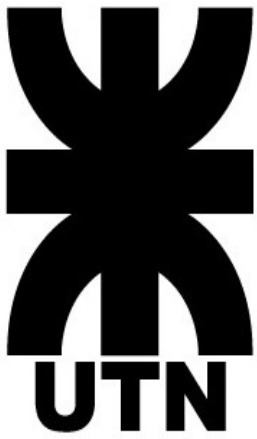
UNIVERSIDAD TECNOLÓGICA NACIONAL

FACULTAD REGIONAL CÓRDOBA



TRABAJO PRÁCTICO INTEGRADOR

MMORPG – Martín Fierro Treasure Hunter

Carrera: Ing. en Sistemas de Información

Cátedra: Arquitectura de Software

Profesores: Ferreyra, Alexis

Frías, Pablo

Integrantes:

Barrionuevo Battistini, Diego Santos Legajo: 58065

- 2015 -

Martín Fierro Treasure Hunter (MMORPG)

El software solicitado consiste en un juego online tipo MMO-RPG. Los requerimientos iniciales son los siguientes:

- Mapas de 20 habitaciones interconectadas.
- Habitaciones de un tamaño aleatorio (entre 20 y 50 tanto el largo como el alto).
- Un máximo de 5 jugadores al mismo tiempo.
- Los jugadores deberán encontrar un único tesoro escondido aleatoriamente, y el primero en encontrarlo gana.
- Existen monstruos que aparecen en forma al azar y no se mueven.
- Si un jugador “toca” un monstruo, el jugador pierde.
- Cada habitación posee los siguientes tipos de bloque: pared, puerta.
- Las puertas se encuentran ubicadas en el perímetro de las habitaciones, conectándolas.
- Las paredes bloquean el paso tanto a jugadores como a monstruos.

