CSE 1142 - COMPUTER PROGRAMMING II

TERM PROJECT

Project Name: Pipe Puzzle

Authors:

1) 150117043 Erdem AĞCA

2) 150117905 Erhan YALNIZ

CSE1142: Computer Programming II, Spring 2019

Date Submitted: May 12, 2019

PROBLEM DEFINITION

The game consists of 5 different levels. The purpose of this game is to connect the pipes to provide that the ball moves.

In these boxes, different box types such as "starter", "end", "empty free", "empty", "horizontal and vertical pipestatic", "horizontal and verticalpipe", "00,01,10,11 curved pipe" are placed.

***Starter*:** The starter tile is the pipe where the ball starts to slide at each level and there is one at each level. Starter tiles can not move. Starter tiles can be vertical and horizontal.

***End*:** The end tile is the pipe where the ball finishes to slide at each level and there is one at each level. Starter tiles can not move. End tiles can be vertical and horizontal.

***Empty Free*:** Empty tiles are tiles without pipe. These tiles cannot move but other tiles can move to the position of this tile.

***Empty*:** Empty tiles are tiles without pipe. Empty tiles are tiles without pipe. These tiles cannot be replaced by other tiles, but they can move to the positions of empty tiles.

***Pipe Static*:** Pipe static tiles cannot move; their locations are static and they can be horizontal or vertical.

***Pipe*:** It is the most common type of pipe in the game. They can be horizontal or vertical and can moved to the position of empty free tiles.

***Curved Pipe*:** Curved pipes are available in four different shapes. Four of these numbers are 00,01,10,11. Different names are given in terms of shape properties. Curved pipes can be placed in the empty tiles position.

The game consists of 5 levels, each with different levels of difficulty. These levels are shown in the game by reading from the input file. There are 5 different buttons of levels on the game screen. When the levels cannot be completed, the buttons are not active and they are not switched to the next level.

Gameplay is provided by mouse. Movable tiles can be moved to the positions of the empty tiles when they are moved by holding them on top. The movable tiles can be carried one unit if there are empty free tiles in the vertical or horizontal directions, but they cannot be moved diagonally.

In the opening screen of the game, it writes a welcome message and there are buttons to select the levels of the game. When we start the game, there is a counter that calculates the number of moves the player has made and able to print on the screen how many moves the player has made until the end of the game. Furthermore the game screen also includes the "Return to Main Menu" button, which allows the players to return to the main menu at any time. When the players click on this button, they return to the home screen and they can select level.In the game, the player checks the level by clicking on the “Check” button if the player thinks they have completed the level by combining them correctly. If the level is completed correctly, the ball starts to slide from the start pipe to the end pipe.

IMPLEMENTATION DETAILS

1) Implementing a CellData class with the following UML diagram.

CellData	
+	id: int
+	type: String
+	property: String
+	CellData(id : int, type : String, property : String)

- The aim of this class is to carry the required values of the cells.
- Each box has an id, a type, a property.
- Id value which type is an integer, is represents to id number dedicated to cells. Type value which type is a string, is represents to types of boxes. Information about the types of boxes is given above but if we want to talk about briefly, these boxes have 7 different types and this type value indicates these types. Property value which type is a string, is represents to properties of boxes and again if we want to talk about briefly, these boxes can have different properties such as 00, 01, 10, 11, vertical, horizontal, free, none, etc.
- We have a CellData constructor whose parameters are id, type, property and whose accessibility is public.

2) Implementing a Game class with the following UML diagram.

Game	
+	board: CellData[][]
+	unlockedLevel: int
+	Game()
+	generateMapData(inputFilePath: String): void
+	isLegal(x0: int, y0: int, x1: int, y1: int): boolean
+	makeMove(x0: int, y0: int, x1: int, y1: int): void
+	gameEnded(): boolean

- Game class consists of the game's backend, mechanics and actions.
- board variable which a multi-array of CellData type is store cells in order of indexes: x and y. unlockedLevel variable which type is int stores last unlocked level in case of level change, saving and loading.
- We created 3 enum variable so that the program can be written and read easily.Therefore, we used this variable type to avoid confusion, instead of assigning a single numeric value to many variables in the program.
- We have a Game constructor whose parameter is empty and whose accessibility is public.In constructor, we set the "unlockedLevel" variable which we created to control locked levels, to 0 so in case of level cannot be loaded it will be first level. Furthermore, we Set board to 4 by 4 array to simulate gameboard for pipe game.

