ya que gestiona la creación y finalización de partidas, así como cargar la lista de mapas (implementado gracias a la ayuda de Andrés Rodríguez).

Entre sus métodos más importantes, destacar los que crean las partidas (en local y en online) y el que las finaliza, así como el de cargar los mapas desde un json.

GUI

Esta clase es la que se encarga de coordinar todos los principales elementos gráficos del juego, entre los que se incluyen los cambios de pantalla y todos los botones y campos de texto con los que puede interactuar el jugador.

Extiende de JFrame. Implementa ActionListener y KeyListener.


Es una clase muy importante ya que en esta clase es la que tiene toda la interfaz de usuario. Por ese motivo tiene muchísimos métodos relevantes entre los que destaco:

- Iniciar la GUI y añadir todos sus componentes.
- Creación e inicialización de todos los paneles entre los que puede ir cambiando el usuario al viajar a través de los menús del juego
- ActionListener y KeyListener de todos los paneles anteriormente mencionados.
- Métodos para llamar a la clase Juego y poder acceder a la base de datos para posteriormente poder mostrar los datos con la información recibida y que el usuario lo pueda ver.
- Del mismo modo, y para prevenir conflictos, cuando el usuario accede a alguna base de datos, los botones del panel en el que esté se bloquean y muestra un loader. De este modo se previene que el usuario cause algún problema mientras se accede a la base de datos, a la vez que mostrando el loader el usuario es consciente de que el programa está trabajando por detrás y no se ha bloqueado.

Usuario

Password

Entrar Registrarse



- Control de errores de algunos elementos, para que el usuario sepa en todo momento si algo falla y por qué falla.

BIENVENIDO

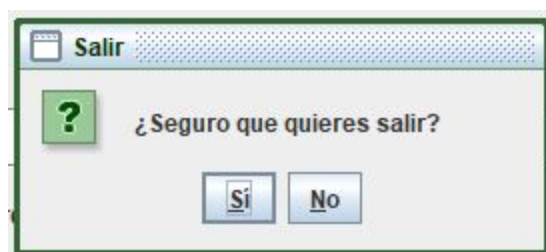
Usuario

Password

Entrar Registrarse

ERROR: Rellena todos los campos

- Métodos para llamar a las respectivas clases si se necesita iniciar una partida, ver el ranking, etc.
 - Mensaje de confirmación para salir del juego



DB Manager

Esta clase ha sido creada para establecer una comunicación entre el juego y las entidades de bases de datos. De esta manera se pueden gestionar las acciones que hace nuestra aplicación con la base de datos.

Entre sus métodos más importantes aquellos para comprobar si un usuario existe (y así evitar registrar otro con el mismo nombre), comprobar las credenciales de un usuario cuando hace login, guardar records en el ranking y obtener los récords del ranking.

Ranking GUI

Accede a la base de datos y muestra el ranking con las mejores puntuaciones. Es un panel que en función de los botones que elija el usuario nos mostrará una información u otra.

Entre sus métodos más destacables, aquellos que acceden al DB Manager para recibir listas de objetos Ranking. También destacar el método para colocar los datos recibidos en los JLabel, así como un método que pasa el tiempo recibido desde base de datos (en segundos) a minutos y segundos.

Gamescreen

Está clase gestiona el repintado del canvas durante el desarrollo de una partida. Para realizar esta clase recibí la ayuda de mis compañeros (Andreu Juan y Andrés Rodríguez), ya que esta clase estaba fuertemente vinculadas con las suyas para poder pintar correctamente el mapa y los personajes en movimientos.

Los métodos más importantes son aquellos que se encargan de repintar la pantalla para que el juego se vea fluido, así como los KeyListener para saber cuándo se debe mover un personaje.

Script de creación de Bases de Datos

Para almacenar datos en nuestra base de datos hemos usado la plataforma de Google Cloud. La hemos usado para que cualquier usuario desde cualquier ordenador pueda acceder a toda la información, de modo que no se quede en una simple base de datos local. De esta manera, los usuarios y los rankings quedan almacenados en la nube.