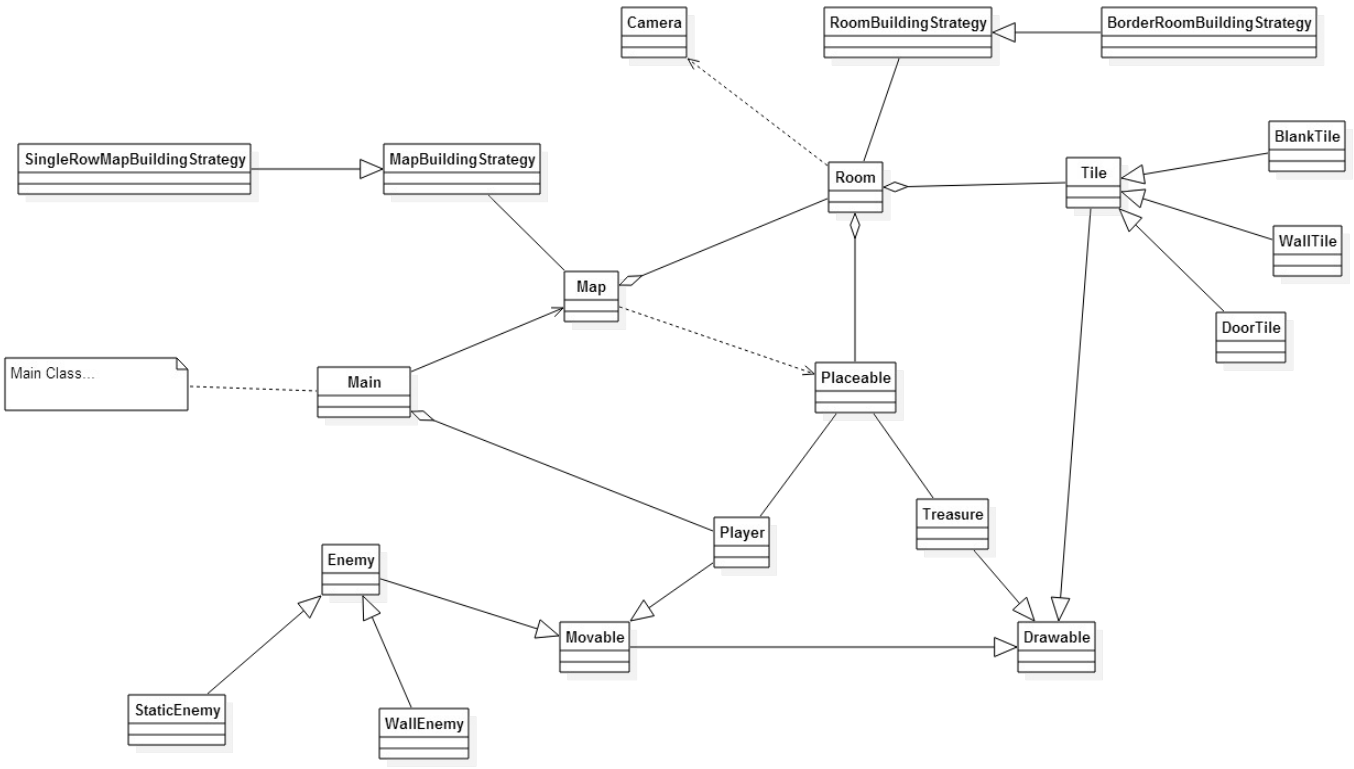
Decisiones de Diseño

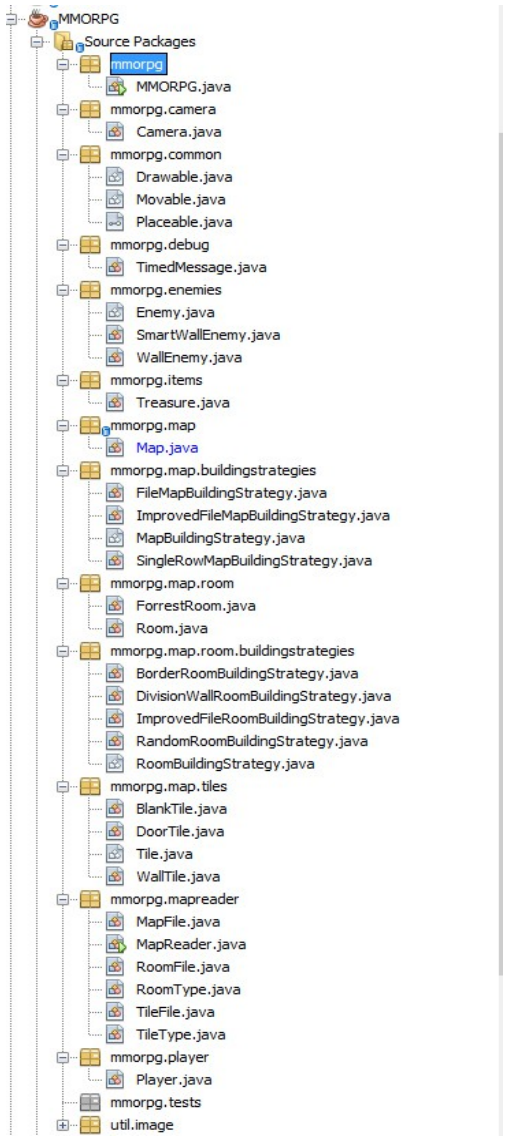
Para llevar a cabo el juego, he propuesto crear una arquitectura cliente-servidor, la función general de cada uno se detalla a continuación:

- Cliente: encargado de descargar el contenido del servidor (como ser imágenes y audio) y reproducirlo en el momento en el que servidor lo indique: por ejemplo, renderizar un mapa con tal recurso de imagen para los bloques, reproducir tal archivo de sonido cuando el jugador choca con un monstruo, etc. El cliente únicamente envía mensajes al servidor y es este último el que decide cómo se continuará el juego en función de estos mensajes.
- Servidor: será del tipo *servidor autoritativo*, es decir que será responsable de cualquier aspecto relacionado con la lógica del juego: decide cómo serán los mapas, dónde estará el tesoro y los monstruos, dónde iniciará el juego cada jugador al conectarse, cuándo termina el juego, etc.