- We have generateMapData method whose parameters is inputFilePath type string and return type is void.This method generates gameboard inside the backend using inputFilePath and we defined in it "br" variable which type is BufferedReader to read level from inputFilePath line by line. This method creates a cellData of the CellData type and holds ID, type, and property, and places these values in the coordinates specified in the row and column. It prints an error messages when the input file encounters an error, close BufferedReader whatever happens.In the same time, if it cannot be closed it prints an error message.
- We have isLegal method whose parameters are x0,y0,x1,y1 types int and return type is boolean. This method checks if move is legal(applicable) by given coordinates of cell that is being moved (x0,y0) and cell that is new position for cell to stay (x1,y1) finally it returns legality of move.After that if the block is imovable it returns false.Furthermore, it controls the displacement state of a unit on the horizontal and vertical axises.
- We have makeMove method whose parameters are x0,y0,x1,y1 types int and return type is void.It changes the coordinates of cells which coordinate values is given.
- We have gameEnded method whose parameter is empty and return type is boolean. This method checks if the pipes are connected correctly and if the game is over. It checks the proper connection of the pipes before and after the pipe that we look at the pipe by evaluating it is doing.This method have a currentCell array which type is CellData and it stores current cell to get type and property.Then by the way currrentCell value is taken from board by giving positions as indexes.

    1. If Starter pipe is Vertical then next pipe will be under first. So x1 will be same as x0 and y1 will be y0 + 1.

    2. If Starter pipe is Horizontal then next pipe will be at left side of first. So y1 will be same as y0 and x1 will be x0 – 1

    3. If CurrentCell is the curved pipe, the required unit decreases and increments are made in the coordinate plane by looking at the previous and next tiles.

    4. If currentCell is a horizontal pipe and entered from left-side we have to assign x1 to column of pipe before then increment column of current pipe.If currentCell is a horizontal pipe and entered from right-side we have to assign x1 to column of pipe before then decrement column of current pipe.

    5. If currentCell is a vertical pipe and entered from top we have to assign y1 to column of pipe before then increment row of current pipe.If currentCell is a vertical pipe and entered from top we have to assign y1 to column of pipe before then decrement row of current pipe.

    6. If current pipe is End and entered from left and End pipe is horizontal then all pipes are connected.If current pipe is End and entered from down and End pipe is vertical then all pipes are connected.If End pipe is connected any another way then specified above it is a faulty connection so result is false.

7. If empty tile or empty free tile comes rather than conditions specified above then connection is faulty

3) Implementing a Main class with the following UML diagram.

Main	
-	game: Game
-	level: int
-	numberOfMoves: int
-	gameDatas: String[]
-	boardGrid: Rectangle[][]
-	boardPane: GridPane
-	gameStatusLabel: Label
-	moveCounter: Label
-	ball: Circle
-	welcomeScene: Scene
-	gameScene: Scene
+	start(window: Stage): void
-	nextLevel(): void
-	generateGridPane(): void
-	setDraggable(r: Rectangle): void
-	setSlot(r: Rectangle): void
-	exchangeRectangles(x0: int, y0: int, x1: int, y1: int): void
-	saveGame(): void
-	loadGame(): void
-	playAnimation(): PathTransition

- game which type is Game, is created to use the game mechanics of backend. level variable which type is int, represents the number of levels we are in. numberOfMoves variable which type is int calculates user's moving. gameDatas array which is in type of String, stores game level files. boardGrid array which is in type of Rectangle, stores rectangles respectively. boardPane which type is GridPane, is a layout that provides Rectangle 4 to stop 4. gameStatusLabel which type is Label, is an article that whether the game is over and which level is shown in it. moveCounter variable which type is Label shows our move number. Ball which type is Circle, is a ball that will move when the animation starts. welcomeScene which type is Scene, is scene of the main menu. gameScene which type is Scene, is scene of the game.
- We have standard start method whose parameters is window type Stage and return type is void and accessibility is public. In this method, we create an instance of our class Game to start backend of our game, create Rectangle array which we later use to show our game elements in gui and define gameDatas string array here which we later use when generating gameboard. We load unlocked level from save file with loadGame method. We construct GridPane layout which will later store our

Rectangles in 4x4 row and column structure. We set padding for this our boardPane (GridPane object), set gap between Rectangles to 0, and define our rectangles, set constraints and add them to boardPane.

