CS1142: COMPUTER PROGRAMMING II

TERM PROJECT

PROJECT NAME: MOVE THE BALL

İNSAF ÖZGE BERKTAŞ

150118072

AHMET ONAT ÖZALAN

150118054

CS1142: Computer Programming II, Spring 2019

Date Submitted: May 12, 2019

**Problem Definition**

There is a board game we have to implement. This board game has a moving ball that has to move from a start point to end point and also this board game is 4 x 4 (has 16 squares). Start point is the blue tile and the end point is red tile. The blue and red tile cannot move. There are also grey tiles which represents free spaces. The empty, pipe, curved pipe tiles can move. And tiles cannot move diagonally.

There are input files that contains id, type and property to build the board game. Those input files of each levels will help us to create the board game. After the creation of the board game we have to drag tiles to make a path for ball to move. After we make the path for the ball for each level, we will return to first level of the game.

**Implementation Details**

UML diagram for the **BoardPane** class.

|  |
| --- |
| *BoardPane*    - level: int  - levelInfo: String  - moveCount: int  - moveCountText: Text  - text: Text    + loadLevel(): void  + readLevelInfo(level: int): void  - configureEventHandler(tile: ImageView): void  - isSourceValid(sourceTile: ImageView): boolean  - getTile(coordinateX: int, coordinateY: int): ImageView  - isTargetValid(sourceTile: ImageView, targetTile: ImageView): boolean  - isLevelCompleted(): boolean |

The **BoardPane** class displays the game board with the help of some methods in it.

he **LevelInfo** data field is String array. In **readLevelInfo** method the **LevelInfo** data field will be loaded with the file names of the game pictures on each level.

 The **readLevelInfo** reads the level information of input text files. Firstly with the help of the scanner reader the input files will be read according to current level. Then parts string array will store the the id, type, property values of the tiles. Then **parts’s** first and second element will be combined together with “.jpg”. **levelInfo** string array will be loaded with those combined string names of the tiles.

In **configureEventHandler**(**ImageView**: **tile**) allows user to drag tiles. There is a nested if statements in this method. In the first if statement condition there is **isSourceValid** method that checks if the source tile (first tile) can move or not. If the condition is true (if the source tile can move) **targetTile** from **ImageView** class will be initialized to null and then there is a for loop. This for loop searches the target tile object in the pane's objects and assigns its reference to the **targetTile** variable. If the first if condition is not true “**Select a valid source tile”** will be printed.

In the first if statement block there is second if statement that checks whether the target tile is movable. If the condition is true, some statements in the if block replaces the source tile and the target tile. If the condition is not true, “**Select a valid target file**” message will be printed.

The method parameter is tile from **ImageView** class, tile stores the information of the pictures. So **isSourceValid** and **isTargetValid** methods checks the tiles if they are movable or not according to **ImageView tile**.

If the **isLevelCompleted**() method returns true, and if the value of level is less than 5, **alert.setContentText** is a notification alert that shows us the message “You have completed the level **level**(int value) ! ”. Then we reset the move count as 0, we increment the value by 1. Then we call the **loadLevel** method in the **configureEventHandler** again.

**isSourceValid** method basically checks whether the source tile is movable or not. When mouse selects the first tile, the **isSourceValid** method will be called and the tile become **sourceTile** in the method. Then **sourceTile** will be checked with the help of **getImage**, **getURL**, and **endsWith** methods to see whether the **sourceTile** is movable or not.

If **sourceTile** is movable we can check the **isTargetValid** condition after the mouse releases. After mouse releases on another tile, **targetTile** will be updated and **IsTargetValid** will be called. This method checks if the **targetTile** is movable. There is **sourceTile** and **targetTile** from **ImageView** class as parameter list. In the first if statement we see that the target tile has to be empty free tiles because source tiles can only change places with empty free tiles. If the target tile is empty free tile, the condition is true. After that we have to get the x and y coordinate values of source and target tile. We did it because we have to develop an algorithm for **sourceTile** to move. Because **sourceTile** can only replace with empty free tiles that places on **sourceTile’s** up, down, right and left. So we get the difference of x coordinates of the tiles and difference y coordinate of the tiles, and the summation of these values must be equal to 75 for a suitable replacement between **sourceTile** and **targetTile**.

So the method returns false if sum is not equal to 75.

If sum is equal to 75 then the method returns true. If the return value is true, we replace the coordinates of **sourceTile** and **targetTile**.

**moveCount** will be incremented in each move (in each replacement).

We use the **setText** method to be written on the screen how many moves has been made during each level.

