Java Game Suite Phase 1

Revision 13

CMSC 495 6382

September 20, 2021

Group Charlie

Sherry Funches, Oyewole Sanusi, Janee’ Jones, Wayne Mack, Jeffrey McGlinn

## **Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision Number** | **Date** | **Description** | **Name** |
| 1 | 9/13 | Creation of Doc and Outline | Oyewole |
| 2 | 9/13 | Overview and Milestone | Oyewole |
| 3 | 9/13 | Phase 1 created/ Modified | Oyewole |
| 4 | 9/19 | Modified Phase 1. Added Phase 2-4 Projected Milestones/Schedule | Sherry |
| 5 | 9/19 | Created Phase 1 Description for Sudoku | Janee’ |
| 6 | 9/19 | Created Phase 1 write up for Word Search | Sherry |
| 7 | 9/19 | Phase 1 documenation for Slider Puzzle | Jeff |
| 8 | 9/19 | Phase 1 Summary Maze | Wayne |
| 9 | 9/19 | Phase 1 documenation for Snakes game | Oyewole |
| 10 | 9/19 | Combined and formated individual files into a unified document. Add table of contents. | Sherry |
| 11 | 9/20 | Add screen capture for maze | Wayne |
| 12 | 9/20 | Formatting, updating page numbers, editing typos etc. | Sherry |
| 13 | 9/20-21 | Review | Oyewole, Janee’, Wayne, Jeff, Sherry |

**Table of Contents**

I. Overview………………………………………………………………………………………..4

1.1 Overview of Project…………………………………………………………………………...4

II. Milestones………………………………………………………………………….…………..4

2.1 Projected Team Milestones……………………………………………………………………4

2.2 Current Status on Phase 1 …………………………………………………………………….6

A. Maze………………………………………………………………………………...….6

B. Snakes…………………………………………………………………………………..6

C. Sudoku …………………………………………………………………………………8

D. Word Search…………………………………………………………………………..11

E. Slider Puzzle…………………………………………………………………………..13

## **I. Overview**

**1.1 Overview of project**

The Java Game Suite program is a simple GUI-based application that provides entertainment to users by allowing them to play several simple single player computer games. The user will be presented with an interface that indicates the available games: a maze, snakes, Sudoku, word search, and a slider puzzle. The user will be able to select a game from the suite and launch it from that interface.

## **II. Milestones**

**2.1 Projected Team Milestones**

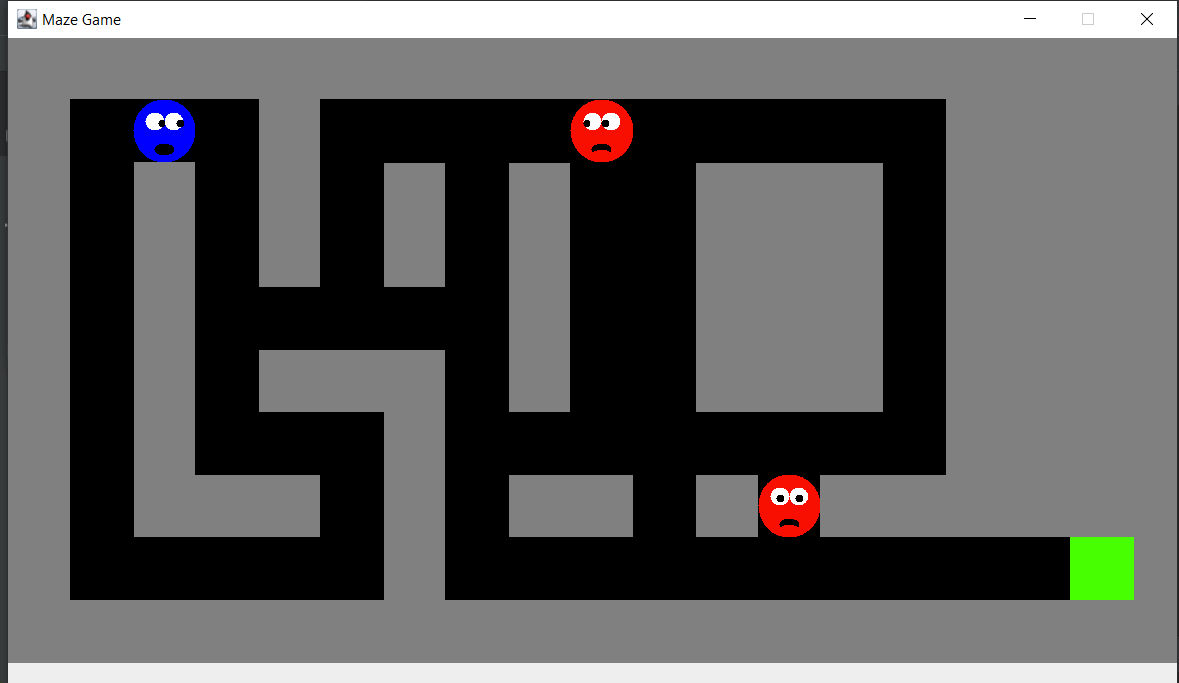
The following table shows our completed and pending goals for the four development phases. The timeline of certain goals have changed based on changes made to the implementation process.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Projected Date of  Accomplished Goal | Name of the Goal | Was the Goal  Accomplished |
| Phase 1 | 9/13 | Design what the system does and how the different pieces fit together. | Yes |
| 9/20 | Start GUI coding | Yes |
| 9/20 | Start other required classes coding | Yes |
| Phase 2 | 9/26 | Continue GUI and other required classes coding |  |
| 9/26 | Finish and share prototypes of working games with group |  |
| 9/26 | Start Testing |  |
| Phase 3 | 10/1 | Integrate all games and main menu into package |  |
| 10/1 | Testing |  |
| 10/3 | Revise code and implement additional features |  |
| Phase 4 | Final Sprint and Delivery | | |
| 10/6 | Finishing Touches |  |
| 10/6 | Update all documents with any changes |  |
| 10/8 | Group Revision |  |

