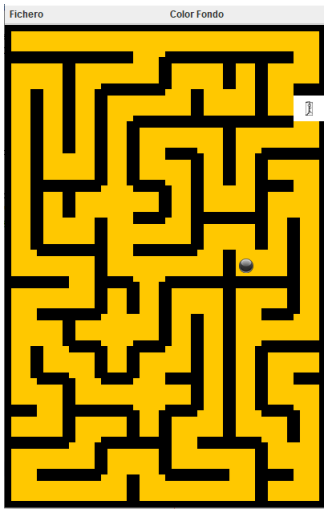
Grado de Ingeniería Informática - UIB

21707 PROGRAMACIÓ II - Grup 3

Prof.: Miquel Mascaró Oliver

Practica III
Juego del Laberinto



Pavel Ernesto García Lorenzo

pavel.garcia1@estudiant.uib.cat

DNI: 45695925p

video 1: https://www.youtube.com/watch?v=Ss9a9iDhvTo

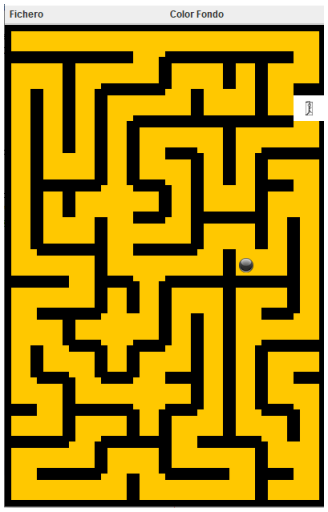
video 2: https://youtu.be/k0Kaa-SFMr4

2020 - 2021

Índice

1. Introducción

La práctica consiste en implementar un juego para encontrar la salida de un Laberinto , este laberinto se debe mostrar graficamente. Los datos acerca de los laberintos del Juego se obtienen a partir de diversos ficheros donde se indica la posición de las paredes, de la salida y el tamaño del laberinto. La posición de la ficha o jugador se establece aleatoriamente en una posición de laberinto. Una vez el jugador llega a la salida , se considera una victória.

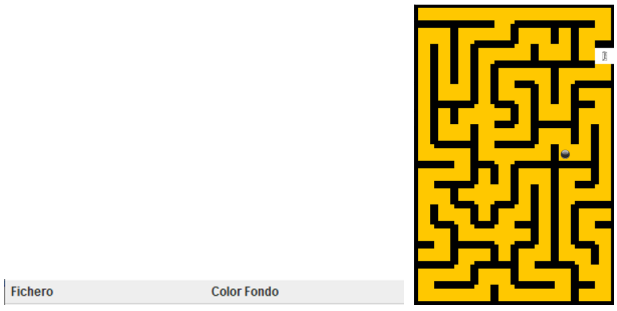


Como funcionalidades extra se ha implementado la opción de cambiar el color de fondo y al obtener una victoria se muestra el camino seguido.

2. Analisis

Al observar la interficie que queremos implementar visualmente lo primero que podemos detallar es una ventana que se compone de diversas secciones:

Una barra o menu con diversa opciones y el juego en sí un laberinto



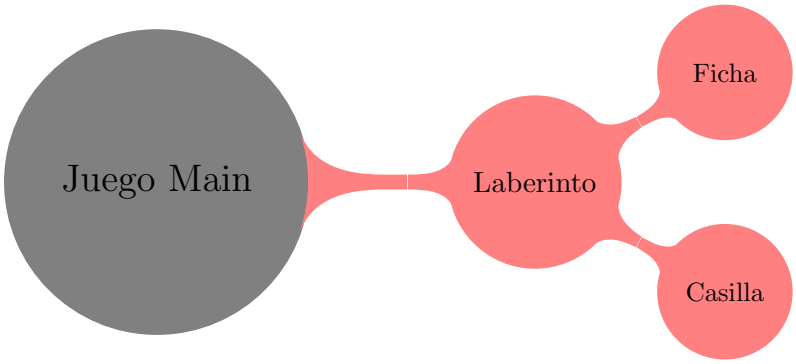
Analizamos el laberinto en profundidad.
El laberinto tal y como especifica el enunciado se compone de diversas casillas y estas casillas pueden diversas configuraciones de paredes.Estas paredes le dan la apariencia de un laberinto.Una de estas casillas puede ser la salida.Tambien sabemos que en una casilla puede estar el jugador o ficha del laberinto.