We place ball to Center of Starter element as it is position is not changing. We placed a nextLevelButton (Button object) to check if level is finished, define how nextLevelButton behave when clicked, use gameEnded method of Game class object game to find if game is ended which means all the pipes are connected correctly and we are played animation and store it inside to determine the behavior after it is finished by placing a event handler. When animation is finished we set move number to 0. if last level is done we set level to first level.

This method have a mainMenuButton object which is an object of button. We define the behavior when button is clicked by changing scene to welcomeScene and when the level is passing successfully update level number with text. Graphics for game we place our GUI elements nextLevelButton, gameStatusLabel, moveCounter, mainMenuButton inside gameStatusLayout and place our boardPane (GridPane object) and ball inside gameView layout. We create our scene that will show our game by constructing it with gameLayout and create a title for main menu.

We create a VBox object which will store levels and create a buttons for levels. We designed separate buttons for each level, we set the number of moves and level indicator text of each level. After that we add all buttons to levelsBox which is VBox's object and create the main layout. We set title to Top section of welcomePane and set levelsBox to Center section of welcomePane.Finally, we set the scene to welcomeScene when the application starts and set the title of window as "PipePuzzle". And Show window.

- We have nextLevel method whose parameters is empty and return type is void and accessibility is private. This method unlocks next level and generates gameboard 4x4 CellData array and 4x4 Rectangle array. With this method, we unlock the next level, by increasing the "unlock level" value one by one. Then, we generate backend which CellData array with size as 4x4 with parameter gameDatas which stores input level file names.And at the end, we generate frontend which is Rectangle array with size as 4x4.
- We have generateGridPane method whose parameters is empty and return type is void and accessibility is private. This method generates Rectangle array(boardGrid) which size is 4x4 in order to fill rectangles with images to show in frontend. Method allows to fill rectangles with matching images of boardGrid elements. We created an image object which name is temp to provide the necessary conditions then according to type maching image and constructing Image object by URI path using File object.
    1. If the type is Pipe and property is Vertical then matching image "Pipe_Vertical.png" and construct Image object by URI path using File object.Then, if it is moveable set it draggable with defined setDraggable method which makes Rectangle moveable(draggable).
    2. If the type is Pipe and property is Horizontal then matching image "Pipe_Horizontal.png" and construct Image object by URI path using File object. Then, if it is moveable set it draggable with defined setDraggable method.
    3. If the type is Pipe and property is 00 then matching image "Pipe_00.png" and construct Image object by URI path using File object. Then, if it is moveable set it draggable with defined setDraggable method.
    4. If the type is Pipe and property is 01 then matching image "Pipe_01.png" and construct Image object by URI path using File object. Then, if it is moveable set it draggable with defined setDraggable method.

5. If the type is Pipe and property is 10 then matching image "Pipe_10.png" and construct Image object by URI path using File object. Then, if it is moveable set it draggable with defined setDraggable method.
6. If the type is Pipe and property is 11 then matching image "Pipe_11.png" and construct Image object by URI path using File object. Then, if it is moveable set it draggable with defined setDraggable method.
7. If the type is Empty and property is Free then matching image "Empty_Free.png" and construct Image object by URI path using File object. Empty Free is a element that we can drag other elements into so we set it as a slot with our setSlot method which makes Rectangle a slot which is a place other draggable elements can be moved into.
8. If the type is Empty and property is none then matching image "Empty.png" and construct Image object by URI path using File object. Then, if it is moveable set it draggable with defined setDraggable method.
9. If the type is PipeStatic and property is Horizontal then matching image "PipeStatic_Horizontal.png" and construct Image object by URI path using File object.
10. If the type is PipeStatic and property is 00 then matching image "PipeStatic_00.png" and construct Image object by URI path using File object.
11. If the type is PipeStatic and property is 01 then matching image "PipeStatic_01.png" and construct Image object by URI path using File object.
12. If the type is PipeStatic and property is 10 then matching image "PipeStatic_10.png" and construct Image object by URI path using File object.
13. If the type is PipeStatic and property is 11 then matching image "PipeStatic_11.png" and construct Image object by URI path using File object.
14. If the type is PipeStatic and property is Vertical then matching image "PipeStatic_Vertical.png" and construct Image object by URI path using File object.
15. If the type is Starter and property is Horizontal then matching image "Starter_Horizontal.png" and construct Image object by URI path using File object.
16. If the type is Starter and property is Vertical then matching image "Starter_Vertical.png" and construct Image object by URI path using File object.
17. If the type is End and property is Horizontal then matching image "End_Horizontal.png" and construct Image object by URI path using File object.
18. If the type is End and property is Vertical then matching image "End_Vertical.png" and construct Image object by URI path using File object.
19. Finally, controlling no image and no info property that If the type is anything other than specified above board cannot be generated as there is no resource for that parts and printing “Board can not be generated” message with exiting system.

