Universidad de Los Andes
Facultad de Ingeniería
Departamento de Sistemas Computacionales

Desarrollo de Mini juegos Lyoko

Integrantes:

Baptista Miguel CI 24135386
Bermúdez Ricardo CI 25171618
Lugo Guillermo CI 20709795
Peña Ender CI 25302230

Profesor:
Mujica Alejandro

Mérida, diciembre del 2017

Mini Juegos Lyoko

Concepción de la idea

La idea principal del equipo era realizar un juego en el que tuviésemos mapas como por ejemplo laberintos, donde fuera posible que hubiese ciertas interacciones con algunos elementos del mismo, utilizando de alguna forma algoritmos explicados en clase sobre grafos.

Los juegos en los que nos basamos o inspiramos fueron algunos clásicos como Bomberman, Pacman y otros no tan conocidos pero bastante entretenidos y que requieren de cierta agilidad e inteligencia como Undertale.

En un principio se tenían planteados dos modos de juego, el primero para que el jugador jugara contra la computadora, y el segundo jugador vs jugador.

Desarrollando el juego

Comenzamos con la idea de desarrollar el juego en C++ utilizando SFML, pero posteriormente nos documentamos sobre las bondades y facilidades que nos brindaba C++ y Java, y nos decantamos por Java.

Una vez decidido que Java iba a ser el lenguaje a utilizar para nuestro juego, comenzamos a la construcción de la interfaz gráfica del juego haciendo uso de la librería Swing. En la función principal definimos cosas como el título del juego, las dimensiones de la ventana, si podía ser redimensionable, etc.

Luego de esto comenzamos a construir la clase Tablero, que es donde se va a desarrollar la primera modalidad de juego. El dibujo del mapa fue cargado desde una clase Mapa que creamos en la cual lo recibíamos de una matriz definida dentro de un archivo. Esta matriz contiene un laberinto que diseñamos para el jugador y uno distinto para el villano (computadora). Para luego comenzar con la realización del dibujado, esto lo hicimos a través de los valores de la matriz, a cada valor le corresponde una determinada textura. Por ejemplo, los 1 en la matriz se pintan como paredes en nuestro tablero porque le asignamos la textura del mismo.

Creamos las clases Jugador y Villano, donde definimos su punto de inicio, su textura, movimientos y algunos otros métodos que usaríamos más adelante.

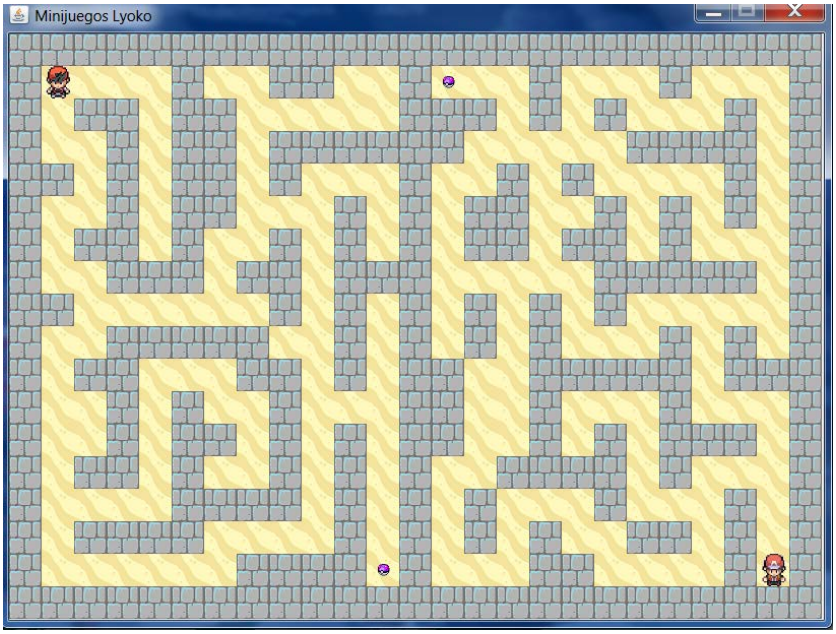
Primero, teniendo ya el jugador mostrado en pantalla y el mapa, diseñamos los movimientos, su cambio de dirección a donde mira el personaje (cambios en los sprites), la interacción con la meta y el mensaje de victoria.

