

CSE1142 : Computer Programming II, Spring 2019

Date Submitted : May 12,2019

Tile Game Project

150116010 Şükriye Soyer

150118002 Gizem Çelik

***Problem Definition***

In this game, there is a board filled with different tiles. Tiles, may be a pipe, empty , starter or end tile. Also there may be have different different locations of tiles. For instance, tiles have properties with it’s types and locations and images too. Here there is tiles that we used to create our board: (These images is for introducing the game that we create.)

(Starter horizontal tile.) (Starter vertical tile.) (Pipe vertical tile.) (Pipe horizontal tile)

Here is curved pipes.

(Pipe 00 tile.) (Pipe 01 tile.) (Pipe 11 tile.) (Pipe 10 tile.)

endHorizontal endVertical emptyFree emptyNone

(End horizontal tile) (End vertical tile) (Empty free Tile) (Empty none Tile)

pipestaticHorizontal pipestaticVertical 

(Pipe static horizontal) (Pipe static vertical) (Pipe static 01 tile.)

We used this images for create the game board.

In this game our aim is placing pipes correct position for move the ball from the start tile to end tile and user must use the mouse for drag them to correct positions.If user places tiles correctly who can move the next level, if user desires. There is some rules in our tile game ;

* Starter ,end and static tiles cant move.
* Tiles cant move diagonally.
* Gray tiles represents free spaces (free space means, where any moveable tile move ).
* The shape of tiles cant change.

The moveable tile cant move more than one tile.

***Implementation Details***

To design our tile game, we created a UML diagram for creating tile class.

|  |
| --- |
| Tile |
| -type: String -loc: String -index : Int -i :ImageView |
| + Tile () +Tile(type: String,loc: String,index : Int) +isMoving(type: String): boolean +images(type: String,loc: String ,x : int,y :int):void +getter/setter methods |

|  |
| --- |
| TileGame |
| +sce : Scene +countt :int +pa : Pane +scen : Scene +tileuse :Tile + win : boolean + win1: boolean + level :int + count :int +int: moves |
| + main(String[] args):void +start(primaryStage):void + fileMaker(f, count):String + Tile(x:int, y: int, p:Tile[][]):Tile +won1 (p:Tile[][] ) :boolean +won2 (p: Tile[][]) :boolean +move(p1:Tile, p2:Tile) :boolean +swap(Tile p1, Tile p2,Tile[][] arr): Tile[][] |

We created a tile class for keep information of every tiles given in level texts.

Firstly we prompt the user a file then we read the file with using FileMaker. FileMaker method does remove the ‘,’ from the input file, and it writes non ‘,’ line to new created file. It returns string named “newFile...txt” like this. When it returns to its caller, this text file will be readed. Then we used input.next for give tile properties from this file. Calling the images method we set tiles image and sets it coordinate. Then we add this tiles to array list and 2 dimensional array. Then we added content of 2d array to our pane. We created the scene that contain pane and its width and height property.

After, we created tile game’s start page. We created labels, texts, buttons and image with specific locations as we set. Then we add this objects to new created pane. We created the scene that contain pane and its width and height property. Then we set stage to start scene.

When user presses start button, program sets the scene’s page as level1’s pane with setOnAction. We used lambda expressions for all., This is important logic of the start page. This is where game actually starts.

When we come to level1. We created setOnMouseDragged action event for drag tiles also we used lambda expression, we write our many codes in there. User will be drag pipes correctly for move to next level. When user presses the any location the tile method will called. Tile method is which computes the tile in mouse dragged with mouse's x and y coordinates. This method takes the mouses x and y coordinates and 2d pipe array which takes all tiles in it. In method it checks coordinates, if mouse coordinates is in the the given interval, then it will return tile from the 2d array around this coordinates. This is what Tile method does.

After, when user releases the object and we used setOnMouseReleased expression with lambda expression. With that, we get the mouses coordinates then we invoke the method named Tile again to get the pipe around this coordinates. Then we add this object to pressed arraylist to index1.