Después de crear las instancias correspondientes en la plataforma, procedimos a la creación de la base de datos a través del siguiente script.

```
CREATE DATABASE my_retrorace;

USE my_retrorace;

CREATE TABLE users (username VARCHAR(50), password VARCHAR(100), id
INT NOT NULL AUTO_INCREMENT, PRIMARY KEY(id));

CREATE TABLE ranking (username VARCHAR(50), id_map INT, time INT, id
INT NOT NULL AUTO_INCREMENT, PRIMARY KEY(id));
```

Base de Datos en Java: Entidades

Para poder almacenar y obtener datos desde nuestra base de datos hemos utilizado objetos persistentes en bases de datos y entidades, de manera que nuestras tablas de bases de datos se convierten en objetos y podemos acceder a ellas mediante controladores y nuestra clase DBManager.

En los controladores además incluimos nuestros métodos personalizados con las queries de SQL específicas que queríamos así como el número máximo de resultados (por ejemplo para obtener los datos de los rankings).

Otros archivos

Clase: Hash

Esta clase incluida en el paquete crypt fue creada con la única finalidad de encriptar las contraseñas. De esta manera siempre trabajamos con las contraseñas encriptadas de manera que los datos de nuestros usuarios están siempre seguros.

Archivo: [img/gui/loader.gif](#)

Este archivo fue incluido porque cuando conectamos con la base de datos la pantalla se quedaba congelada y el usuario no podía saber si el juego había dejado de funcionar o si realmente estaba funcionando.

De este modo, cuando se interactúa con la base de datos se muestra este gif, para que el usuario comprenda que el juego sigue funcionando y está procesando la información.

Problemas surgidos durante la implementación.

La fase de implementación no ha sido sencilla y cuando se solucionaba un problema normalmente aparecía otro y en muchas ocasiones debíamos comunicarnos con nuestros compañeros para poder solucionarlo. Sin embargo, me gustaría destacar los siguientes problemas:

- **Acceso a la base de datos con la pantallas congeladas:** Como he mencionado en varias ocasiones, al acceder a la base de datos deja el programa bloqueado. Esto se solucionó con la creación de un Thread: mientras el hilo principal mostraba un gif de carga, el Thread se encargaba de acceder a la base de datos.
- **Acceder a Google Cloud SQL desde cualquier PC:** La base de datos en la nube de Google es muy útil, pero su configuración dio bastantes quebraderos de cabeza. Especialmente porque para conectarme a su base de datos no bastaba con el nombre de usuario y contraseña de un usuario autorizado en la base de datos, sino porque necesitaba incluir la IP de cada ordenador que se quisiese conectar para que Google no la bloquease. Después de buscar librerías y opciones de Google la solución fue más sencilla de lo esperado: en la lista de IPs permitidas incluir 0.0.0.0/0, de manera que todas las IPs pudiesen acceder.
- **Problemas con el Gamescreen y una actualización de Windows:** La parte visual ha dado muchísimos problemas (parpadeos, pantallas en negro...). Pero uno muy curioso fue cuando en una parte avanzada del proyecto, la imagen del juego aparecía en negro en algunos ordenadores y en otros no. Tras varias horas revisando código descubrimos que la última actualización de Windows (por algún motivo desconocido) nos hacía aparecer la pantalla en negro, por lo que tuvimos que volver a una versión anterior de Windows.

- **Otros problemas menores:** Null Pointer Exception ha sido uno de los errores con los que más he tenido que tratar por el tema de ver cuando se creaban y se accedían a algunos elementos.

