Informe TPE

2010

Segundo Cuatrimestre

**Grupo 1**

**Integrantes:**

Tomás Mehdi

Federico Di Nucci

Agustina Fainguersch

Explicación del diseño y problemas que surgieron durante la implementación del trabajo.

Cuando comenzamos a hablar sobre el diseño del juego nos encontramos con diferentes ideas que podíamos utilizar. Por eso comenzamos con hacer el árbol de clases y la tabla con sus variables y métodos para ver si nos resultaba coherente y cuál de las ideas propuestas por los integrantes resultaba ser la más apropiada. Lo primero que obtuvimos fue un árbol en el cual solo Player, Cell y Menu extendían de Object. La clase más compleja resultaba ser Cell que contenía una clase para cada uno de los objetos que se podían colocar en el tablero incluyendo Key y Coin que a su vez implementaban la interface Grabable. Pero una vez que comenzamos a escribir el código nos dimos cuenta de que no era coherente que una Key o una Coin fuesen Cells pues esto no es cierto, así que decidimos hacer otro árbol con los objetos Grabbable extendiendo de Object para así poder colocarlas “dentro” de las Cells. Otros problemas que nos surgieron al principio fueron los de las Clases Board y Player. Estábamos todos de acuerdo que el tablero debía ser una matriz pero no sabíamos bien cómo hacer para levantar el nivel en la matriz, es decir no sabíamos si levantar todos los levels al comienzo o ir levantando a medida que el juego avanzaba. Por su parte la clase Player nos causó problemas pues no lográbamos ver cómo debíamos hacer para mover al jugador por el tablero y a su vez mostrar cómo se iban levantando las Keys, las Coins y como se transportaba el jugador.

Pero finalmente nos convencimos de que cada unos de nuestros pequeños objetitos ya fuera una Wall, Coin, Key, Tranporter o cualquier otro, iba a ocupar una Cell en el tablero en un momento dado del juego así que decidimos efectivamente extender a cada uno de ellos de Cell y a este otro de Object. Las otras dos clases importantes del Back que extienden de Object resultaron ser Board y Game y estas trabajarían en conjunto con las Exceptions, los levels y el Player.

Para lograr la diferenciación de las llaves y los transportadores implementamos la interface Coloreable en la cual le asignamos un color a cada juego de llave y puerta o par de transportadores para que así se puediera reconocer gráficamente a cuál nos referíamos. Para esto usamos un array de colores y le fuimos asignando los mismos a cada par de objetos a medida que se cargaban.

Lo que menos nos costó fue separar el backend y el frontend pués diferenciamos bien lo que era el contacto con el usuario y lo visual de lo que era el funcionamiento propio del juego. Además creamos loadAndSave, Parser y test para separar bien esas cosas que no lográbamos encontrar el mejor lugar para ubicarlas. Si bien no nos costó implementar el parser, si nos generó un disgusto ya que como previsto por la cátedra nos quedó muy imperativo.

Luego el resto del procedimiento consistió en ir implementando cada una de las Clases establecidas por nuestro diseño asegurándonos de que cada una funcionaba bien por separado.

Sin embargo cuando ya teníamos el proyecto bastante avanzado nos encontramos con un problema mayor que nos hizo replantear el diseño de nuestro juego ya que no lográbamos encontrar la forma de solucionarlo. El problema consistía en la implementación del movimiento ya que la que habíamos hecho no nos permitía borrar las monedas y llaves a medida que se agarraban, sino que realizaba el movimiento completo y una vez que llegaba al destino del movimiento desaparecían las monedas o llaves agarradas en el movimiento. Esto nos llevó un buen tiempo y nos complicó mucho, pero finalmente pudimos hallar una solución para este problema. La solución consistía en implementar una iterface con listeners llamada GameListener que contaba con los métodos para ejecutar un comando cuando se agarraba una moneda, cuando una BreakableWall recibía un golpe o para cuando el jugador se movía. De esta forma logramos completar la implementación del TPE de manera satisfactoria.