CSIS-3275: Mini Project Report

Group 2

Jagjit Singh - 300316857

Japneet Singh - 300323547

Kosha Raval – 300324116

Vinay Kolli - 300316562

Overview

The application for battleship game has the aim of destroying the computer ships which are assembled on the 7 x 7 grade. There are N ships on the grid covering I cells in total. To destroy a ship the player attacks by filling out the textbox given below and pressing the enter button or clicking on the button, the attack will be done on the ship and the user will be informed by the program whether they have hit or missed the ship. This continues until all the ships have been destroyed. The program keeps track of the number of shots that are hit and miss. The goal of the player is to win by destroying all the ships in the least number of shots.

Added features

Timer: Users are given a choice if they want unlimited time or 100 seconds to play the game.

Randomly allocating ships: Ships are randomly allocated every time the user starts the game.

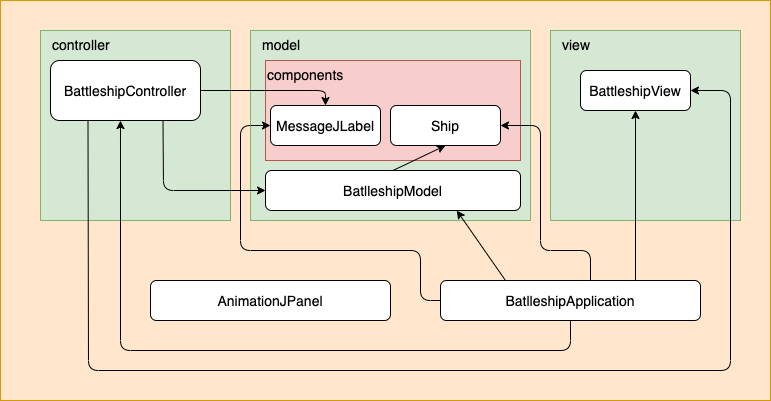
Originality

* Created an animated title on top of the screen.
* Created dialog boxes to get user input when required.
* Created a progress bar to keep track of the remaining time.

Functionality

|  |  |
| --- | --- |
| Class Name | Description |
| BattleshipApplication | The BattleshipApplication is the starting point of the application.  Methods of class:   * runTimer: At the start of the application the user is asked whether they want to play the game for 100seconds or unlimited time. When the user selects “No” the runTimer will set the timer for 100seconds and when the timer runs out the user is notified by MessagePanel. * createShips: This method contains 3 types of lists of strings for the hits and returns the location and number of hits. * createFrame: Creates the frame of the application with following parameters: Dimension screenSize, JPanel messagePanel, JPanel boardPanel, JPanel formPanel, JPanel timerPanel, AnimationJPanel animationJPanel. * createGridPanel: is a method that instantiates a new Jpanel and sets the layout, current size for the location, and paints the full area of the bounds of that panel. * createBoardPanel: This method sets the board image as the background of the frame. * createMessagePanel: is a method is setting the background color to black and displaying the prompts. * createTimerPanel: is a method that gets displays after the user clicks the timer option. It sets the layout, background color to black and adds borders to the entire layout, and returns the timerPanel. * createFormPanel: is a method that consists of a text field that will accept the user input and a button to proceed ahead with the given input. * createJLabels: This method creates the customized JLabels to create a grid layout for displaying the miss and hit when they are to be displayed on the application. |
| AnimationJPanel | The AnimationJPanel class helps in creating a custom JPanel for the application, which contains the animated title of the game.  Methods of the class:   * paint: This method inserts the image in the panel. * actionPerformed: This method is called every 10 milliseconds. This method changes the coordinates of the image, which causes an animation effect. |
| BattleshipController | The BattleshipController class is inside the controller package and it the communicator between the BattleshipView and BattleshipModel class.  The constructor gets the model and view.  Methods of the class:   * parseGuess: This method excepts a string parameter entered by the user in the provided textbox. The parameter has an alphabet and a number which together determine the location of the ship. The alphabet part of the parameter is converted to a number as a row value and the number is the column value. This method also gives a warning to the user if the incorrect inputs such as blank input or when the alphabet or number does not exist on the board. * handleFireButton: This method handles the button click of the application. * processGuess: This method processing the input to determine if the user has missed or hit the ship. It displays an appropriate message for the user. |
| BattleshipView | The BattleshipView class is inside the view package and it creates and keeps the display updated of the application based on the notification provided by the BattleshipModel through BattleshipController.  Methods of the class:   * displayMessage: This method displays a message for the user. * displayMiss: This method sets an image of a miss on the cell when the user guesses the incorrect location of the ship. * displayHit: This method sets an image of a ship on the cell when the user guesses the correct location of the ship. |
| BattleshipModel | The BattleshipModel class is inside the model package and it is responsible for maintaining the states of the application.  The class has a constructor that has 5 parameters: boadSize as int, numShips as int, shipLength as int, shipSunk as int and array of Ships as Ship.  Methods of the class:   * Fire: It needs a parameter of type string which is a guess for the location of the ship. This method helps in determining whether the guess of the user hits the ship or it has missed it. * isSunk: This method is called by the fire method when the guess of the user is true. * generateShipLocations: This method randomly generates the ship location on the board. * generateShip: This method creates a ship on the randomly generated location by generateShipLocation. * Collision: This method uses one parameter of the List of locations that are created by generateShipLocation method to avoid redundancy in ship location on the board. * getBoardSize: method gets the size of the board * getNumShips: method gets number of ships on the board * getShipSunk: method gets ships that have been hit on the board respectively. |
| MessageJLabel | The MessageJLabel is a customized JLabel class in the component package under the model package. This class is used for creating the cells on the UI. The constructor takes one parameter of String which the cell-position.  It also has a method getCellPosition which returns a string object of the cell-position. |
| Ship | The Ship class is in the component package under the model package. The application is storing ships in objects of type Ship. This class maintains two Lists: one to store the location of the ship and the second stores the points that have been “hit” by the player.  The Ship class has the following constructors and methods:   * Ship (List<String>locations, List<String> hits) – The constructions get the list of locations and hits. * Getter and Setter methods for location and hits. |