Andrés

Listado y descripción de archivos

- data\casillas.json: Contiene cada una de las casillas disponibles en el juego, con sus particularidades, ruta de la imagen e id.
- data\mapas.json: Contiene cada uno de los mapas que contiene el juego, con un título, una ruta de la imagen para poder ser previsualizado en el menu de eleccion de mapas y por último la distribución de casillas.
- img*: Contiene cada una de las imágenes necesarias para la visualización de las casillas, jugador y previsualización de mapas.
- src\retrorace\Casilla.java: Modelo de casilla para poder ser cargadas del JSON también contiene los métodos necesarios como por ejemplo el cargar la imagen del directorio o devolver la propiedad de esta casillas.
- src\retrorace\Mapa.java: Modelo de Mapa para poder ser cargadas del JSON también contiene los métodos necesarios como por ejemplo inicializar el mapa, devolver casilla de una coordenada proporcionada, también repinta cada casilla.
- src\retrorace\Gamescreen.java: En esta clase nos ayudamos entre todos en mi caso ayude en la parte del doble buffer para evitar los parpadeos de la pantalla, esta clase se encarga de visualizar el juego llamando a cada elemento para que se repinte hasta que la partida acaba.
- src\retrorace\Sesion.java: En esta clase ayude en el método encargado de cargar los mapas existentes.

Detalles de implementación

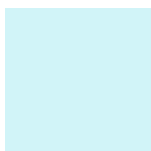
En mi caso las partes más significativas son:

Casillas

Propiedades

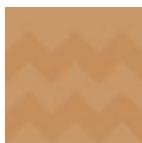
- **Transitable:** Bloque de fondo el cual no interactúa con el personaje.
- **Intransitable:** Bloque el cual el personaje no puede atravesar.
- **FinalizableOn:** Bloque en el cual finaliza la partida el personaje activado.
- **FinalizableOff:** Bloque en el cual finaliza la partida el personaje desactivado.
- **Sostenedor:** Bloque en el cual puede sostener el personaje.
- **Eliminatorio:** Bloque el cual daña al personaje.
- **CheckpointOn:** Bloque en el cual el personaje puede volver en caso de ser dañado activado.
- **CheckpointOff:** Bloque en el cual el personaje puede volver en caso de ser dañado desactivado.
- **Agua:** Bloque en el cual la velocidad de movimiento del personaje es afectada.
- **Trampolin:** Bloque en el cual el salto es modificado mejorando su altura.

Cielo



- Id: 1
- Transitable.
- Bloque de fondo para relleno.

Tierra



- Id: 2
- Intransitable.
- Bloque el cual no puede atravesar el personaje.

Hierba



- Id: 3
- Sostenedor.
- Bloque el cual puede sostener el personaje y en el cual puede caminar.

Antorcha



- Id: 4,9,12
- CheckpointOff y CheckpointOn.
- Bloque el cual al ser atravesado en caso de ser dañado el personaje volverá a este punto, tiene dos estados activado y desactivado el activado tiene una animación la cual va cambiando de posición el fuego.

Fin



- Id: 5,10,11
- FinalizableOff y FinalizableOn.

- Bloque el cual al ser atravesado el personaje, finaliza la partida, tiene dos estados activado y desactivado el activado tiene una animación la cual va cambiando de posición la bandera.

Pinchos



- Id: 6
- Eliminatorio.
- Bloque el cual al ser atravesado el personaje, elimina el personaje.

Nube



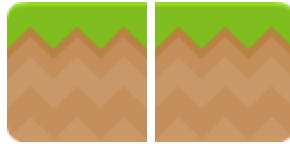
- Id: 7,8
- Transitable.
- Bloque para dar más detalle al fondo está dividido en dos.

Pasarela



- Id: 13
- Sostenedor.
- Bloque el cual puede sostener el personaje y en el cual puede caminar.

Plataforma



- Id: 14,15
- Sostenedor.
- Conjunto de bloques formado por Plataforma_Izquierda, y Plataforma_Derecha para formar una plataforma elevada en la cual puede transitar el personaje.

Agua



- Id: 16
- Agua.
- Bloque en el cual la velocidad del personaje es reducida.

Trampolin



- Id: 17
- Trampolin.
- Bloque en el cual la altura del salto es más elevada.

Seta



- Id: 18
- Transitable.
- Bloque decorativo con una seta.

Planta



- Id: 19
- Transitable.
- Bloque decorativo con una planta.

Cactus



- Id: 20
- Transitable.
- Bloque decorativo con una Cactus.

Roca



- Id: 21
- Sostenedor.
- Bloque decorativo con una roca la cual puede ser pisada.

Mapas

Instalación y configuración

Instalación del entorno de desarrollo

Todas las cosas necesarias ya están directamente linkeadas, para que no tenga que configurarlas a mano.