**2.2 Current Status on Phase 1**

**A. Maze**

The maze game has many of its data structures and algorithms built and operational. The main menu was built but is currently incomplete and only runs the game. The game runs in beta displaying only one complete level and repeating it to test out the algorithm. The graphics programming, the data structure for the maze grid, and the logic with the enemies movements have been completed.



What now remains is the remaining levels and the coding for the leaderboard. The current iteration of the game allows the player to play and complete a level. All remaining functionality is currently under development

**B. Snakes**

I’ve completed phase 1 to include all of my design plans, which includes the GUI and algorithm for the snake movements. Our main java code enables us to launch games quickly via the gui interface. Refer to figure 1.1.

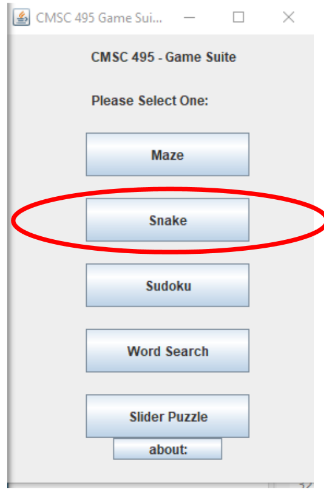
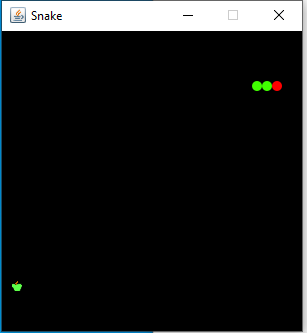
 

Figure 1.1

I’m currently modifying speeds to see which setting best fits the tempo of the game. I’ve also programed the maneuvering keys so players can take control of the snake. Next steps will include coding an out of bounds feature that players cannot cross and will trigger an automatic game over if crossed. Refer to figure 1.2.

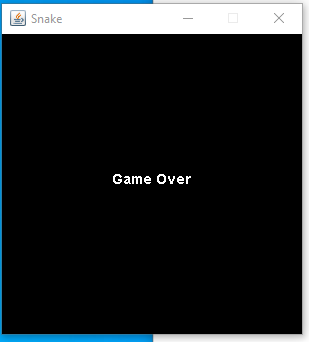


Figure 1.2

**C. Sudoku**

At the beginning of the creation of the Sudoku puzzle for the game suite, the idea was to create three Sudoku puzzles all with a different difficulty level labeled Beginner, Intermediate, and Advanced. The user would be able to go through the main menu, choose Sudoku, then use another menu to choose the difficulty of their choice. Going through it seemed as if it could be done, but trying to keep the simplistic nature, I decided to just create one difficulty level and have various games. The errors or issues that I came about with the three different difficulty levels was having to create another menu within the Sudoku puzzle itself. The layout was there, but the coding was giving me trouble. I gave it about a week to see if I could get something to produce from it, but I kept receiving the same error. Through using Stack Overflow for research options and to gain the knowledge of how to configure a working product, it turned out to be a dead end.