Application structure



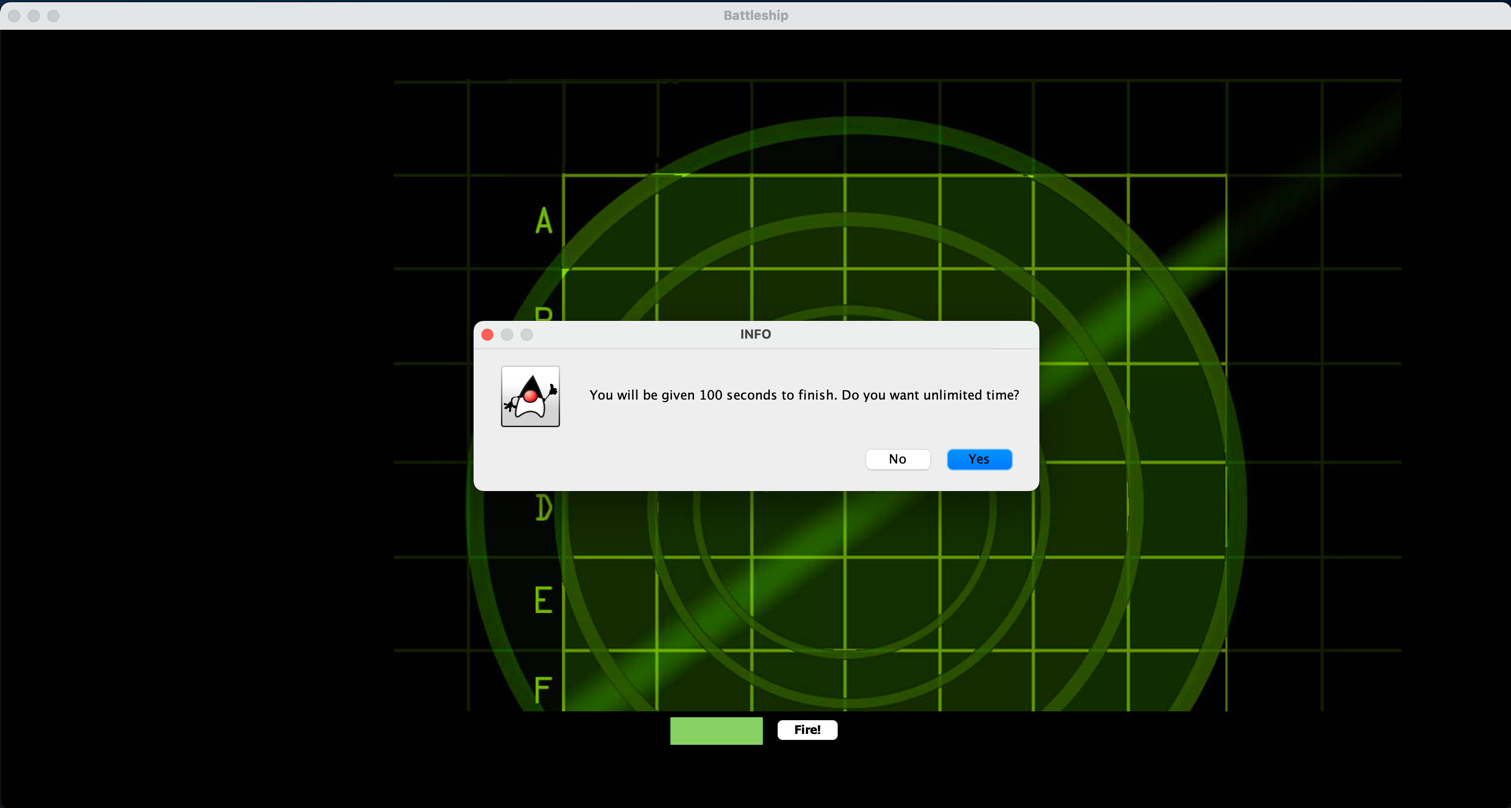
Problem faced

We faced difficulty in aligning the board image cells with the ship image and miss image.

