MARMARA UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

CSE1142: Computer Programming II, Spring 2019

Mushroom Meshwork

NAME	STUDENT NUMBER
Bilgehan GEÇİCİ	150117072
Anıl ŞENAY	150117023

Submitted to: SANEM ARSLAN YILMAZ

Date Submitted: May 12, 2019

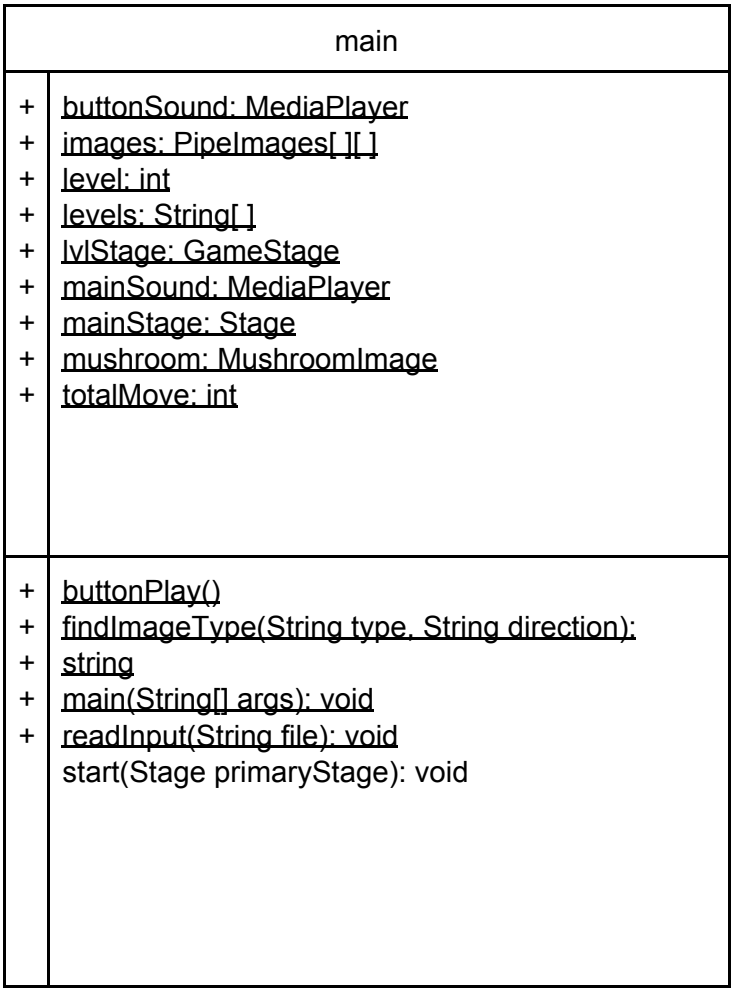
1-) Problem Definition

The mushroom game aims that, user have to carry forward the mushroom through the pipes if there is an appropriate path in order to get the mushroom to the red background pipe. User is supposed to fix the pipes to the acceptable places in order to create appropriate path. To create appropriate path, the user changes the pipe's locations in correct way to help the mushroom to pass through the pipes clearly.

The user only moves the pipes vertically and horizontally so the pipes cannot move diagonally and each pipes or empty areas can move 1 square distance (in game board there are 16 squares and each square have same side length). The blue background pipe and red background pipe cannot move and if the user have not completed the current level, then the user would not have moved on with the next level.

2-) Implementation Details

Uml diagram of main class



- **buttonPlay()** method starts and stops the menu navigate music when user clicked main menu buttons.
- **findImageType(String type, String direction)** methods takes two parameter and it returns the exact url of the images in type of string based on given text file.
- **main(String[] args)** method contains launch method and it starts the game.
- **readInput(String file)** method reads the types of the images from the given text file and then it creates images based on the given text file and finally adds the exact position of the images to images array
- **start(Stage primaryStage)** method creates a background for main menu to the game and it places some buttons like start button, settings button, and credits button and then it adds a cup image right side of bottom for leader board section. Also it adds some sound effects and when the user interacts with the main menu it plays the given music and effects.

Uml diagram of CreditsStage class

CreditsStage	
+ + +	animation: Timeline creditsScene: Scene credits_text: ImageView
- -	CreditsStage() finished(): void

- **CreditStage()** method creates a credits scene for the end of game and it slides the credits text from bottom to up.
- **finished()** method terminates the game when the user completed all the levels then it resets the game.