Moving forward, the product being produced is the Sudoku game with one difficulty level, across various games. The user can start a new game, check their progress, and exit. Once they complete the game, clicking exit will take them to the main menu so that a new game can be chosen. Creating the code for this version of the game did not generate too many errors, but there still were some. The Sudoku game was broken down into three separate packages: controller, model, and view. This made the layout more simplistic and where and what needed to be done for the final solution. The controller package would allow the member to choose the button control and sudoku puzzle controls. The button controls display to the user options to select a new game, check a game, and exit a game. The sudoku puzzle controls display to the user the numbers as well as the help. The user can choose if they would like the help on or not. If the help is on, then it will display a blue background once the number is chosen. This shows the user the number chosen goes in the appropriate boxes. The model package would entail the game code and the action code. The game code would let the user know that the numbers 1-9 would only be allowed in each 3x3 block, as well as in each horizontal and vertical line. No numbers would be able to repeat and each block will hold the correct number chosen. Using the check button, the user will be able to check to make sure that the numbers are being inputted into the appropriate box. Lastly, the view package holds the main class, which is the Sudoku class, the button panel, the field, and the sudoku panel. The view package brings everything together that is run from the Sudoku class.

Calendar

Description automatically generated

Figure 1: User is presented once Sudoku is chosen from the main menu.

A screenshot of a computer

Description automatically generated with low confidence

Figure 2: User selects number, with the help on, shows where that number belongs.

Table

Description automatically generated with medium confidence

Figure 3: User checks progress, Green annotates that number is entered in correct block, Red annotates that number is incorrectly entered in block.

Table

Description automatically generated

Figure 4: User has completed the game. User can select New Game to play again (new game board generated) or Exit Game to be directed to the main menu.

**D. Word Search**

The primary goal for the word search program’s first development phase was to add functionality to the mock-up GUI that was created for the user guide. This goal was successfully achieved and overall my implementation was consistent with plans indicated in the user guide and design plan. The first hurdle was to settle on how to display the letters on the word search grid so that the program could determine if highlighted letters matched a word on the word list. I considered two options: using Java2D graphics and creating TextLayout objects or using JLabels and a JPanel with a GridLayout. The benefit of former was that I could potentially do hit testing on individual letters. However, creating my own grid in a modified coordinate system with the letters evenly spaced in the desired manner, especially if the size of the puzzle changed, proved challenging. I tested and opted for the latter approach.

Next, I implemented and tested the system’s ability to recognize highlighted words. I made a design decision that only a continuous full stroke of the cursor over the center of a word’s letters would be a valid highlight. To determine if a line went through a letter I iterated through an ArrayList containing all the JLabels displaying letters on the grid. I determined that if I centered the text in a JLabel, calculated the center of the label, and drew a Rectangle2D object around the text, I could see if a line intersected the letter. Using set operations I determined if a set of letters for a word were exactly the same as those that were intesected by a line.

The next functionality that I tackled was modifying the word list so that when the user clicks on word a strikethrough will be drawn through it and when unclicked the line would be removed. I opted here to use JLabels to hold the text and calculated the length and position of the line to be drawn through the text by using FontMetrics to measure the length of the text in the label. I used a HashMap to store the line associated with a JLabel so it could be easily be located and removed if clicked.