Si desea descargar el proyecto y abrirlo directamente, puede clonarlo desde nuestro GitHub:

<https://github.com/SOSandreu1095/retrorace>

Instrucciones para la compilación del proyecto

El proyecto está preparado para que compile sin necesidad de realizar cambios.

Si tiene problemas, asegúrese que estén importadas las librerías correctamente. (Ver la carpeta lib), y también asegurarse de las carpetas: data, music, img.

Instalación de la Base de Datos

Un aspecto positivo de nuestro proyecto es que una vez creada la Base de Datos en Google Cloud y configurarla correctamente con nuestro proyecto, el usuario no debe instalarse ninguna base de datos ni configurar nada, ya que el programa accede directamente a la base de datos desde cualquier IP y ordenador.

Instrucciones de configuración

En nuestros ordenadores, el programa funcionaba bajo las siguientes circunstancias: Java 1.8 y en Windows sin la última actualización (que nos presentaba algunos problemas). Sin embargo creemos que se ha fixeado el error que había, ya que un compañero nuestro lo ha probado y no ha habido problemas.

Lo de la última actualización lo decimos porque, al actualizar Windows el canvas no se veía correctamente, fuimos a versiones más antiguas del proyecto, y tampoco iba bien. A uno de nosotros se le ocurrió restaurar una versión de Windows más antigua y funcionó correctamente. Por lo que si está probando el programa en Windows y falla, le recomendamos que intente volver a una versión anterior.

Además, es importante que en el mismo directorio donde esté el .jar, se encuentren las siguientes carpetas:

- data → Contiene los mapas y las casillas
- music → Contiene los sonidos del juego
- img → Contiene las imágenes
- lib → Librerías como GSON y otras para la base de datos.

Manual de usuario

Objetivo del juego

El objetivo del juego consiste en llegar a la meta en el menor tiempo posible superando los diferentes obstáculos que hay en el mapa.

A lo largo del mapa podrás interactuar con una serie de objetos, como unas antorchas que te proporcionarán un punto de retorno en caso de ser eliminado.

Objetos

A lo largo del mapa, hay cinco objetos que afectan al usuario.

Antorcha

Al pasar sobre ella nos proporciona un punto de retorno si somos eliminados. El jugador volverá a la última antorcha que hubiese encendido.



Pinchos

Al entrar en contacto con ellos seremos eliminados y volveremos a la última antorcha encendida.



Trampolín

Al saltar sobre un trampolín el salto del personaje será mucho mayor.



Agua

Al entrar en contacto con el agua la velocidad del personaje se reducirá.



Meta

Al llegar a la meta la partida finaliza.



El resto de objetos son de propia colisión o decorativos.

Mecánicas

En todos los modos de juego deberemos llegar a la meta en el menor tiempo posible teniendo en cuenta los [objetos](#), ya que estos nos afectan.

Un jugador

En esta modalidad jugaremos solos, una vez lleguemos a la meta nuestra marca será guardada en el Ranking.

Dos Jugadores

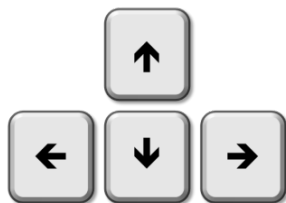
En esta modalidad dos personas jugarán la misma partida sin necesitar otro ordenador.

Multijugador en LAN

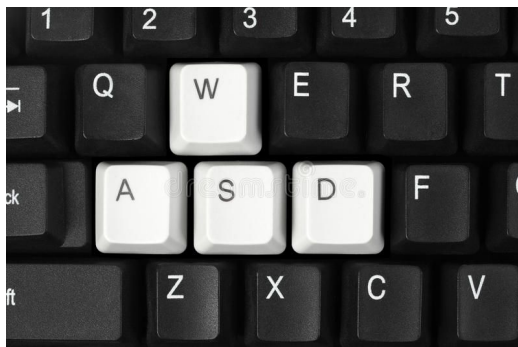
Para jugar esta modalidad será necesario estar conectados a una red ya sea con internet o sin, y disponer un servidor encendido al que nos conectaremos a la vez que otros jugadores y competiremos contra ellos.

