Mader Blanco, Conrado
Menzella, Facundo
Ramundo, Federico 09/06/2011

Trabajo Práctico Especial

Programación Orientada a Objetos

Informe

Jerarquía diseñada

Usamos una clase "Game", la cual maneja todo lo referido a back-end. En un comienzo solo teníamos un "Desktop", pero esta nueva clase, además almacena un estado de juego, el cual es observable desde el front-end.

Con respecto a los personajes, tenemos una clase "Character "de la cual extienden todos. Nos pareció correcto hacer una diferencia entre el héroe y los demás personajes, dado que tienen comportamiento diferente. Lo mismo con los enemigos, hicimos una distinción entre "Golem", "Snake" y "Dragon"; pensando en un futuro que se quisiese modificar algo, seria mas reutilizable.

Algo parecido nos paso con las "Tiles". Tenemos una clase "Tile" la cual es abstracta. De ahí hicimos una diferencia entre celda caminable y no caminable. Usamos una "Enemy Tile" y otra "Free Tile". También tenemos una "Wall" que extiende a Tile, que aunque es de las "no caminables", no es una Enemy Tile.

Problemas Encontrados

Iluminación del tablero.

El problema se presento en decidir, si cada celda debería tener un booleano que indique si estaba iluminada o no, o tener otra matriz de booleanos, de las mismas dimensiones que la de celdas, que indique lo propio.

Decidimos que lo mas indicado seria la segunda opción, dado que al recorrerla no deberíamos acceder a la celda, solo verificar un "true" o "false", y al iluminar las 8 celdas aledañas al héroe, sería mas eficiente.

Guardado de la partida.

En un comienzo, optamos por guardar el objeto "Game" entero, pero nos dimos cuenta que no era la mejor opción. "Game" solo posee un desktop y una variable que indica el estado del juego(playing, won, loose).

Optamos por guardar solo el "Desktop", y agregarle a la clase "Game" 1 constructor, que se utilice a la hora de cargar, el cual recibe un file. La variable del estado del juego, naturalmente se inicia en "playing", con lo cual no era necesario guardarla.

Mader Blanco, Conrado
Menzella, Facundo
Ramundo, Federico 09/06/2011

Disposición de las imágenes

Nos pareció natural, que cada imagen, este almacenada en la misma clase a la que pertenecía. Esto tiene sentido desde el punto de vista lógico, pero no desde el orientado a objetos.

Al ser parte del front-end, la clase no debería saber que imagen se le asigna, con lo cual en la clase “Window”, encargada del mismo; colocamos un HashMap de Strings a Image el cual se inicializa a partir de un método que se llama ya sea tanto en “newGame” como en “loadGame”.

Detección del fin del juego

Para este tema, planteamos 3 soluciones posibles:

La primera, utilizar una variable boolean, la cual se coloque en false, se cambie a true en el momento de fin de juego y que esta consulte si el héroe esta vivo o muerto.

La segunda, lanzar una exception y capturarla.

La tercera, utilizar una implementación de la interfaz observer, junto con una clase que extienda a observable.

La mejor solución que encontramos fue esta ultima, dado que la primera resultaba demasiado imperativo, y la segunda nos parecía que le estábamos dando un mal uso a las excepciones, dado que el fin de juego no es una excepción, sino algo que se espera que suceda.