3. Diseño Descendente

En esta sección se detallará el diseño descendente llevado a cabo para el desarrollo de la práctica. A partir del analisis anterior se desarrolla un esquema principal.Este se especificará indicando sus dependencias y sus métodos.

3.1. Esquema

En la siguiente figura se puede observar el diseño descendente implementado gráficamente que surge a raiz del analisis realizado previamente , basarse en este esquema nos permite obtener una visión general de como solucionar la práctica.



3.2. Clases y Métodos implementados

En esta sección se mostrará el proceso y la estructura paso a paso de la realización del proyecto así como una explicación de la funcionalidad de cada una de las partes y clases, método a método.

3.2.1. Main

En esta clase se establecen los métodos necesarios para gestionar el juego Laberinto y mostrarlo por pantalla. Esta clase principal hereda de la Clase *JFrame* .También la clase controla el flujo y el funcionamiento del juego (*entradas de teclado y mostrar las componentes de la ventana*) los métodos utilizados son:

- **public static void main(String[] args):**
- **public JuegoLaberinto():** Método constructor que instancia un JuegoLaberinto ,en este método se especifica el tamaño de la ventana,posicion y contiene el método initComponents().
- **private void initComponents():** Este método inicializa todas las componentes presentes en la ventana del juego.La barra menu y todas sus componentes , el laberinto inicializado con el primer laberinto *maze1.txt* , ademas del KeyListener encargado de la entrada por teclado.

3.2.2. Laberinto

En esta clase podemos encontrar todos los métodos que permiten gestionar un Laberinto así como su implementación , también encontramos se método paint component para poder visualizarlo.

- **public Laberinto(File f):** Método constructor que instancia un Laberinto dado un fichero. Este fichero contiene los datos del laberinto. A raiz de este fichero se completa una matriz de objetos *Casillas* instanciados con los valores pertinentes y se inicializa la posición de la *Ficha*.
- **public void paintComponent(Graphics g):** Pinta un objeto laberinto.
- **private void inicializarFicha() :** Inicializa la posición inicial de la *Ficha* generando dos número aleatorios para el eje i y el eje j donde $i \in (0, nColumnas-1), j \in (0, nFilas-1)$.
- **public void moverFicha(int i, int j):** Este método se encarga de colocar una ficha en la casilla correspondiente,solo se puede avanzar un posición por lo que se incrementa la posición en i o en j pero no ambas($\triangle i \oplus \triangle j$).

3.2.3. Casilla

En esta clase se encontramos las funcionalidades e implementacion de una casilla del laberinto,una casilla puede tener distintos estados y diversas combinaciones de paredes.Se designan y se distinguen las paredes segun la orientacion pared Norte,Este,Sur,Oeste y se representa mediante un array o vector donde sus valores $n, e, s, o \in \{0, 1\}$.

Los estados posibles de una casilla són:

- *Meta : la casilla es la salida del laberinto.*
- *Jugador : El jugador esta en la casilla.*
- *Pisada : El jugador ha pasado por la casilla.*

- **public Casilla(Rectangle2D.Float r, int pN, int pE, int pS, int pO):** Método constructor que instancia un objeto Casilla a partir de un objeto Rectangle2D de donde obtenemos la posición global,y si inicializan la informaciñon acerca de las paredes.
- **protected void paintComponent(Graphics g):** Método que pinta una casilla con sus paredes,considerando los estados posibles por ejemplo en el caso de que sea la salida.

3.2.4. Ficha

Es la ficha del juego , representa al jugador, la ficha no almacena su posición debido a que la posición de la ficha está relacionado con el laberinto en sí, ya que este se encarga de mover la ficha.

- **public Ficha():** Método constructor que instancia un objeto Ficha.

- **void paintComponent(Graphics g, float x, float y):** Pinta la ficha en la posición determinada por parametro x , y .

4. Conclusión

En esta práctica he podido reforzar el manejo de las interfaces gráficas de java, así como el del diseño descendente que me ha permitido a partir de una idea genera ir la desglosando para poder implementar las funcionalidades.