Controles

En todos los caso se urán la flechas del teclado para dirigirnos derecha, izquierda y salto, a la vez también se ha añadido la opción de saltar a la tecla de espaciadora. En caso de jugar a la modalidad Dos jugadores uno de ellos deberá usar estas teclas.



Las siguientes teclas solo estarán disponibles en la modalidad Dos jugadores las cuales serán para que el segundo jugador pueda controlar su Personaje.



Ranking

El ranking muestra los mejores tiempos conseguidos en los diferentes mapas.

Personal

Muestra los mejores tiempos del usuario.

Global

Muestra los mejores tiempos de todos los usuarios registrados.

Ciente

Inicialización

Para iniciar el juego nos dirigiremos al directorio donde esté ubicado el juego y ejecutaremos retrorace.jar:

data	28/05/2018 18:46	Carpeta de archivos	
img	28/05/2018 18:47	Carpeta de archivos	
lib	28/05/2018 18:46	Carpeta de archivos	
music	28/05/2018 18:46	Carpeta de archivos	
retropace.jar	28/05/2018 18:45	Executable Jar File	115 KB
retroaceserver.jar	28/05/2018 18:45	Executable Jar File	19 KB

Hacer login

Una vez ejecutado nos aparecerá un Login en el que deberemos introducir nuestras credenciales o crearnos unas nuevas.

Si tenemos credenciales, las introducimos en los campos de texto y posteriormente pulsamos "Entrar" o la tecla Enter.

Al hacer Login nos aparecerá el Menú del juego.



BIENVENIDO

Usuario

Password

Registrar un usuario

En el caso de no tener un usuario para hacer login, podemos crear uno pulsando el botón "Registrarse".

Una vez en la ventana de registro, deberemos introducir un usuario no existente (si el usuario existe seremos notificados para su corrección) y un password que deberemos repetir (en caso de que no sean iguales seremos notificados).



REGISTRO

Usuario

Retrorace

Password

••••

Repetir Password

••••

Registrarse Cancelar

ERROR: El nombre de usuario ya está en uso

Una vez creado automáticamente volveremos a la ventana de Login y seremos notificados que el usuario ha sido creado correctamente.



BIENVENIDO

Usuario

Password

Entrar Registrarse

Usuario registrado correctamente.

Menú del juego

Cuando hayamos hecho Login nos encontraremos en el menú principal en el que un mensaje nos dará la bienvenida mostrando nuestro nombre de usuario con el que nos hemos conectado. A partir de aquí podemos pulsar los siguientes botones:



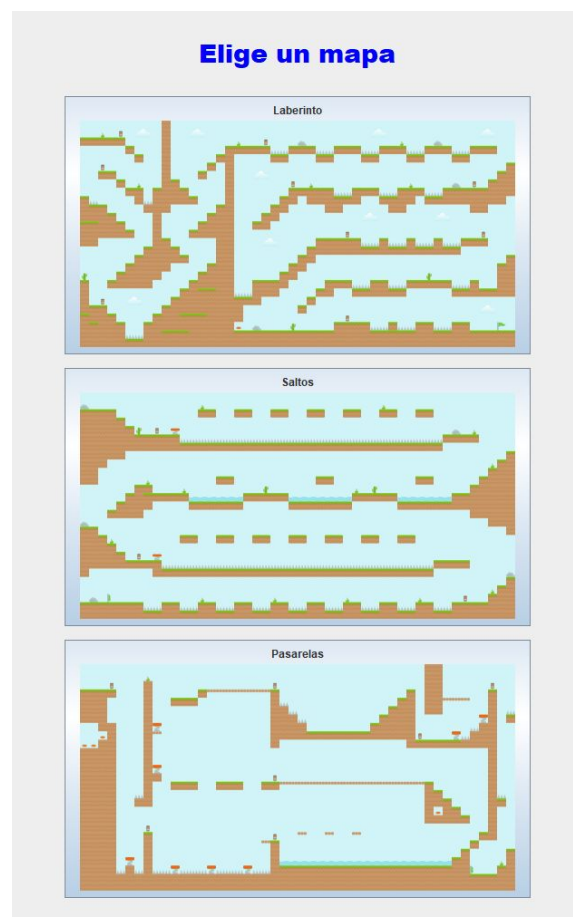
Un jugador

Una vez elegida esta modalidad nos aparecerá el selector de mapas, al elegir uno de ellos el juego comenzará.