Lógica del juego

El siguiente diagrama de clases muestra cómo he decido plantear la arquitectura que envuelve la lógica del juego, al mismo tiempo que describo brevemente las clases más importantes:





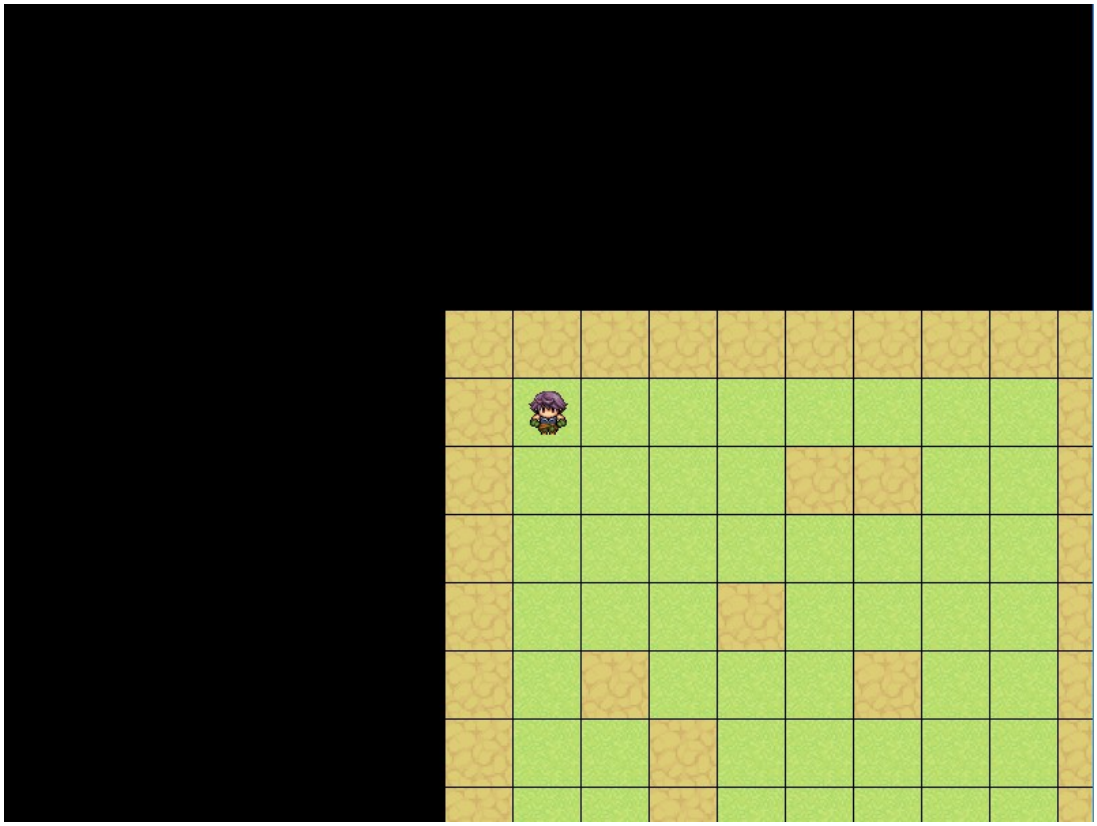
- Mapa: el mapa posee un conjunto de habitaciones y una estrategia de construcción (patrón strategy). Con ésta el mapa se construye, indicando qué habitaciones están interconectadas, por ejemplo, una estrategia simple sería crear un mapa con una sala a continuación de la anterior, otra estrategia sería construirse a partir de un archivo de texto, etc. El mapa es el que, en el cliente, renderizaría y actualizaría la sala en la que el jugador está actualmente. Por defecto, la forma de construcción del mapa y sus salas es por medio de un archivo JSON que cada servidor podría modificar para *customisar*(personalizar) su juego.
- Habitación (o sala): posee un conjunto de bloques, y una lista de objetos de visualización, a los cuales renderiza y actualiza (en el cliente). Estos objetos pueden ser jugadores, enemigos, el tesoro, etc (todos implementan la misma interfaz). Además, posee una estrategia de construcción (algo similar al mapa), la cual decide cómo estarán dispuestos los bloques en la sala.
- Bloque (*tile* en inglés): existe una jerarquía que le permite a cada bloque especificar su recurso de imagen a mostrar, si es o no un bloque *caminable*, si es una puerta, etc. Existen diferentes tipos de bloques: paredes, puertas y vacíos. Cada sala puede tener más de un