We discussed the problem and came up with the following solution:

* Created a JPanel with the size of the board image.
* Then created as many JLabels as there are cells in the board image.
* Added these JLabels to the JPanel.
* Ship image and miss image are displayed inside these JLabels.

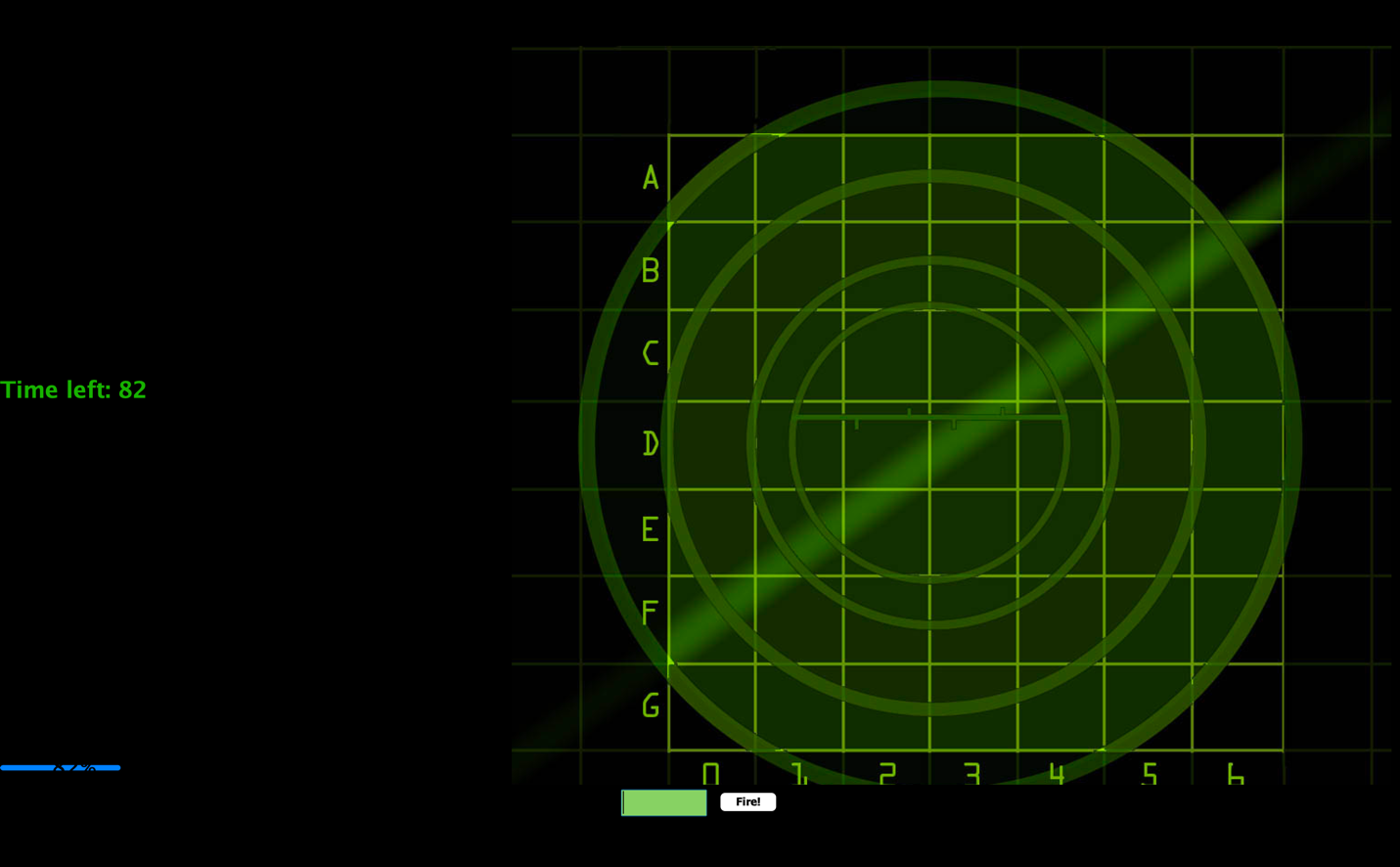
UI Screens













Member Contribution:

|  |  |
| --- | --- |
| **Worked on** | **Member** |
| BattleshipApplication AnimationJPanel | Jagjit Singh |
| BattleshipModel, MessageJLabel, Ship | Japneet Singh |
| BattleshipView, Documentation | Kosha Raval |
| BattleshipController | Vinay Kolli |