En esta modalidad será la única donde se guardaran las puntuaciones.

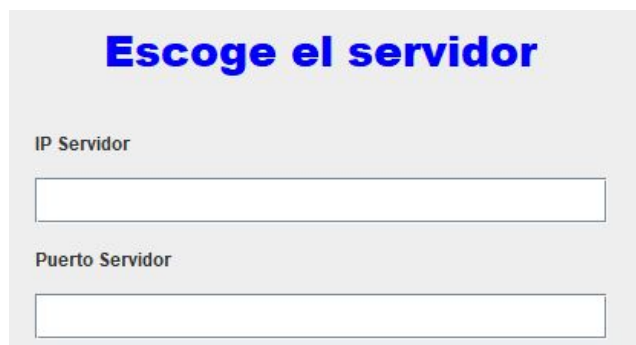
Dos jugadores

En esta modalidad igual que en la modalidad Un jugador, al ser elegida, nos aparecerá el selector de mapas.



Multijugador en LAN

En esta modalidad a diferencia del resto al ser elegida nos aparecerá una ventana en la que deberemos introducir la IP del



servidor y su puerto para conectarnos a él.

Una vez introducidos los datos y pulsado el botón “Conectarse” la partida empezará.

Ranking

En él nos aparecerán nuestras mejores marcas de cada uno de los mapas, también nos aparecerán a nivel global y quien ha realizado dicha marca.

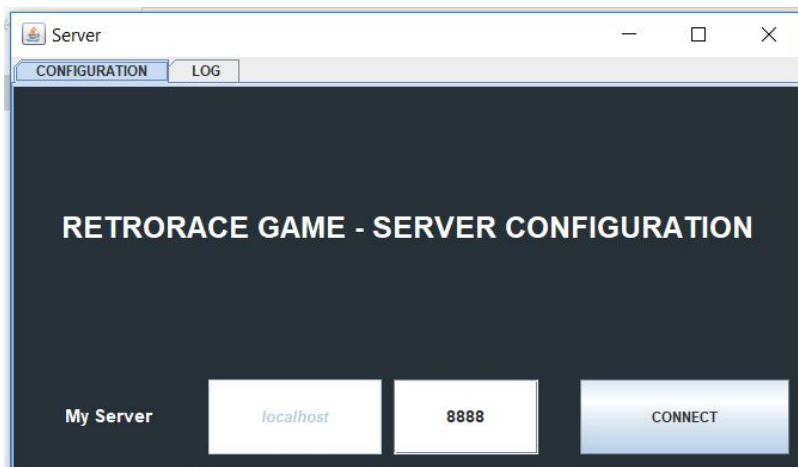
MI RANKING		RANKING MUNDIAL	
Laberinto		Saltos	
Pasarelas			
Usuario	Tiempo		
andres	1:58		
-	-		
-	-		
-	-		
-	-		
-	-		
-	-		
-	-		
-	-		
-	-		

Servidor

Para configurar el servidor, abra el archivo “retroraceserver.jar”

data	29/05/2018 18:32	Carpeta de archivos	
img	29/05/2018 18:32	Carpeta de archivos	
lib	29/05/2018 18:32	Carpeta de archivos	
music	29/05/2018 18:32	Carpeta de archivos	
retrorace.jar	29/05/2018 18:31	Executable Jar File	115 KB
retroraceserver.jar	29/05/2018 16:47	Executable Jar File	19 KB

Y se le abrirá el servidor para que pueda configurarlo:



Lo que va a tener que hacer únicamente es elegir el puerto que desee (en este caso 8888). Y darle al botón de conectar.

En el menú log del programa podrá ver información al respecto de lo que ocurre en el servidor:

