Trabajo Práctico 1
Rompecabezas Deslizante

Profesores

- Daniel Bertaccini, Gabriel Carrillo

Integrantes del grupo

- Tomas Baez, Maximo Alegre, Milagros Ibarra, Bruno Berini

Introducción

Para poder plantear la solución de este trabajo práctico, utilizamos los conceptos vistos en clase de Separated Presentation. En particular, utilizamos la arquitectura de GUI más sencilla de implementar, la cual es Forms and Controls.

En la estructura de las clases que utilizamos, cada una tiene sus responsabilidades bien definidas de acuerdo a forms an controls, siendo:

Tablero y celda: las clases encargada del código de negocio.

InterfazPuzzle e InterfazMenuPuzzle: clases encargadas del código de la interfaz

Usar los conceptos vistos en otras materias anteriores para que las clases no tengan demasiado acoplamiento y no se genere dependencia una de otra, nos sirvió para poder implementar extras y modificar el código más fácilmente sin necesariamente tener que “romperlo”.

En general también se extrajeron muchos métodos para hacerlos privados y no violen el encapsulamiento de la clase. Además, esto mejora la legibilidad de todo el código de cada clase y sus métodos particulares.

Reglas del juego

Para inicializar el juego, se debe ejecutar la clase interfazPuzzle. Allí, se le presenta al usuario una pantalla, en la cual debe seleccionar una imagen, luego la dificultad, y en iniciar se inicia el rompecabezas deslizante para resolver. Según la dificultad que se elija, el rompecabezas se crea de diferente tamaño. En fácil el tamaño es 3x3, en normal es 4x4 y en difícil es 5x5.

El jugador debe resolver el rompecabezas desordenado, tal que la imagen quede ordenada, siendo la última celda (abajo, en la esquina inferior derecha) el que quede vacío. Es decir, solo hay una solución posible.

Para implementar la imagen, se asigno a cada parte de imagen un número correspondiente, por lo que, al fin y al cabo, siempre estaremos ordenando números.

Ejemplo, dado este rompecabezas deslizante:

1	5	6	8
15	10	9	2
12	7	4	3
13	14		11

Para considerarse que esta bien resuelto debe quedar:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

El jugador puede clickear sobre la celda adyacente al vacío que quiera mover, o sino puede usar las flechas para mover la celda que corresponda al adyacente al vacío, y de esa manera ir resolviendo el juego.

Implementación

A partir de nuestra elección como patrón de diseño Forms and Controls, implementamos el código en un paquete general llamado Puzzle, que contiene tanto las clases de negocio como de interfaz visuales. Sin embargo, están cada una de las clases, sobre todo las de negocio y las de GUI, separadas entre sí, para respetar el principio de “Separated Presentation”

Lógica de Negocio

La lógica del juego se implementa en la clase Tablero y la clase Celda. Respecto a Tablero, se utiliza una matriz bidimensional de tamaño n x n para representar el tablero del rompecabezas. Dentro del tablero, cada una de las celdas puede contener un valor numérico y sólo una de ellas estar vacía. La responsabilidad de la clase Tablero es manejar toda la lógica de cómo se inicializa, cómo se mueve, como interaccionan las demás celdas alrededor de la celda. Por otra parte, la clase Celda sirve para saber su posición en el tablero, y ver su valor numérico(si es que lo posee) o ver si es vacía.

En general, volviendo a la clase Tablero, la lógica consiste en varios métodos que se usan luego en la interfaz visual. Apenas se crea el tablero, lo primero que se hace es mezclarlo, que internamente en el método de mezclarTablero se usa collection.shuffle. Para mover una celda, se verifica que sea adyacente a una celda vacia, y que el movimiento sea válido. Una vez comprobado eso, se procede a intercambiar la celda vacía con la que el usuario desea mover. Para este paso es importante tener siempre actualizada la posición de la celda vacía (en fila y columna del tablero) ya que nos permite realizar los movimientos. Luego, por cada movimiento realizado, incrementamos la cantidad de movimientos en 1.

También otro de los métodos en general importantes que resumen la lógica, es el de comprobar si el tablero está bien resuelto, tal como se detalló anteriormente en este informe. Para eso se recorre el tablero completo con un contador, y compara el valor del contador con el valor de la celda que avanza. Si todo es correcto y la última es la que esta vacía, devuelve true y el juego se da por terminado, sino devuelve false y el usuario debe continuar resolviendo.

Interfaz Visual (GUI)

Dado que utilizamos Forms and Controls, los controles se ubican en la clase InterfazPuzzle, con métodos para permitir el movimiento al clickear o mover las flechas, que se encuentran exclusivamente en esta clase. Se puede ver más específicamente luego el concepto de separated presentation, cuando en varios de los métodos utilizamos llamadas a métodos que se encuentran en la clase Tablero, que corresponde a la lógica. Sin embargo, nunca utilizamos en la clase de negocio llamadas a métodos de la GUI.

Además, en esta clase “InterfazPuzzle” se muestra lo que es el tablero gráfico del juego, junto con un texto con la cantidad de movimientos, y la imagen guía/pista para el jugador.

Por otra parte, en la clase InterfazMenuPuzzle lo que realiza exclusivamente es ejecutar el menú principal del juego, que permite seleccionar imágenes y la dificultad para jugar el rompecabezas. La división de la imagen en partes iguales la realiza una clase aparte dedicada ImagenDividida.

Extras

Como se mencionó con anterioridad en este informe, cambiamos que en vez de resolver con números del 1 al 15, se tenga que armar una imagen, de la cual el usuario tiene una guía para armarla correctamente. Para esto se implementa otra clase aparte (ImagenDividida) que divide la imagen en tantas partes como el nivel que elija el usuario lo solicite, y asigna cada parte a una celda en específico.

También incluimos los niveles, con diferentes tamaños de rompecabezas, según el nivel elegido. Fue muy sencillo implementarlo, ya que solo tuvimos que cambiar la variable de tamaño del rompecabezas, con el setter de tamaño de tablero.