And there is method named like is Moving in our Tile class. This method returns false if object is not removable. If object is movable then method will return true. We checked 2 object with this method.

There is a swap method in our Test class. It will be active if objects are movable and first tile is not empty(cause empty free tiles is a board itself) and it controls where the second tile is with its index.

We created method named as move for is second tile is near to first pressed tile. In this method it controls 2 tile's indexes. First index is first tile(dragged), second index is second tile(released). If it is second tile is in right side upper side down side or left side of first tile, it checks this conditions for if it can move. In this method it controls 2 tile's indexes. For example if first tile index is 1, and second(released) tile has index 2, then it means they are swapable. If it is like: first tile has index 1 and second tile has index 6. It does not swap them. Because they are diagonal. If they can swap, it will return true. Else, it will return false. When given conditions are true, it will invoke this method.

Finally there are 2 different win methods for game’s win condition. However our win algorithm is not dynamic. It just works for given level conditions. First win method takes 2d array then it checks the location of tiles. If they have same places as written in our won method. It will return value of true to its caller.

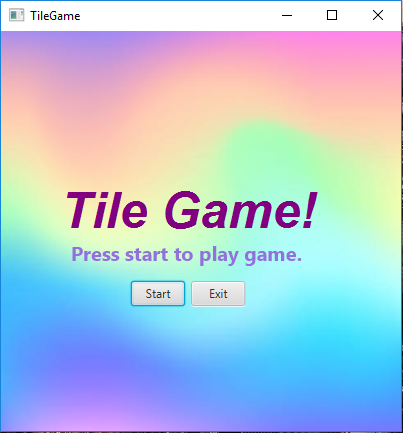
If user completes the level, ‘next’ button will be displayed. We used setOnAction for next button with lambda expression. If user presses this button, next level will be appeared to screen if user desires. If user can not press ‘next’ and move tile to false location. Next button will be disappeared from the screen. When user presses ‘next’ button, program will start looking for next levels text doc. If next levels file exist in users computer, It will continue to execute the code by reading next level file. It will clear the existing pane and place new levels tile to it. There are same steps as we wrote in start of implemention part. It will be continue like this if file exist and user wins. If there is not next level’s text doc, it will display a screen which writes “You completed all levels! Congrats!”. Our implementation is like this.

We completed many parts of the game. We can not complete the balls animation and win condition. For win condition, we completed it for just 5 levels. We cant designed win condition dynamically. Cause this was very hard part to us. For dynamic properties: Our code can read infinite level texts if users computer has next levels text files. Our code can display anly levels board. Finally our drag code works on any level board.

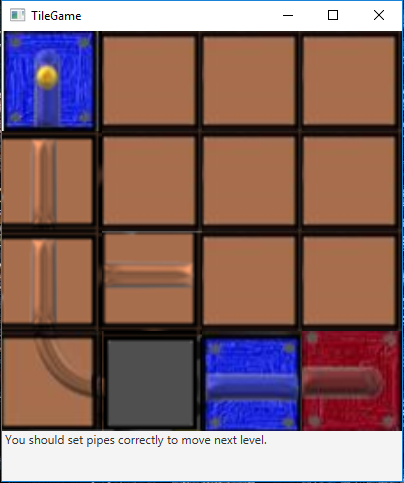
Our Project has many difficulties in it. We had difficulties when we drag tile to another location. It was hard to us because, first we did drag step with click event. Secondly we had difficulties in moving ball from start to end tile. We can’t implemented this feature in our Tile game. And lastly, this is where we really had difficulty, win part. To make his part dynamically was really hard for us. We we can’t implemented this part dynamically but it Works for first 5 level in our game.

There are many additions in our game. First addition is our start scene. In start scene there is ‘Tile game’ label and ‘press start to play game’ text. Also this pane has start and exit button too. When user presses ‘start’ button, first level of the game will be displayed to screen. Exit button is for exit the game. And we have ‘next’ button for any level but it can display when user wins the level.