Uml diagram of Endgame class

EndGame	
+	**EndGame()** **writeLeaderBoard(String nick, int score): void**

- **EndGame()** method prints a screen at the end of the game which contains user's total move and name input section and it takes user's name for the leader board section.

- **writeLeaderBoard(String nick, int score)** method takes the names and then it writes the names to the leaderboard section.

Uml diagram of MushroomImage Class

MushroomImage	
+	MushroomImage(Image image, double x , double y): void

- **MushroomImage(Image image, double x, double y)** method takes 3 parameter and it adds a mushroom image to the game board and it arranges it's x, y, height and width properties.

Uml diagram of ImagePane class

ImagePane	
+	**ImagePane()** **print(): void**

- **print()** method prints the image array to the game board and if it encounters 'starter' image, then it adds the mushroom image to the starter image.

Uml diagram of GameStage class

PauseStage	
	PauseStage()

- **PauseStage** method creates a pause menu to the game and it can be accessible with Esc key. It is also be possible to access main menu or settings menu from pause menu

Uml diagram of NextLevelStage class

NextLevelStage	
+	winSound: MediaPlayer x: double y: double
+ +	NextLevelStage() playMusic: void

- **NextLevelStage()** method creates a informative screen which is related with next level. When the user completes current level then the screen shows up and if user wants to continue with game then the user can click the Next Level button.
- **playMusic** method plays the winSound which is property of NextLevelStage class when the user completes the current level.

Uml diagram of SettingsStage class

SettingsStage	
+	**SettingsStage(): void**

- **SettingsStage()** method creates a settings screen to the game and it contains sounds settings. The user can arrange either music or sound effect volume.

Uml diagram of LeaderBoard class

LeaderBoard	
	nicksText: String scoresText: String
- -	LeaderBoard() sortLeaderBoard(): void

- **Leaderboard()** constructor creates a leader board section to the game. It gets the total moves of the user at the end of the game and also gets his/her name and then it prints the leader board.
- **sortLeaderBoard()** method keeps the scores in the array list and it sorts the array list from descending order.

Uml diagram of GameStage class

GameStage	
+	firstLevel: ImagePane
+	levelSound: MediaPlayer
+	moveInLevel: int
+	scene: Scene
+	switchEffect: MediaPlayer
+	wrongMove: MediaPlayer
	GameStage()
+	changeTitle(): void
+	startMusic(): void
+	stopMusic(): void

- **GameStage()** constructor creates the game board for the game with readInput method based on given file.
- **changeTitle()** method changes the title of the game when user passes the current level.
- **startMusic()** method starts the music when the game starts. Each level has a different level music.
- **stopMusic()** method stops the current level's music.

Uml diagram of PipeImages class

PipeImages	
+	direction: String
+	final_index_X: int
+	final_index_Y: int
+	initial_index_X: int
+	initial_index_Y: int
+	isLevelFinished: boolean
+	path: Path
+	previousMove: String
+	pt: PathTransition
+	type: String
	PipeImages()
	PipeImages(Image image, String type, String direction)
+	canMove(initial_X: int, initial_Y: int, final_X: int, final_Y: int): boolean
+	checkNext(x: int, y: int): void
+	isFinished(): boolean
+	switchSound(): void
+	whereIsStarter(): PipeImages
+	wrongMove(): void

- **PipeImages(Image image, String type, String direction) constructor** sends the image to its super class ImageView, then it get the initial X and Y properties and divided by 100 to find image's indexes in image array. After that it gets the second X and Y properties of images and it changes the initial and final X and Y values so it swaps the images between them based on user's actions.
- **canMove(initial_X: int, initial_Y: int, final_X: int, final_Y: int)** methods control the images can move or not and if an image can move than it restrict its move only with 1 block distance so the images cannot move diagonally.
- **checkNext(x: int, y: int)** method checks the images which are located in x and y. It will work recusively until find the 'End' pipe. At the beginning, it controls previousMove's type which is property of PipeImages class. Based on

previousMove it determines next move. By pipe's type it updates previousMove value. This method also adds paths with pipe's type and direction

- **isFinished()** method checks the level is finished or not. First it checks starter's type(vertical or horizontal) and then adds its path. After that it declares checkNext method recursively to check the next pipe image. After recursion if isLevelFinished variable is true, return true for this method.