Después partimos de la idea de que el Villano debía moverse tomando un recorrido sobre un grafo. En consecuencia creamos una clase Grafo, tomando como base que una casilla en el mapa simbolizaba un nodo y sus casillas adyacentes los vecinos del nodo (siempre y cuando no sea una pared), por esta razón no consideramos los arcos.

Al principio pensamos que el recorrido del Villano fuese el camino mínimo resultante de aplicarle DFS al grafo, pero nos dimos cuenta que esto haría que siempre recorriese la

mínima cantidad de casillas, por lo que iba a ser difícil ganarle. Así que el recorrido de él lo tomamos como el camino que generaba el DFS hasta llegar al nodo deseado (la meta). Tuvimos que decidir entre si el Villano se movía cada cierto tiempo o por cada movimiento del jugador, y finalmente establecimos que lo más justo era que el Villano se moviera sólo si el jugador lo hiciera.

Cabe destacar que la clase Grafo guarda en un arreglo de enteros, posiciones X y Y, es decir los desplazamientos se tomaban cada par de coordenadas. Esto lo recibía el Villano y es lo que hablamos anteriormente del recorrido.



Resultado final de la Modalidad 1.

Continuamos a la siguiente modalidad, creamos otra clase tablero (Tablero2), donde dibujamos todo el juego diseñamos otro archivo que contuviera el segundo mapa. Creamos además una matriz entera idéntica (en principio) a la del archivo, la utilidad de esta va a ser que en ella modificaremos algunas casillas de acuerdo a algunas interacciones en el juego.

Para esta modalidad ya no precisamos de un villano, sino de otro jugador, el cual tendría las mismas características que el primero.

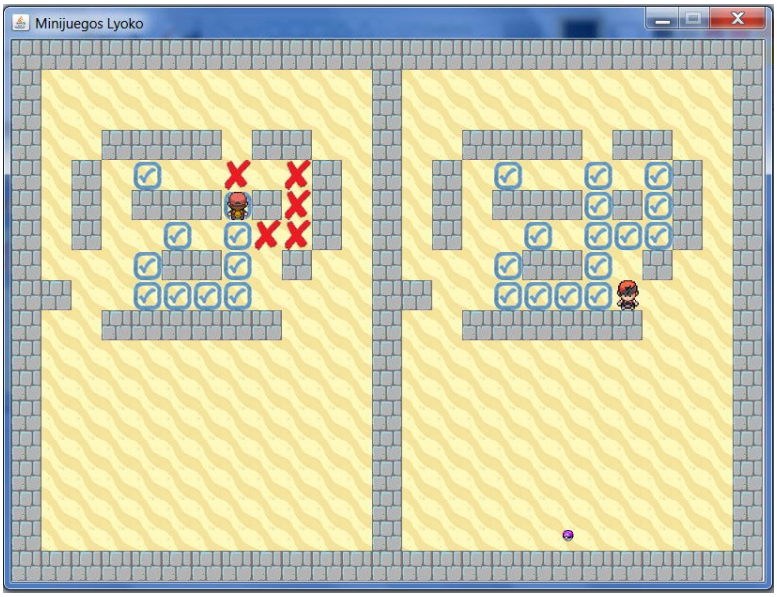
Este es un mapa espejo, donde hay ciertas casillas marcadas con X, y los jugadores deben pasar por cada una de ellas y regresar a la meta, considerando que cuando pisas una, no puedes volver a hacerlo.

Para manejar la Lógica de este mapa, se nos ocurrió para las casillas X (antes de pisarlas) le dimos un valor diferente de piso y pared, entonces al pasar por encima de esas casillas, cambiamos el valor de esa posición en la matriz definida dentro de la clase. Para que cuando hiciera el repintaje cambiara su textura (debido a las condiciones del pintaje), se viera diferente y además, no pudiera volver a ser pisada (se toma como colisión).

La meta se hace visible sólo cuando se han pisado todas las casillas X.

Para el jugador 1 definimos los movimientos con w, a, s, d, y para el jugador 2 tomamos up, left, down, right para moverse hacia arriba, izquierda, abajo y derecha respectivamente.

Como con el diseño de las casillas se hace posible quedarse atrapado, decidimos dar algo de ayuda, por lo que se implementó un botón de reinicio para cada jugador, que lo devuelve al estado inicial (resetea las casillas de su lado del mapa y su posición). Cuando se nos ocurrió esto, se lo incluimos también al primer mapa.