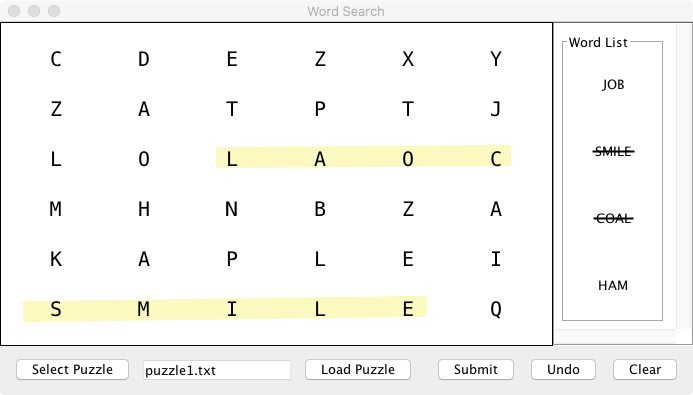


Figure A. Implementation of highlights and strikethroughs

Lastly, I added basic functionality to the JButtons. First I determined having the user select a puzzle from a drop down menu as specified in the user guide would be problematic if games are added or removed in the future. Instead of opting for a configuration file to create the list of games which would have to be updated every time a game changed, I decided to use Java’s JFileChooser and have the user select a game from a specific puzzle file directory. This involved changing the display and adding a warning if the user tries to submit an empty file. I will update the user guide and test plan to reflect these changes and test that the user cannot access files from other directories. I also implemented the clear button to remove all lines and strikethroughs from the board and word list and the undo button to only remove highlights from the grid.

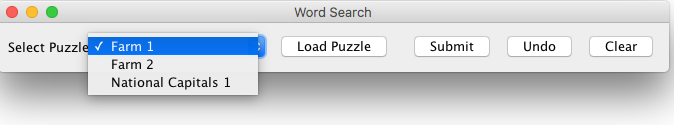


Figure B. Original selection of puzzle from a drop down list.

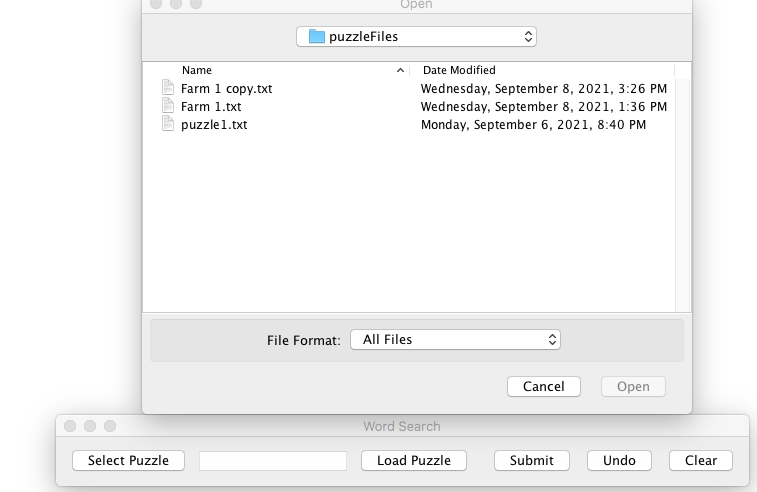


Figure C. New implementation using JfileChooser to select files.

In sum, the word search’s development is on schedule. Integration of the module that reads data files with puzzle information and the completed prototype is scheduled for phase two. The user guide and test plan will be updated to reflect the new file selection process.

**E. Slider Puzzle**

The portion of the Java game suite that I worked on this week is a picture slider puzzle. My goal for phase 1 was to have the GUI complete and begin to implement the button handlers and other controls so that I could have a prototype by the end of the week. First, I created a GUI that presents the user with a 4 x 3 grid of image tiles, a brief instruction informing the user how to complete the puzzle, an option to change the puzzle, and puzzle hints that display the completed puzzle image. I did not encounter a lot of issues when creating the GUI other than resizing the image files so that they would not appear distorted as a complete image or as individual image tiles. Below is an image of the GUI after launching the program:

A collage of a cat

Description automatically generated with low confidence

Next, I implemented the button handlers and other controls that allow the user to complete the puzzle, choose a new puzzle and display the puzzle hints. Below are screen captures of the GUI after choosing a new puzzle and selecting a hint:

Background pattern

Description automatically generated

Graphical user interface, application

Description automatically generated

Lastly, I needed to make sure the winning combination would display a message to the user informing them that the puzzle is complete:

Fireworks in the sky

Description automatically generated with low confidence

Overall, I feel that phase one was successful because I have a working prototype. Moving forward, I will be focusing on improving the GUI and its functionality, as well as focusing on integrating the game with the complete Java game suite.