- **whereIsStarter()** method finds the location of the starter image than it adds the path to the starter image where it is located

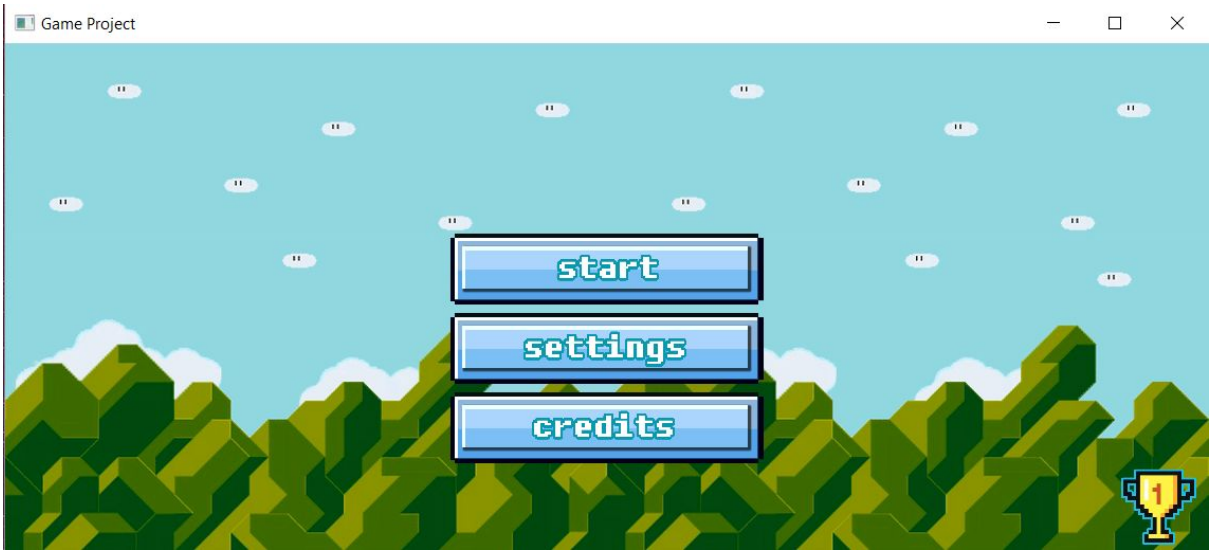
- **switchSound()** method plays the swapping sound effect when the user swaps the images

- **wrongMove()** method plays the wrong move sound effect when the user tries to attempt wrong move.

3-) Test Cases

The following visuals are the result of the project.

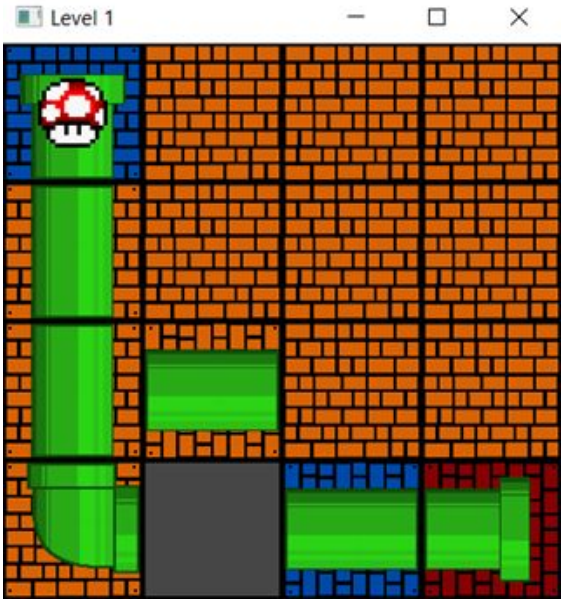
· When the user starts the program on the eclipse the main menu comes up.



(Main menu)

In the main menu there are start game button, settings button, and credits button and also there is a cup photo which shows up the leaderboard section. The three buttons placed at the middle of the background and the cup photo is placed at the bottom-right side of the background.