Resultado final Modalidad 2

Una vez terminada esta modalidad, pensamos en ¿por qué no creamos un tercer mapa?, por lo que comenzamos a hacer bocetos de cómo sería (ya que no estaba planeado).

Tomamos la idea del laberinto y de que el juego fuese multijugador, para este decidimos agregar ciertas interacciones, como portales, los cuales te transportan a otra parte del mapa, pudiendo regresar por el mismo (uno te teletransporta al otro y viceversa).

El juego consiste en recolectar ciertos elementos ubicados en el mapa, con la finalidad de desbloquear la meta, para ello debes utilizar los portales y encontrar el camino hacia la meta.

Para la realización de este mapa, nuevamente generamos el mapa partiendo de una matriz prediseñada contenida en un archivo, y creamos una matriz en la clase Tablero3, en la cual representábamos los portales de forma específica, identificándolos cada uno con números diferentes y luego establecimos las relaciones de cada pareja de portales, por ejemplo: sabíamos que si el jugador se movía hacia la casilla (2, 3) (la cual representaba el portal 1), lo teletransportamos hacia la casilla (7, 13), una casilla por encima del portal, esto para que no apareciera encima del portal pareja y no se generaran posibles errores. Se hizo lo análogo para el resto de las parejas de portales, ubicamos el jugador en una posición adyacente al portal (que no fuese pared).

Análogamente al mapa anterior las modificaciones que se hiciesen debido a las acciones de esta modalidad, las guardábamos en la matriz que está en la clase. Por ejemplo, cuando pasamos por la casilla que tiene el objeto recogible, este va a desaparecer en el repintaje debido a que modificamos el valor de la casilla correspondiente en la matriz (en este caso se sustituye un 2 que este en la matriz por un 0).

Se creó una variable para cada jugador que lleva la cuenta de los objetos que han recogido cada uno, para cuando haya recolectado todos, se abre el paso a la meta.

Resultado final modalidad 3

Recomendaciones finales

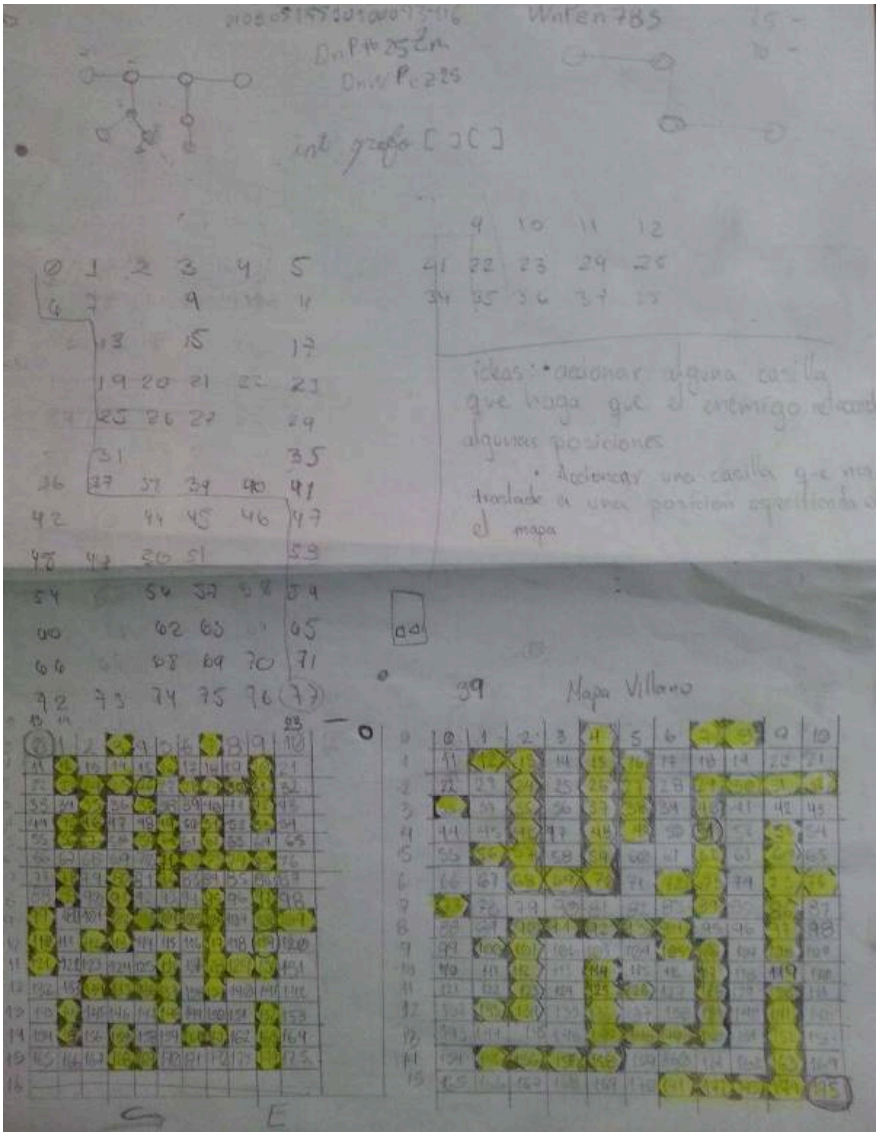
Algunas ideas para continuar con el desarrollo de estos mini juegos, son el añadir sonidos, por ejemplo al pisar las casillas, al ganar, moverse, etc.