- We have setDraggable method whose parameters is r type Rectangle and return type is void and accessibility is private.This method makes Rectangle moveable(draggable).Firstly in this method, detecting drag property is controlled then, source of event is gotton by "getSource" method which use to determine which tile is selected of event and stored the Rectangle inside temp.After this temp variable’s x and y coordinates(indexes) are found and stored by using Point2D which is an object property, we initiate and end dragging event by follow this path.
- We have setSlot method whose parameters is r type Rectangle and return type is void and accessibility is private. This method determines the target tile and enables the movable tiles to be replaced by finding the coordinate of the target tile. We check if the move is legal (applicable) by passing coordinates to method of Game class object game. If move is legal then we make move

(swap elements) by passing coordinates to method of Game class object game.Furthermore, increasing numberOfMoves variable which is created to calculate user's moving, then, updating the label to show our move number with moveCounter value.

- We have exchangeRectangles method whose parameters are x0,y0,x1,y1 types int and return type is void and accessibility is private.Briefly, this method provides us to change rectangles.Firstly, it removes Rectangles that will be exchanged from boardPane (object of GridPane), then, it swaps their coordinates and add to boardPane.Finally, it swap Rectangles inside the boardGrid too to make change successful.
- We have saveGame method whose parameters is empty and return type is void and accessibility is private.We created this method to save cases of levels(unlocked level).Therefore, we save unlocked level to "save.dat" file.While it is doing, an object of file "save.dat" is created and to write in save.dat file, Formatter class is used.Then, the file is closed and in case of any error happens, an error message is printed.
- We have loadGame method whose parameters is empty and return type is void and accessibility is private. We created this method to load cases of levels(unlocked level). Therefore, we save unlocked level to "save.dat" file.While it is doing, an object of file "save.dat" is created and to read save.dat file, BufferedReader class is used.For this this method firstly, reads one character which is level number but string type and it can be cast it to integer it's string value, we used 0 value which is ineffective element and we defined this operation as integer.
- We have playAnimation method whose parameters is empty and return type is PathTransition and accessibility is private. This method is used to play animation which is ball rolling inside pipe to end, after level is finished. We create a Path object which we later add directions for animation translation. In this game project we designed 2 different animation because animation shape and procedure of level 1,2,3 are the same and fourth,fifth level's animation shapes and procedures are the same but these two are different each other.so we defined them with different ways by using required condition structures.We create a PathTransition object called animation. This will be used to define our animation.We set our path to Path object path, apply this animation on ball (Circle object), set duration to 3 seconds and set cycle count of animation to 1 to play animation once. We return animation to set our level to advance when animation is finished using setOnFinished event handler.

QUESTIONS

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

    - In our project, all the desired sections are completed and working smoothly.

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

    - During the implementation, we experienced the biggest difficulty in dragging tiles with the mouse. The solution to this problem has taken much of our time. Another problem we experienced was that after the levels were completed, the ball did not return to the starting point and the previous level remained in the finishing tile. The other problem was the animation. It took us a long time to set the axes in which the ball slid.

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

- The boxes that were assigned were intended for preview, so they were not properly cut and adjusted, and when we used these boxes, we found that pipes could not be combined properly.Therefore, we prefer to use the boxes we designed from scratch. We designed all boxes so that they can be seamlessly connected to each other, taking into account the specifications specified.