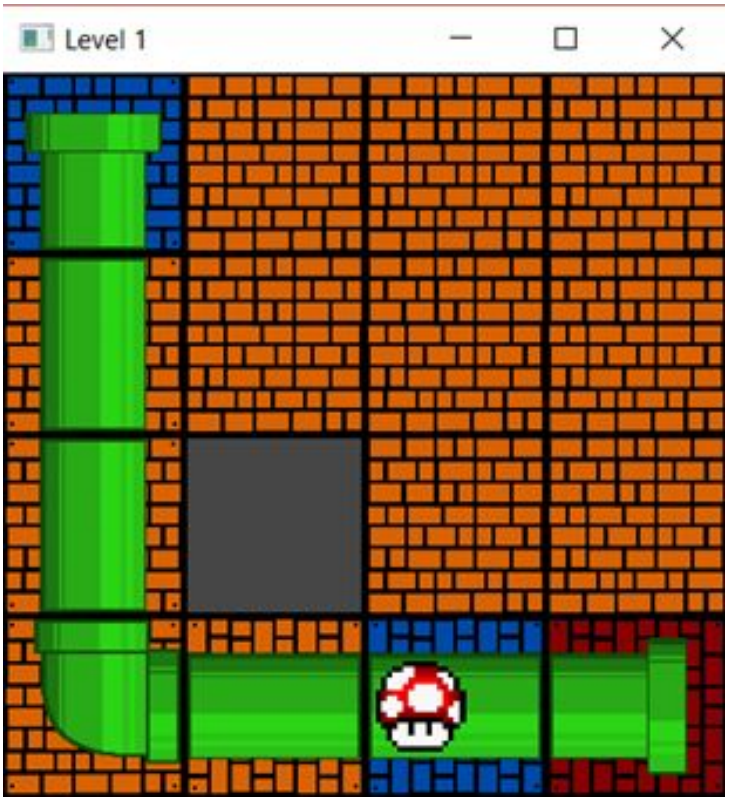
- When the user presses the start button the main menu disappears and the game starts with level 1.



(Level 1)

The game starts with level 1 and the user is supposed to do the appropriate path in order to make a pipe connection to reach the mushroom to the end of the pipe.

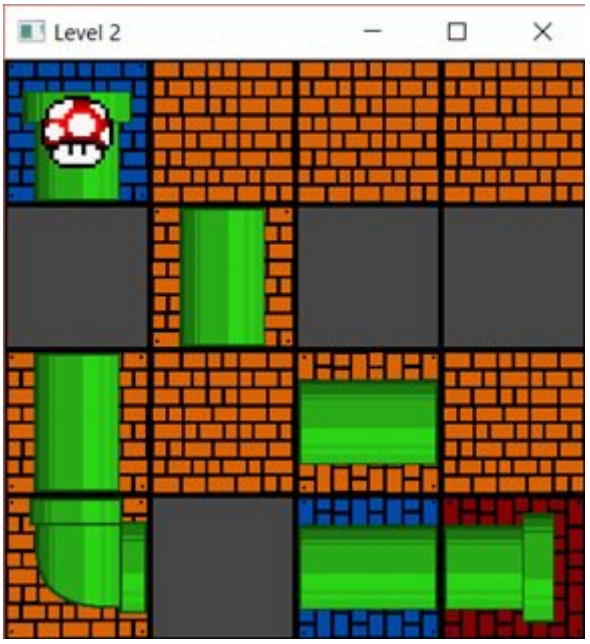
- When the user creates the path, then the animation of the mushroom comes up to the screen.



(Level 1 Animation)
After the any level completed by user, then the small informative menu comes up.

(The informative scene after the level 1)

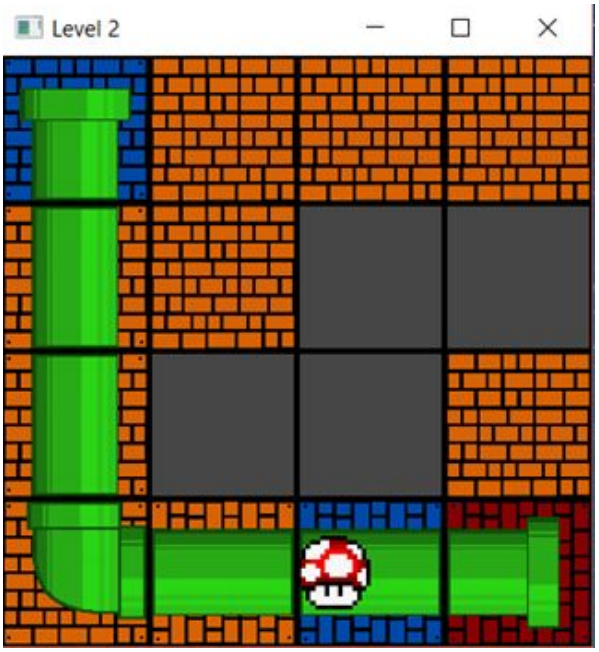
- When the user completes the level 1 the program moves on with small informative scene and asks from the user to pass the next level or go back to the main menu.