The **isLevelCompleted**() is a boolean method and it is a solution for each level. In **isLevelCompleted**() we define some statements for levels. For level 1, level 2 ,level 3, level 4 and level 5 we created some tiles from **ImageView** class and initialized their values with **getTile** methods. Parameter list of **getTile** contains their own x and y coordinate values. For example;

**ImageView** tile2\_1 = **getTile**(0, 75). In **getTile** method returns the tile that takes place in the given coordinates. So the return value is a tile(object). If those tiles in the **isLevelCompleted** method matches with the **getTile** method’s return values the level is completed. So **isLevelCompleted**() method returns true.

The **getTile** method finds which tile takes place based on the given coordinates. The for-loop checks every object in the pane that has the same x and y coordinate values.

The **loadLevel** is a void method. After calling of the **readLevelInfo** the **levelInfo** String array stores the names of the images as string. So in the for-loop we assign the addresses of images to tile object from **ImageView** class. After **setFitWidth** and **setFitHeight** methods on each iteration each tile becomes 75 x 75 square. **setX** and **setY** methods help place the tiles like 4 X 4 grid. After those statements **getChildren.add(tile)** statement creates the final pane. After that **moveCountText** will take place on the pane based on **setX**, **setY** and **add()** methods.

|  |
| --- |
| *LaunchGame*    + start(primaryStage: Stage): void  + main(args: String[]): void |

We created **boardPane** from **BoardPane** class.

**boardPane** calls **loadLevel**() method. The ImageView has been created for each tile in this method. We created **scene** from **Scene** class and we set the sizes of the **scene.**

**setScene** method sets the **scene** into **primaryStage**.

**setResizable** method allows us to resize the scene. We decided to initialize **false** in the method so that user cannot resize the scene while program is running.

**primaryStage.setTitle**(“Game Board”) sets the title as “Game Board”. **primaryStage.show()** makes the scene and the stage visible.

**Questions & Answers**

**1) Which parts are complete/incomplete in your project?**

There are two major issues of the program. First problem is; after the user completes the level,

the ball does not move from starting tile to ending tile. There is no moving animation for the

ball.

Second problem is; the program only works with the input files attached in the .rar file. The

program can read the other input text files but the game can’t be played. So the program is

not dynamic.

**2) What are the difficulties you have encountered during the implementation?**

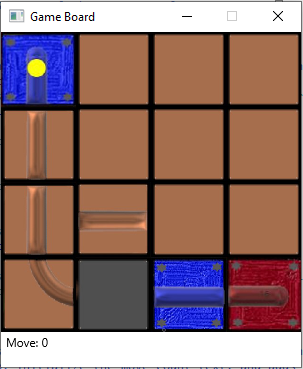
The most hardest part of the implementation was writing the codes for dragging tiles.

**3) What are the additional functionalities of your project added by your team?**

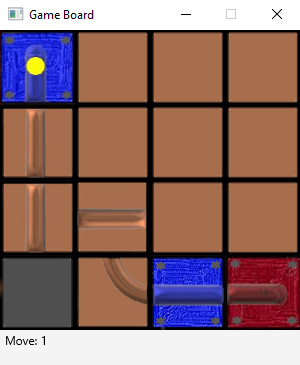
The program prints some messages when the user completes the level, finishes the game, and selects an invalid source or target tile.

**Test Cases**

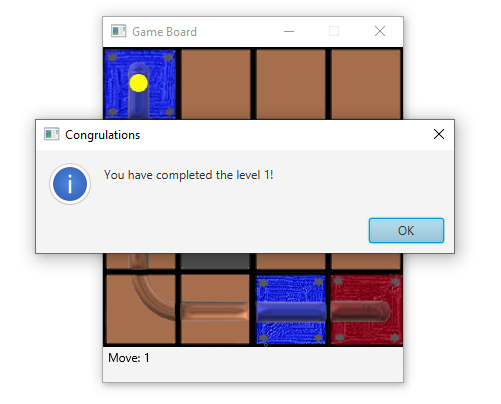
**1) Level 1**



*Appearance of the first level after the program launches.*

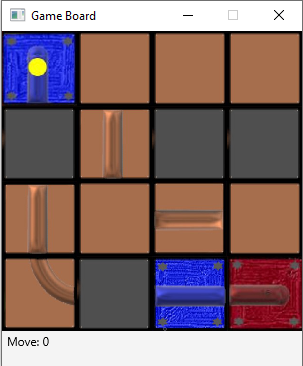


*The move is incremented by one after a move is performed by the user.*

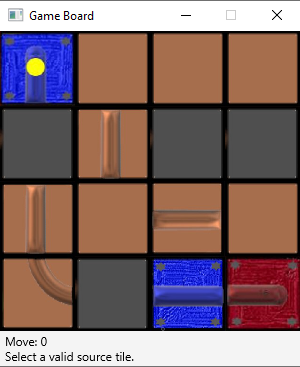


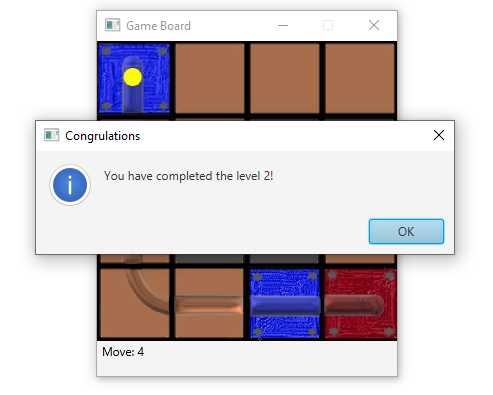
*The message is displayed after the user completes the level 1.*