- We created a menu for our game and this menu contains various decorations.Firstly, we printed “Welcome to PipePuzzle Game” sentence on top of it.In this way, we mentioned the name of our game in our menu.Secondly, we added 5 different buttons which represent game’s level numbers to this menu and we designed these buttons by adhering to the desired specifications.

TEST CASES

This part consist of some screenshoots and explanation about of these.

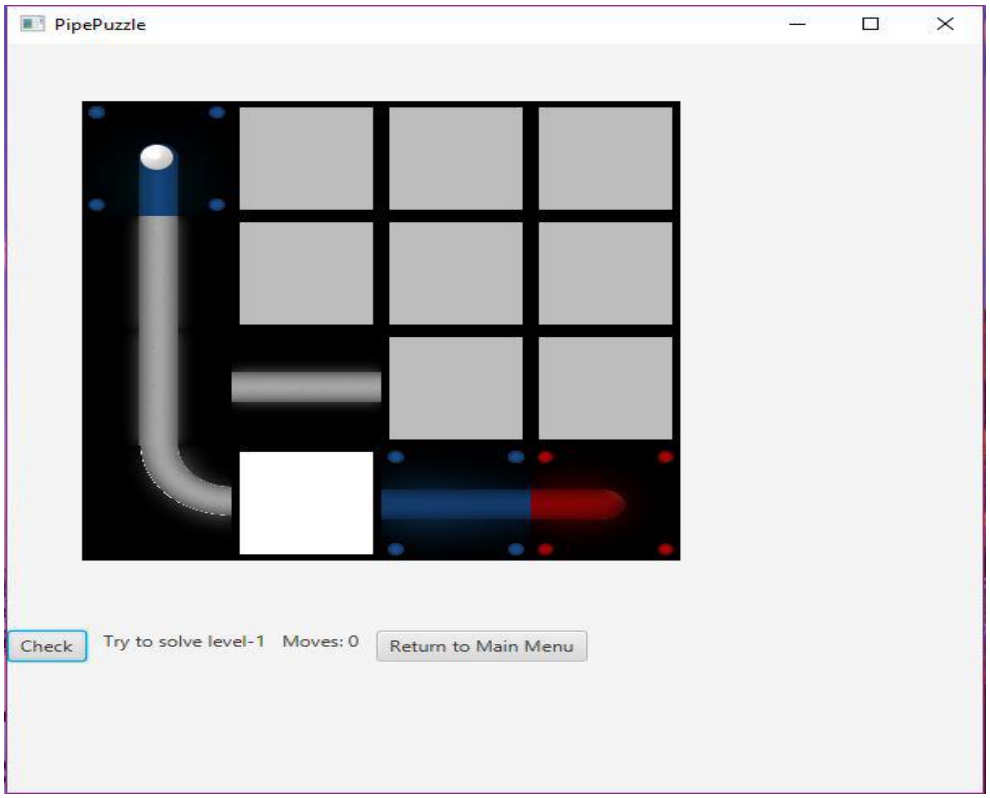
1)Main Menu

This image was taken from the main menu. There is a main menü and this menu has a welcome message and 5 buttons to launch different levels. However you can not skip to another level before without completing to previous one