(Level 2)

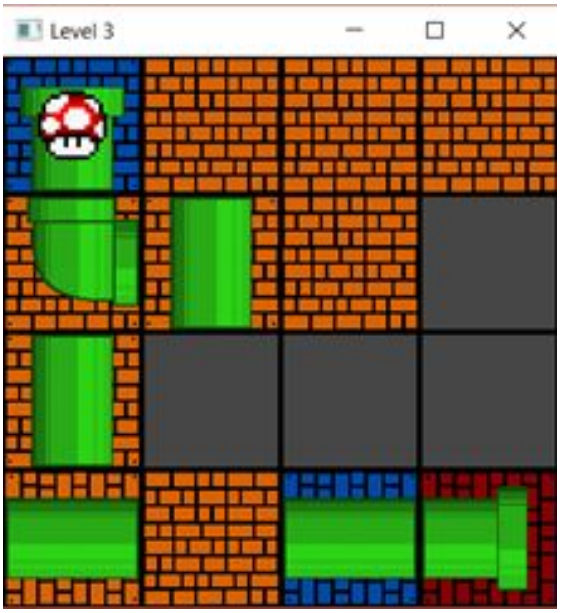
Here it is the same logic as the level 1 when the user creates an appropriate path with changing the places of pipes then the mushrooms moves through the pipes and then level completed.

- After the user creates the path the animation comes up to the screen.



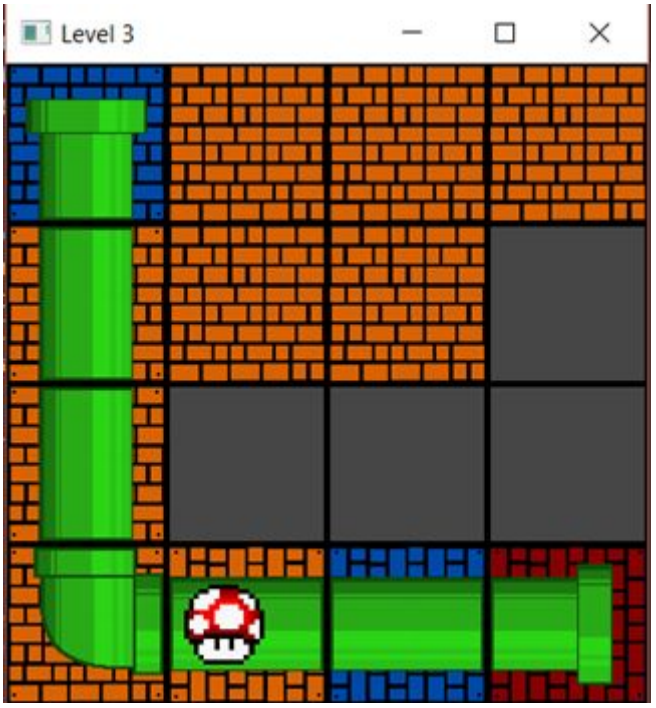
(Level 2 Animation)

- When the user completes the level 2 the program moves on with small informative scene and asks from the user to pass the next level or go back to the main menu.



(Level 3)

Here it is the same logic as the any level when the user creates an appropriate path with changing the places of pipes then the mushrooms moves through the pipes and then level completed.