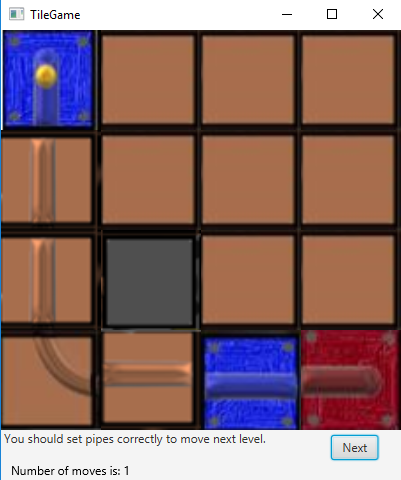
**Here is test cases our Tile Game:**



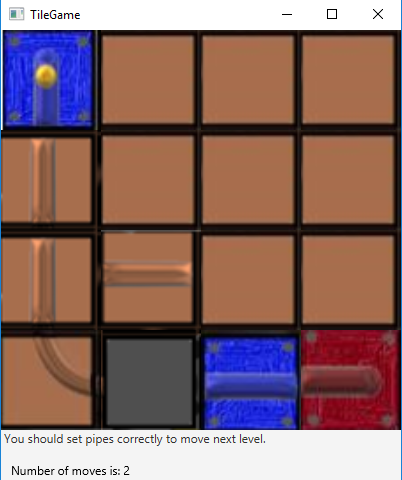
This is the start screen of the our Tile Game.



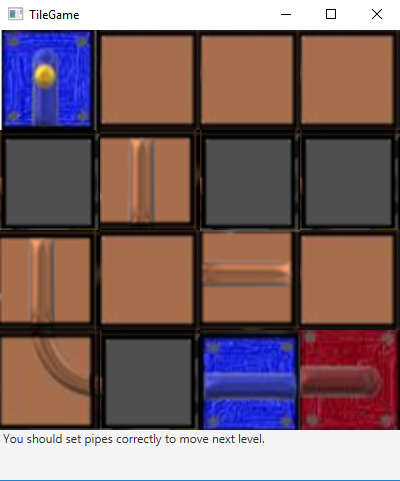
When user presses ‘Start’ button of start screen, first level will be appear in our Tile game. There is rule text in it.



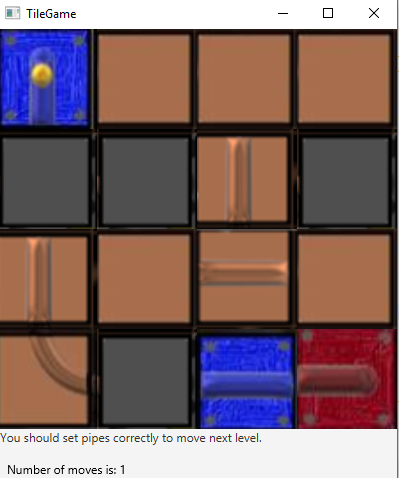
This picture represents finished status of level1. When user finishes first level, ‘next’ button will be appear in screen.



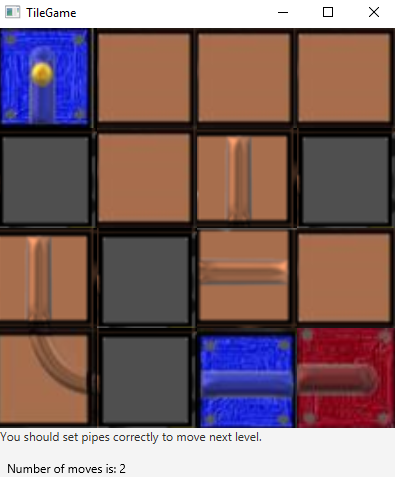
If user changes location of correctly placed tiles, ‘Next button will be removed from the screen. User can not play the next level without completing the path.



