Martin Capisanan

Andrew Frazier

Prof. Mushkablat

COMP 467

4 December 2017

**1. Project Name**

Minesweeper

**2. Project Goal**

Minesweeper game consists of a group of squares where each square represents a region. Each region either contains a mine or not. Your mission is to reveal (or "open") all the mine-free regions and mark the ones that contain mines. You open the region by left-clicking it, and mark a region by shift-clicking. On top of that this minesweeper game has extra features that is attracted for audience from 4 years old to older.

**3. Project Description / Abstract**

The project is a pc app that is a game similar to the original minesweeper, but it has more upgraded features compared to the original simple minesweeper. It will have 3-4 different types of visual effects that will make the game more appealing to the eye.

**4. User Specification**

- Nonfunctional and Functional requirements

- Functional - features, functions, UI/UX of the project

* Main Menu
  + User has the ability to choose to start a new game and quits the application.
* Options
  + User will be able to turn off visual effect and sound effects.
* Pause Menu
  + The ability to pause the game if the user chooses not to continue the game anymore at the moment.
* High Scores
  + Users will be able to see his/her highest score attempted in the minesweeper app game

- Non-functional requirements - Executional performance

* The application should be able to run in windows at an optimized level.
* Frames Rate -The base casing rate must be twenty edges for each second. The normal casing frame must be more noteworthy than 30. Edge rate can be screen specifically from the realistic motor.
* Response Time - The normal reaction time amongst clicks and response must be under 0.5 seconds. The most extreme reaction time amongst snap and response must be two seconds. Including some straightforward classes and strategies that will register and show the time expected to process any operation can test this necessity.
* Maintainability - The code composed for the diversion must be viable. This can be accomplished by gathering measurements, for example, DIT (depth in inheritance), MPC (message-passing coupling), WMC (weighted method complexity) and DAC (data abstraction coupling). Likewise including documentation will enhance the viability size of the framework.
* Required Sources - The diversion must have the capacity to keep running with least of 512 MB of RAM. The diversion must utilize short of what one gig of hard plate space. Checking the aggregate size of the organizer in which the amusement was introduced, for the hard circle space can test this necessity. For the Slam utilized, when playing the amusement, we can check the physical memory in the Windows Task Manager performance tab.

**5. Architecture**

* Unity Engine
  + Model: Image/Tool Settings/Data
  + View: C# and assembly
  + Controller: C# script that runs on Unity preface

UML Diagram

LoadScene(winMenu)

Display High Score

Replay

Options

Quit

LoadScene(loseMenu)

Display High Score

Replay

Options

Quit

QuitApplication()

Exit Game

Adjust Volume

Toggle Sound FX

Menu listener

Quit listener

ShowPanel(Options)

LoadScene(mainGame)

Escape button listener

Win event listener

Lose event listener

Game Algorithm

Main Menu Class

Start button listener

Options listener

Quit listener

**6. Code Guidelines**

- Naming conventions

**Game Algorithm**

Load Unity prefabs; //get prefabricated unity objects into the main game scene

Attach Scripts to game objects;

Initialize tile objects with Element Script;

Utilize Random to determine which elements are mines;

Add all Mines to a 10x13 grid array;

OnClick()

If element is a mine{

Show all mines;

Load lose screen;

}

Else

{

Show number of adjacent mines;

If no adjacent mines{

Uncover all adjacent empty elements;

}

For each element in the grid

If the element is not a mine and is covered

Continue game

Else

endgame

}

-Our code will be organized using the camelCase standard. To ensure that code looks uniform and is easy to read, we will strongly use indentation

1. Below is a list of possible choices, here is a link for popular choices in JS http://sideeffect.kr/popularconvention#javascript
   1. camelCase - first letter begins with a lowercase letter and the following words are uppercase.
   2. Underscore - all variables start with underscore
   3. Space vs Tab with indentation
   4. Variables starting with a space

- Paradigms

- Event driven

* Call routines
* Click event listening

- Functional

* All objects in the app has dependencies

-Structured

* Pre-fabricated assets controlled by the Unity Engine and C# scripts

- Framework

- Unity 5.0

- Error handling technique

- Unity Engine fault checking

- Unity Engine fail safe known issues

- Logging

- Logging for user data high scores.

- Security guidelines if any

- Constraints

User, Hardware, Software

- Must have a 512mb of ram installed in the computer

- Only works on windows computers.

- Usability guidelines

- Menu organization, buttons, links, etc.

- There are three buttons the first button is a start button the second button is options and the third button is quit.

- The title of the project is also in the home page which is called   
Minesweeper :P

- When the user finished the game it will show his/hers best completion time in seconds. Also the best time completion will always show on the right side of the game.

- Templates

The Unity Engine has its own environment using pre-fabricated objects.

-Conclusion

Making guesses is sometimes inevitable. We try to make the guesses as educated as possible. The chance of winning depends mostly on the mine density, but also on the board size. Minesweeper can be challenging but it is also fun.