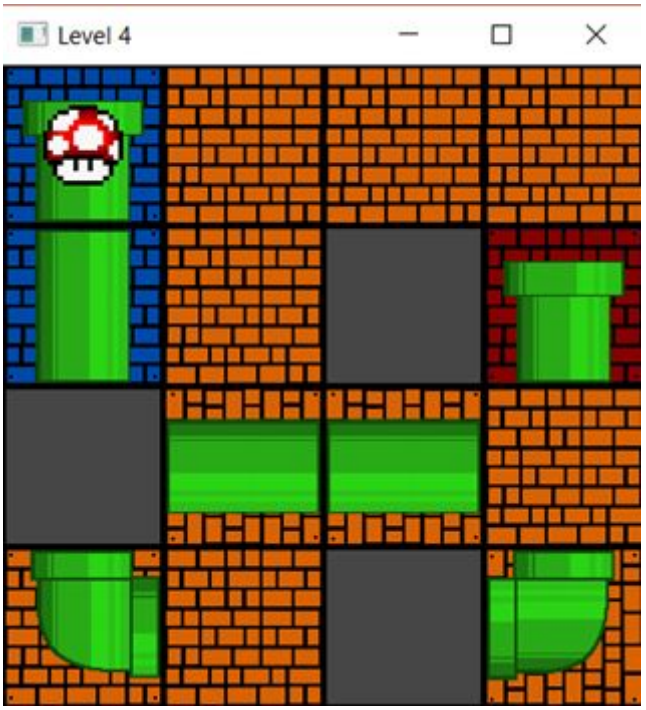


(Level 3 Animation)

- There is a move counter in the game and when the user changes any changeable blocks then the program automatically counts the number of moves in the game here it is sample of the counter.

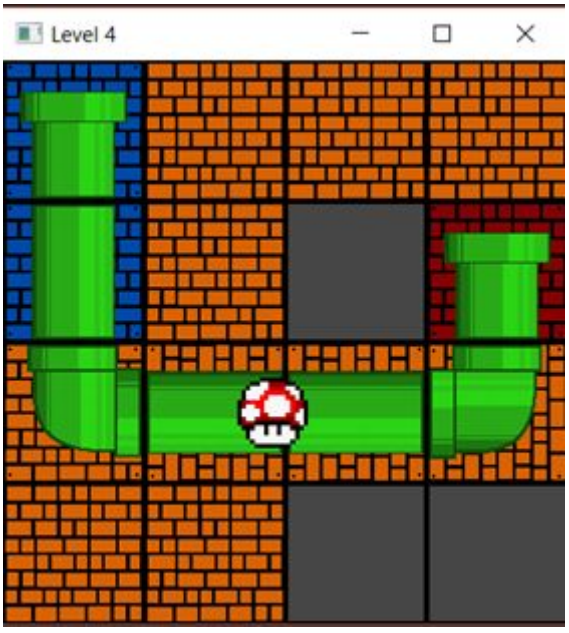
(The counter after the level 3 passed by user)

· When the user completes the level 3 the program moves on with small informative scene and asks from the user to pass the next level or go back to the main menu.



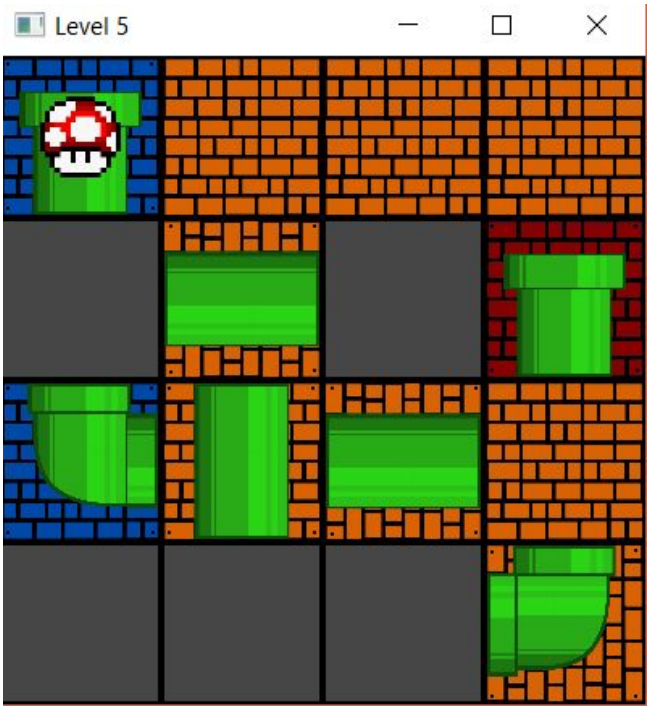
(Level 4)

After the level 4 is done by the user the mushroom starts to slide through the pipes