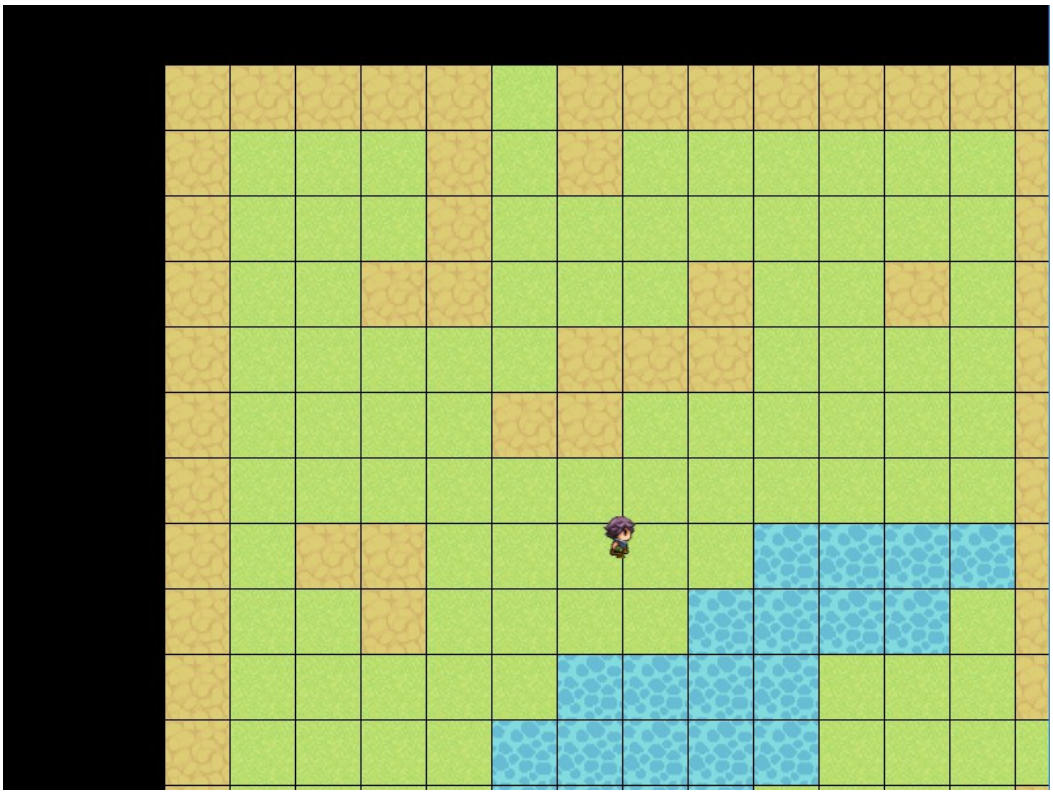
bloque del mismo tipo, por ejemplo, en un bosque puedo tener un bloque tipo pared tanto para las piedras como para los árboles.