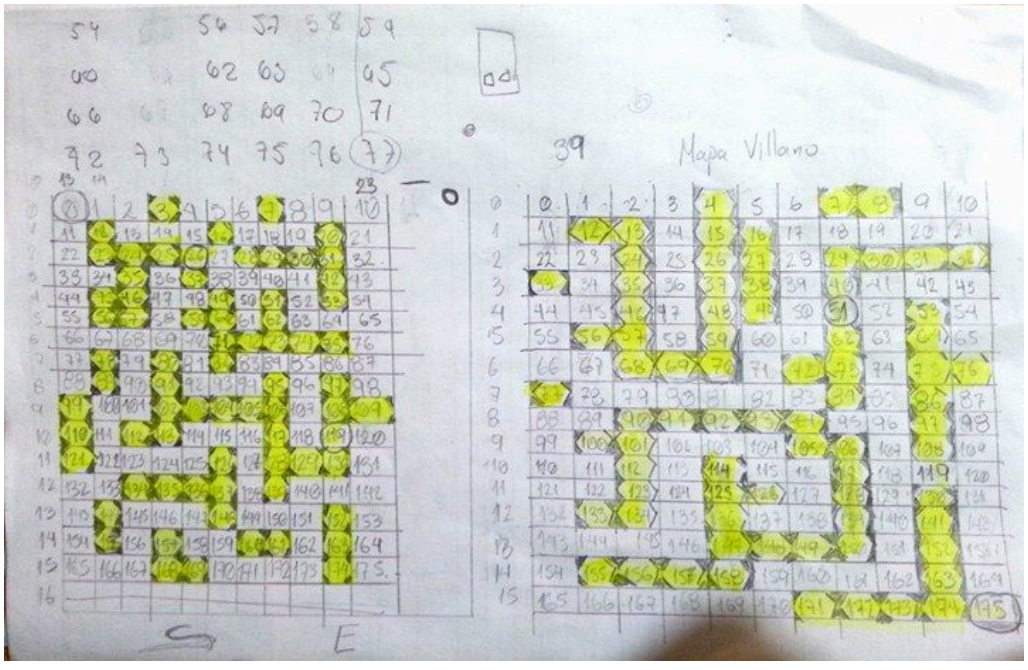
También un menú para escoger el modo de juego que se desee jugar.

Un mensaje de victoria, derrota o empate más elaborado.

Variedad de niveles (distintos mapas) para los distintos modos de juego.

Bosquejos de ideas durante el desarrollo del proyecto





el juego 1

monedas

0 Piso
1 - Pared
2 - Monedas
3 - Portales
4 - Regas

En Matriz

Jugador 1

Portal 1 hacia 2 - (3,2) → (13,6)
Portal 2 hacia 1 - (13,6) → (3,2)
Portal 3 hacia 4 - (5,4) → (1,3)
Portal 4 hacia 5 - (1,6) → (6,4)
Portal 5 hacia 6 - (16,1) → (8,11)
Portal 6 hacia 5 - (8,11) → (16,1)

Jugador 2

Portal 16 hacia 17 → (13,14) ⇄ (3,23)
Portal 18 hacia 19 → (6,21) ⇄ (14,19)
Portal 20 hacia 21 → (10,13) ⇄ (5,12)