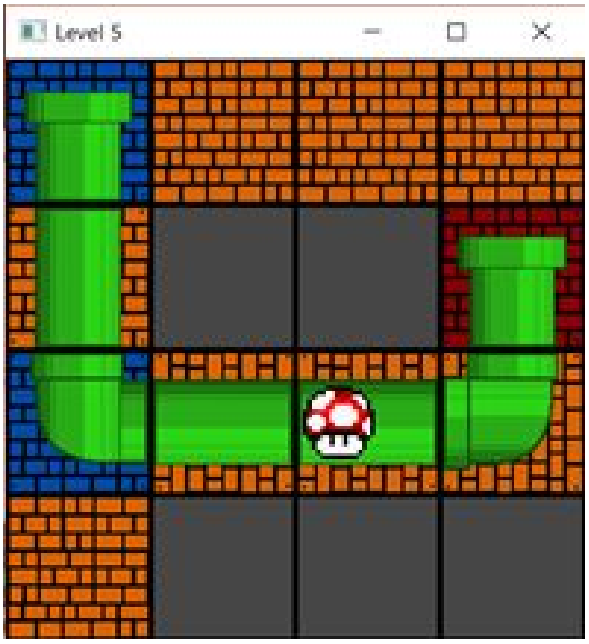
(Level 4 Animation)

- When the user completes the level 5 the program moves on with small informative scene and asks from the user to pass the next level or go back to the main menu.



(Level 5)

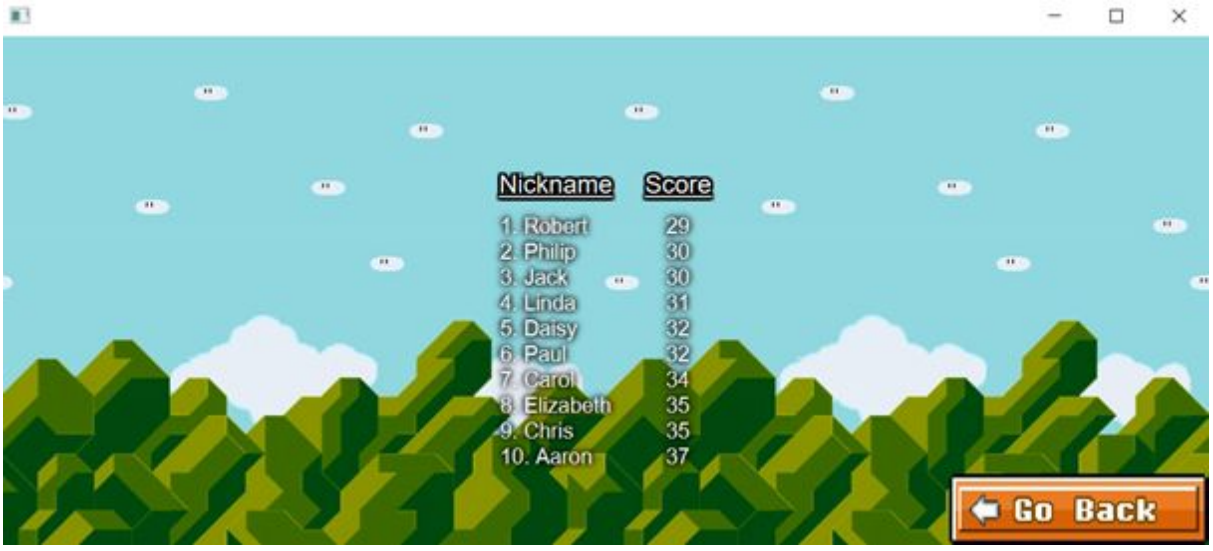
After the level 5 is done by the user the mushroom starts again to slide through the pipes



(Level 5 Animation)

- After the end of game, the program wants a name from the user for the leaderboard. In leaderboard there are some person's name and their number of moves. Their number of moves are already sorted from descending to ascending. Here it's the sample of the leaderboard scene and name input scene.

(Name input scene)