2) Level 1

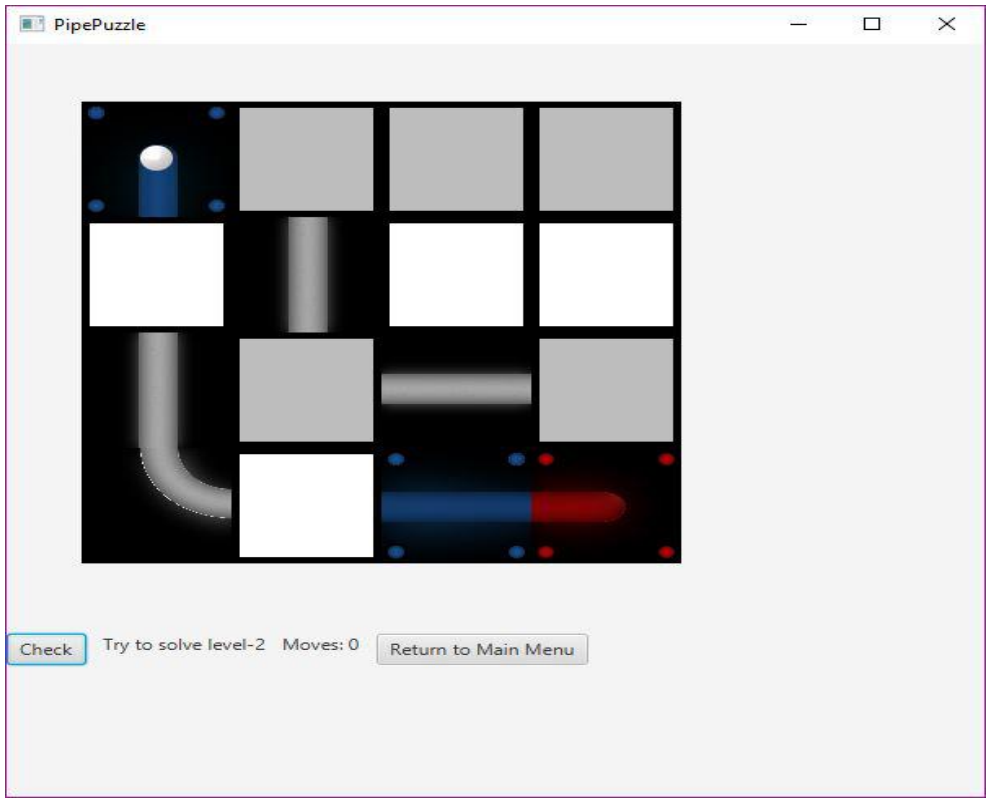
This image was taken from the first level. The initial image of the first level with the shapes we have redesigned.

Image of the first level completed. As seen in this photo the check button is located in the lower left corner and the return to the main menu is located in the lower right corner. The level at which we are and the move counter is located at the bottom of the game screen.



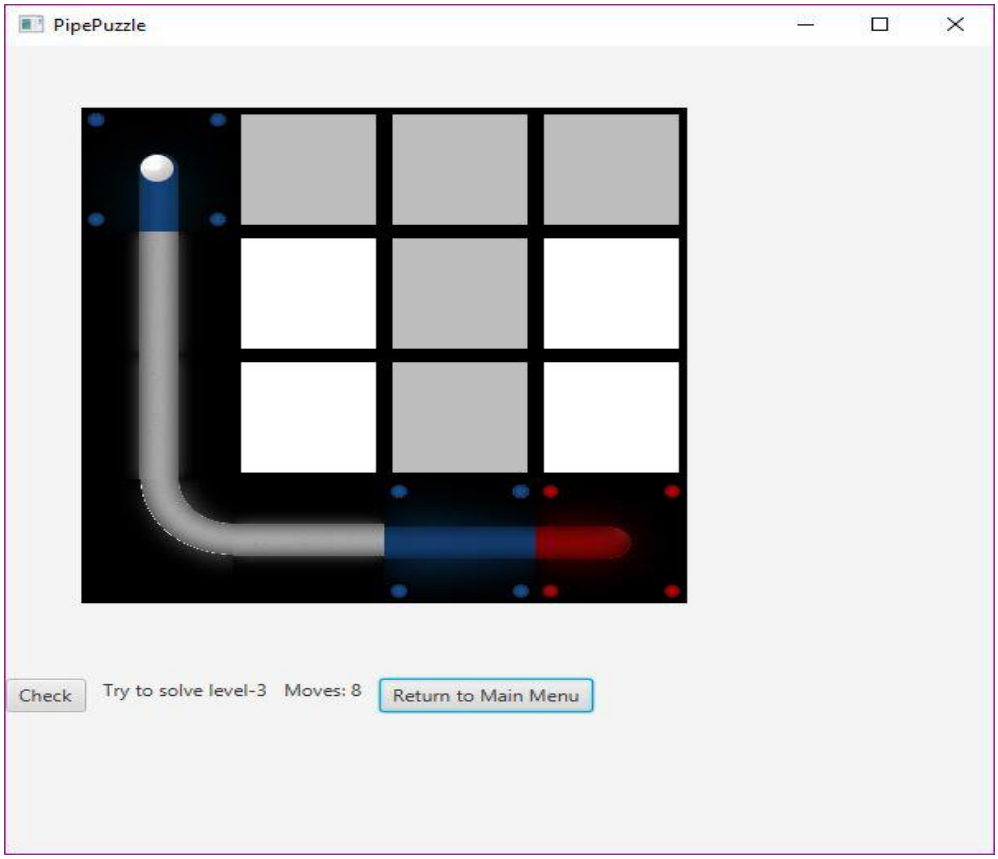
3) Level 2

This image was taken from the second level. While the tiles with pipes can not transported to empty tiles, the tiles with pipes and empty tiles can be transported to empty free tiles.