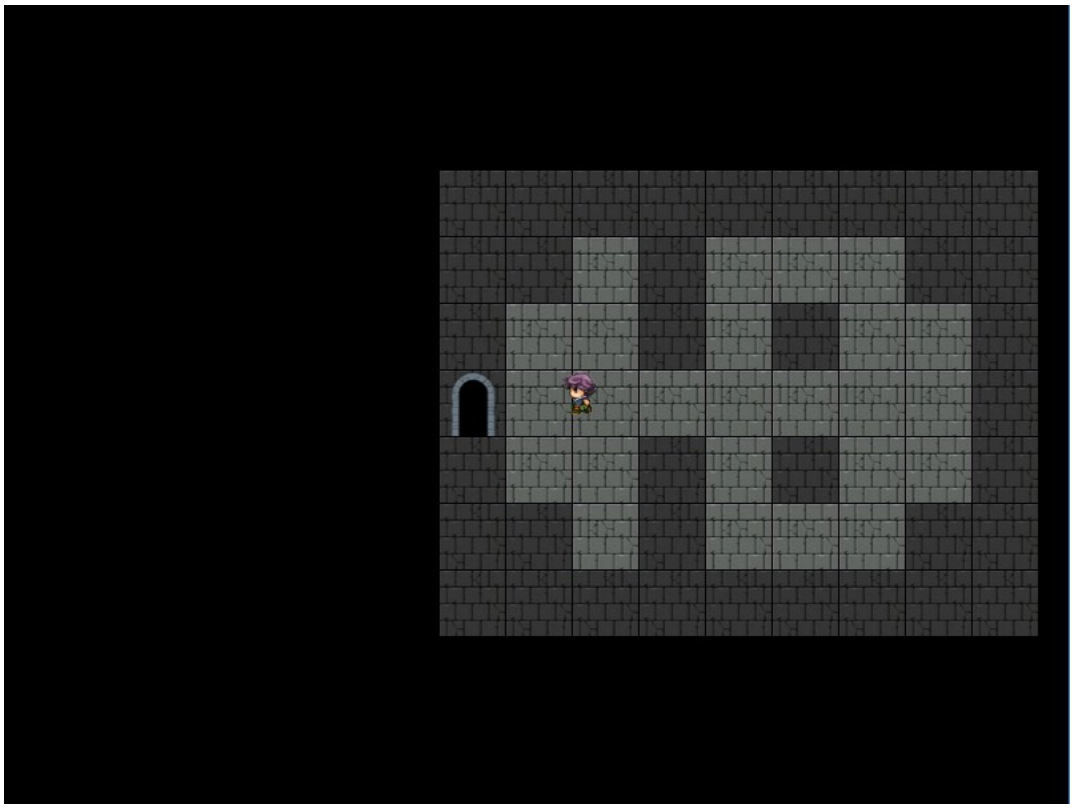
- Enemigos (o monstruos): existe un jerarquía pensada en la flexibilidad, así, todos los enemigos comparten el comportamiento predeterminado, como ser poder ser chocados por un jugador, no atravesar bloques de tipo pared, etc. A su vez, pueden existir distintos tipos de enemigos, algunos pueden ser estáticos, otros pueden moverse en dirección aleatoria, otros pueden seguir a algún jugador cercano a ellos, etc.
- Jugador: es controlado por un cliente, se visualiza en una sala determinada e interactúa con el resto de los objetos visualizables en la misma.
- Jerarquía de objetos visualizables: todos los objetos dentro de una sala, como ser: jugadores, monstruos, tesoro, etc.
- Jerarquía de objetos renderizables/movibles: todo objeto que pueda ser renderizado en el juego debe extender *Drawable*, y todo objeto que pueda moverse (ya sea por un jugador o por si solo) debe extender *Movable*, clase que a su vez extiende *Drawable*. Drawable posee atributos como ser la posición, una figura geométrica (para calcular colisiones) y un recuso de imagen o animación. Movable, por su lado, posee atributos como ser la velocidad y direccón de movimiento.

Impresiones del juego

A continuación muestro el progreso al respecto mediante las siguientes capturas de pantalla:







Tecnología utilizada

Para la creación del juego he decidido utilizar el lenguaje Java, por su portabilidad entre sistemas operativos y por familiaridad con el mismo. También he hecho uso del framework *Slick2D* para la creación del juego, este fue escogido por mi experiencia en juegos pasados.

Referencias

- Repositorio del proyecto: https://github.com/dsbarrionuevo/MFTH
- Slick2D: http://slick.ninjacave.com/