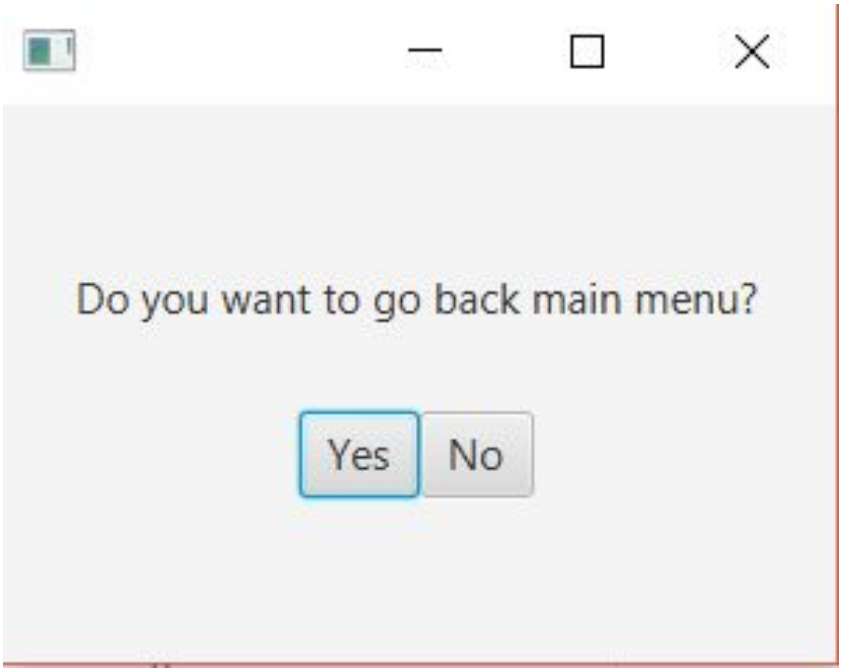


(Leaderboard scene)

- When the user passes all the levels then the game board closes and then a mini extra scene shows up with same background, which is a credit scene. Here is the sample of the credit scene.

(Credit Scene)

- After credit scene done. A small informative scene comes up and it asks from the user what to do after the game.



(Informative scene after the end of the game)

- There is a settings button in the main menu and when the user clicks the button a small scene comes up to the screen.

(Settings Menu)

In the settings menu the user can change the music volume also there is section effects volume in the settings menu and the user can change volume of effects. When the user done with the settings menu the user can press the go back button in order to reach the main menu again. If the user opens the settings menu in any levels and if the user presses the go back button, then the game continues where it is paused.

- There is also credit scene in the main menu the user also can see the credits section from the main menu

4-) Developing and Enhancement Process

Design Part:

In the design part of the game we chose the Mario Theme. First of all, we made the tiles in form of pipe and brick and which are the main design of the Mario game. In the background, again we have inspired from Mario game. We used “Graphics Interchange Format” (GIF) to make the background. We thought that gif is the best way to introduce the main menu and the game. In settings part, credits and the other parts, again we stuck on the Mario theme. We have paid attention to make sure the sections are compatible and not so fancy. We have made sure that all the parts are casual enough.

Sound Effects & Music Part:

In the sound effects part, we have tried to find best sound effects in order to fit the game theme. The sound effects that we chose are swapping sound effect, menu navigate sound effect, level completed sound effect and wrong move sound effect. We also have paid attention to make sure all the sound effects are in harmonious. In the music part we chose the 8-bit music style in our music's. We thought that 8-bit music are compatible with our game. We used Eric Skiff's 'Resistor Anthem' album which has no copyright and also, we informed him about using their albums and music's. Again, we have tried to catch the harmony between music. We arrange our music in the game's difficulty.

Development Part:

Firstly, we started to work for the basic structure of the game. After reading the data from the file and printing them on the screen, we started working for the displacement movement. We experienced a problem in the relocation process. But we realized that the cause of the problem was that we didn't do the physical change in the array, so we fixed the problem. After the movement of the displacement works smoothly, we have written a method to check if the level is finished. This method checks the next pipe recursively from starting point to see whether it reached to the end. As this process checks all the pipes properly, mushroom's path is also drawn at the same time. In other words, when each level is over, it also draws the path. We didn't have to write a separate method to draw the path. Then, when the level was completed, we moved the mushroom at the beginning on the completed path until the end with a path transition. When this path transition is over, a new screen appears to continue with next level.

Before starting the project, we determined what we wanted to do. In this list, we have written everything we add to the game except the basic form of the game. With our own designed pictures (gifs, buttons, pipe images etc.), we have added a new theme to the game. We have added features such as pause, credits, settings, leaderboard screens. Also, in the game creating part we encountered so many bugs. We have tried to fix the bugs so the game is so stable than the old version.

We have completed the game completely and with the features we added extra. There was something else we wanted to add: Fullscreen. But after completing most of the project, it would be very difficult for us to implement this project. Because we hadn't designed it to be fullscreen from the beginning. That's why we gave up this idea.