|  |  |
| --- | --- |
| Program #3 | Planning Document  James Scott  Colin Riley  Stephen Belden  Shaya Wolf  Neil Carrico  2016-April-08 |

**Project 3**

The first thing we did for this part of the program was read in that raw maze information file. Because the values in the file were ints/floats, we were able to use the convertToInt() and convertToFloat() methods to correctly read the bytes without typecasting. We read in this file successfully, but our code is ugly. Despite constant efforts to clean and maintain the code there are still design flaws, such as this function, that we will fix when we have time.

Next we drew lines on the tiles. These lines matched with the data inside the file and were drawn correctly. We checked with another group as well to make sure that our layout matched the default maze. Assuming they didn’t change their code, it should still match.

You should now be able to move around the tiles wherever you would like. This was included in Project 2, but now the tiles have their picture on them and the game has some resemblance to a maze.

Lastly, we added functionality to the reset button. This was done in such a way that the tiles all return to their original positions on the sides of the game board. It is not an undo button.

This program is finally starting to look like we imagined back in February! We have run into problems with the same exact code working for some computers in the group, but not others. The gameWindow is not displaying correctly for some of us, but works as expected for others. Despite hours of troubleshooting, we have no idea why this is happening. Hopefully we can fix this in the coming weeks.

**Future Plans**

There is still quite a bit to do on this project. We still need to put functionality behind the “New Game” button. Win conditions and legal/illegal moves will have to be enforced. We will also need to do a lot of testing on our final program to find other bugs that we can’t plan for.

Meeting times continue to fluctuate based on class schedules and class presentations. Most work is being done throughout the week separately, and then being combined at the end of a week. We will continue to meet Fridays to finish projects before they are due.

**Anonymous Class Explanation**

Previously, we were instantiating new action listeners in the actionPerformed() method, which required the use of anonymous classes. This implementation was vague and unclear, so we removed these anonymous classes by deleting them and instead called the getActionCommand() method. This is definitely more proper and clean than what we had. We will be keeping these changes and avoiding anonymous classes in the future.

**UML**

<<Interface>>  
Action Listener

GameWindow

+ << constructor>>GameWindow

+ actionPerformed(actionevent)

+ setup()

+ addButtons(GridBagConstraints)

+ setClicked(Tile)

+ readInt(FileInputStream)

+ convertToInt(byte[])

+ convertToFloat(byte[])

+ newButton : JButton

+ resetButton : JButton

+ quitButton : JButton

+ lastClicked : Tile

Tile

+ << constructor >> Tile(int, Point[])

+ << constructor >> Tile(int)

+ getID() : int

+ setID(int) : void

+ getLines() : Line[]

+ setLines(Line[]) : void

+ isEmpty() : Boolean

+ makeEmpty() : void

+ makeLive() : void

+ switchState() : void

+ reset() : void

+ mousePressed() : void

+ debugPrint() : void

<<Override>>

+ paintComponent(Graphics)

+ mouseClicked(MouseEvent)

+ mouseEntered(MouseEvent)

+ mouseExited(MouseEvent)

+ mousePressed(MouseEvent)

+ mouseReleased(MouseEvent)

- ID: int

- lines : Line[]

- isEmpty: Boolean

- border : Border

- NoBorder : Border

**Uses**

JFrame

JLabel

**Uses**

<<Interface>>  
Mouse Listener

Line

+ << constructor >> Line(Point, Point)

+ getBegin() : Point

+ getEnd() : Point

+ debugPrint() : void

- begin : Point

- end : Point