**2) Level 2**

****

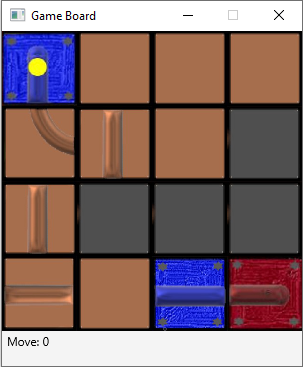
*The appearance of level 2 after the user completes the level 1.*

*If the user selects an invalid source tile, the message is displayed.*

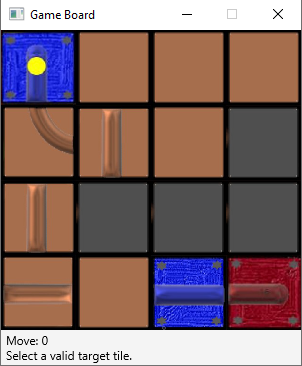


*The message is displayed after the user completes the level 2.*

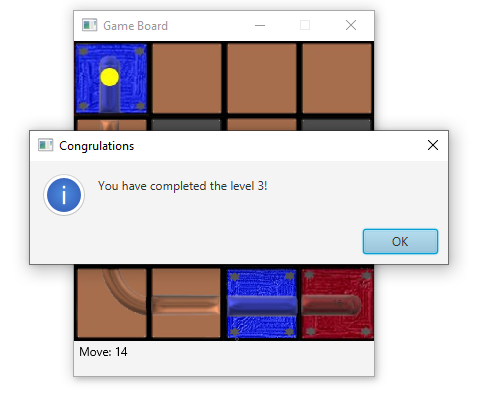
**3) Level 3**

****

*Appearance of level 3 after the user completes the level 2.*

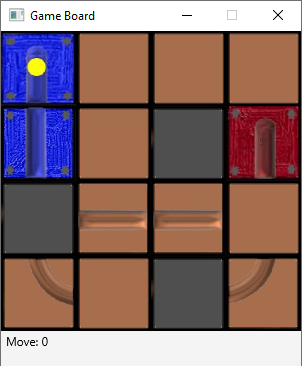


*If the user selects an invalid target tile, the message is displayed.*

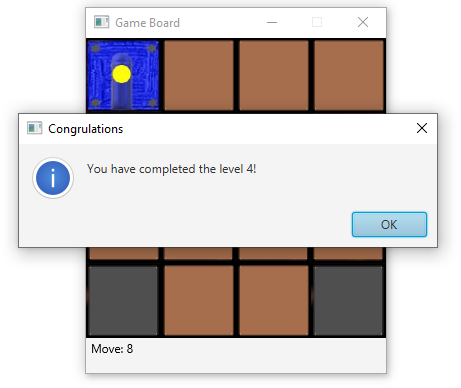


*The message is displayed after the user completes the level 3.*

**4) Level 4**

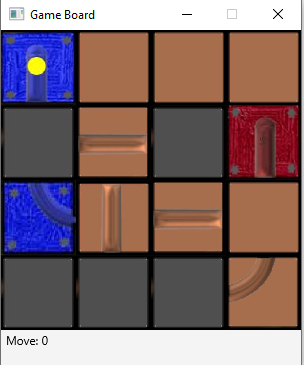
****

*Appearance of level 4 after the user completes the level 3.*

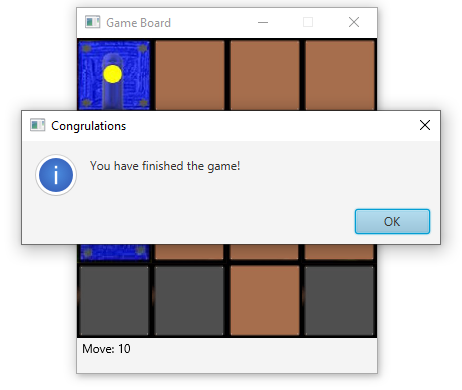


*The message is displayed after the user completes the level 4.*

**5) Level 5**

****

*Appearance of level 5 after the user completes the level 4.*



*The message is displayed after the user completes the level 5 and finishes the game.*