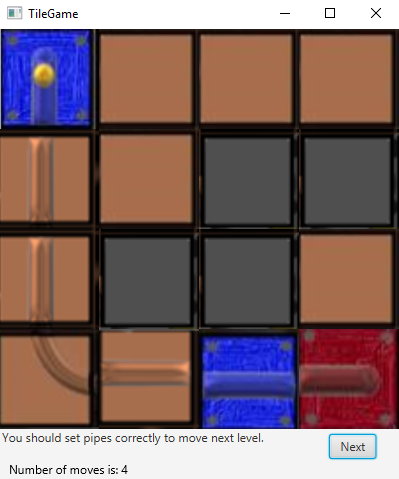
When user presses ‘next’ button of completed status of level1, second level will be appear in our Tile game.



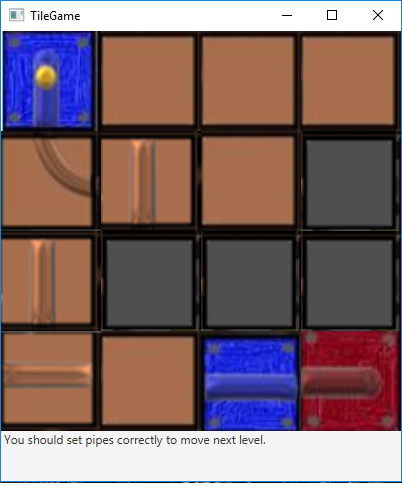
Here is level 2 with 1 move statement.



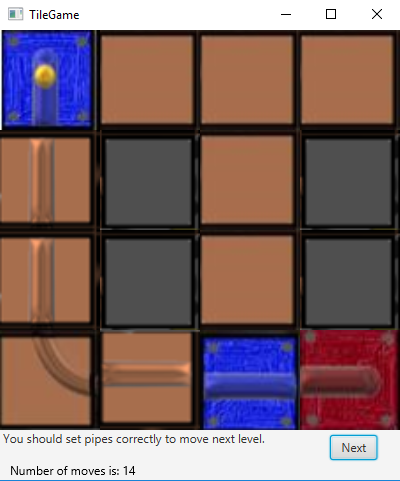
Here is level 2 with 2 move statement.



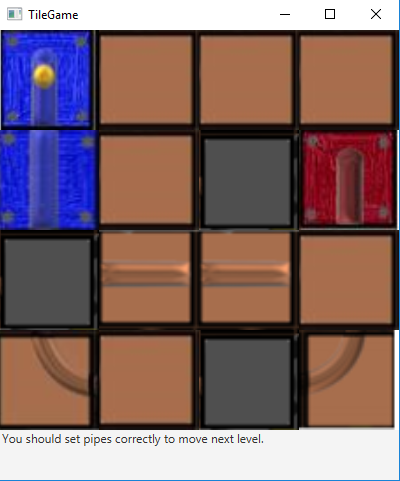
This picture represents finished status of level2. When user finishes second level, ‘next’ button will be appear in screen.



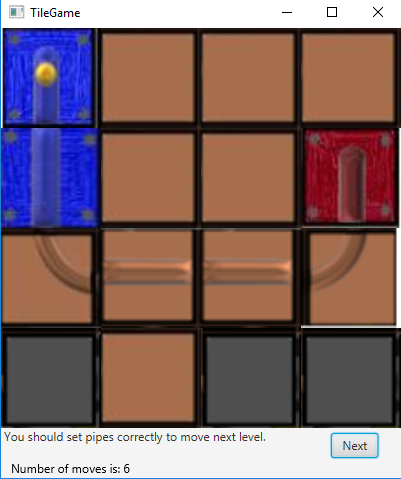
When user presses ‘next’ button of completed status of level 2, third level will be appear in our Tile game.



This picture represents finished status of level3. When user finishes third level, ‘next’ button will be appear in screen.



When user presses ‘next’ button of completed status of level 3 fourth level will be appear in our Tile game.



This picture represents finished status of level 4. When user finishes fourth level, ‘next’ button will be appear in screen.



When user presses ‘next’ button of completed status of level 4, fifth level will be appear in our Tile game.



This picture represents finished status of level 5. When user finishes fifth level, ‘next’ button will be appear in screen if level 6 exists.



If level 6 does not exists, then this message will be displayed to screen.

This is our Tile Game test cases.