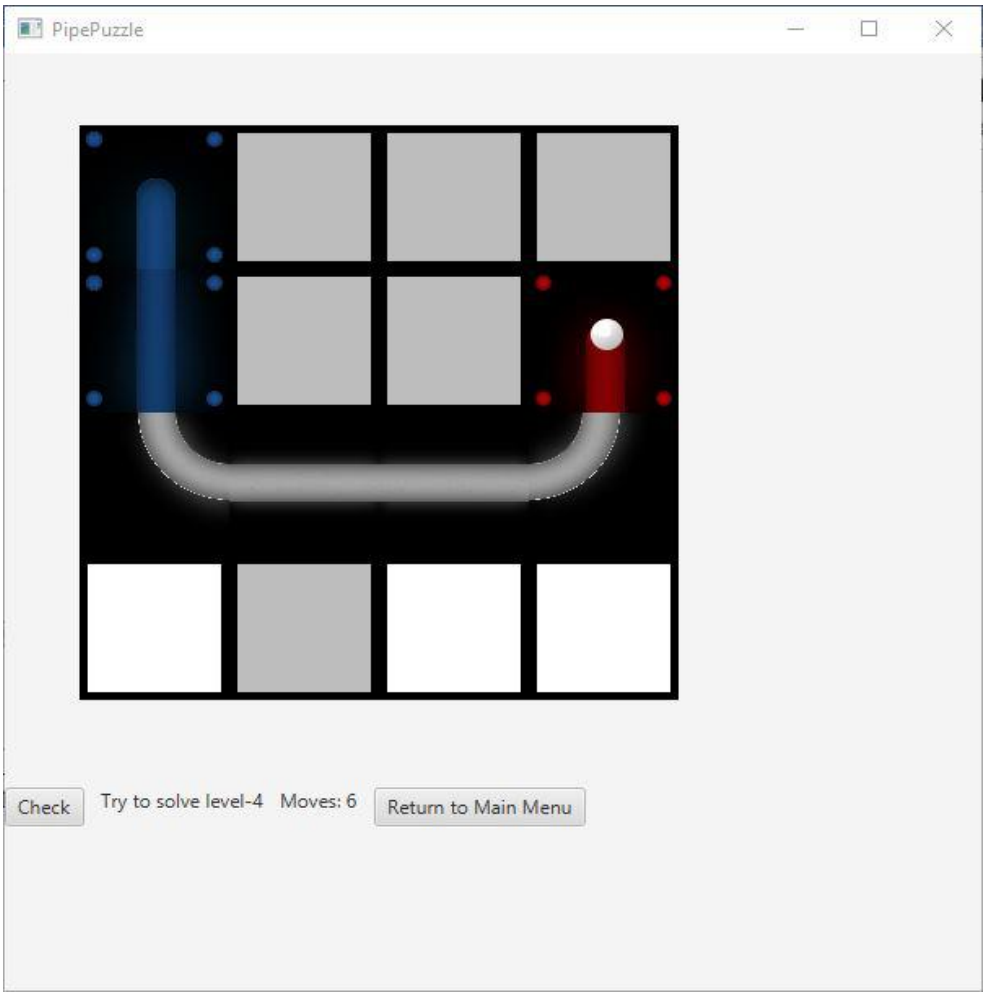
4) Level 3

This image was taken from the third level. As shown in the picture, this level of the game has been successfully completed in 8 steps.



5) Level 4

This image was taken from the fourth level. The user has completed this level in 6 steps After the level has been completed successfully, the ball's image in the endpipe.



6) Level 5

This image was taken from the fifth level which is the most hard level of our game.In this level, moves value is grater than the other levels because connecting pipes is more complex than other.After the user has been completed succesfully this level and the animation has been started ve took this shot.