*Created: June 05, 2012*

*Latest Revision: June 06, 2012*

Minesweeper Technical Design Document

Minesweeper is a single player puzzle game in which the player must use an elimination process to identify all of the safe positions in a mine field. The puzzle is solved when all mines are flagged and the rest of the tiles in the minefield are revealed.

# Section 1: Overview

This Technical Design Document will go over the components necessary to make a complete minesweeper game, as the algorithm used to determine what symbols to place where in the puzzle’s playing field. It is key that this code is planned first, as determining what to draw on the screen must both be 100% accurate as it is a puzzle game, and must be calculated quickly and efficiently so that it is both runnable on any machine no matter how powerful, and fast enough so that the game does not limit the pace at which the player plays. Also this code must be written as this game is being created from the ground up, without using any code from an outside source, which will save money from having to pay royalties.

The main elements of this game are:

* Game
* Game Board
* Minefield
* Mines
* Timer
* High Score Table
* Player

Game

Game Board

Minefield

Mines

Timer

High Score Table

Player

Input

* Input

# Section 2: Main Concept

The main elements of this game explained:

* Game *Contains all the elements of the game except the high score table.*
* Game Board *Contains a list of all the high scores in the game.*
* Minefield *Rectangular grid with mines placed randomly in it, which acts as the* *playing field.*
* Mines *Instant-kill objects which causes a game over if revealed.*
* Timer *Used to determine score. The faster the players solves the puzzle, the higher up on the high score list they are placed.*
* High Score Table *Contains a list of all the high scores in the game.*
* Player *User that plays the game.*
* Input *A keyboard and mouse are used as input. A mouse to play the game, and a keyboard to enter names into the high score table.*

# Section 3: Details

The game encompasses the entire project. The game contains three main components: the Game Board, the High Score Table and the Player.

The game board has a minefield, which by default is one of three dimensions, based on the difficulty chosen by the player (which can be chosen from the game menu or upon starting a new game).

Game board



Timer

Flag

Mine

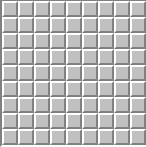
Minefield

Mines Left Counter

*Minesweeper Concept*

Play Options: 9X9 with 10 mines, 16X16 with 40 mines, or 16X30 with 99 mines, or can be customized with a height between 9-24 tiles, width between 9-30 tiles, number of mines between 10-668.

Initially the mine field is completely blank, which can look like so:

 or 

Default layout of a 9x9 mine field.

As shown in the very first image (Minesweeper Concept), the minefield depicts numbers and blank revealed squares (tiles). To determine what to place where the following rules should be used:

* The number shown on a square represents how many mines are adjacent to that square, whether orthogonally or diagonally.
* If no mines are adjacent, the adjacent squares should be revealed until the border of the revealed squares are all numbered squares or the edge of the playing field.
* This “multiple-revealing” mechanic is key to both the game’s playability as well as enjoyment factor, as there is no way to click a mine on the first click of a game, thus allowing users to hope for a good “multiple-revealing” starting click, thus acting as one of the many ways to provide additional incentive to play multiple games.

Algorithm:

* + If a square is not revealed:
    - Move on to the next square.
  + If a square is revealed:
    - If the square is a mine:
      * The game ends and the high score table is brought up.
    - If the square is not a mine:
      * Figure out what to draw on all of the surrounding squares.
      * If a surrounding square is a wall:
        + Simply skip it and move on to the next surrounding square.
      * If a surrounding square has already been revealed:
        + Move on to the next surrounding square.
      * The order in which to surrounding square to move to is:
        + First look at the square adjacent and to the right of the initial square.
        + Then rotate in a clockwise rotation around the initial square checking both orthogonal and diagonal squares along the way.
        + A maximum of 8 squares should have been checked.
      * If the adjacent square has no adjacent mines:
        + Reveal it as a blank space and add this square to the list of squares to be checked using the algorithm again.
      * If the square has adjacent mine(s):
        + Display a number representing the number of adjacent mines.
  + If a square is right clicked:
    - Toggle through the 3 states of an unrevealed square:
      * Flag
      * Question Mark
      * Default (blank, not revealed)
    - If the square is toggled from a default to a flag:
      * Subtract one from the mines left counter.
    - If the square is toggled from a flag to a question mark:
      * Add one to the mines left counter.
    - If the square is toggled from a question mark to a default:
      * Do nothing, continue as usual.

Another component of the game board is the timer, which counts up in seconds when the player makes their first click (which officially starts the game). Here are some examples of what it can look like:

 or 

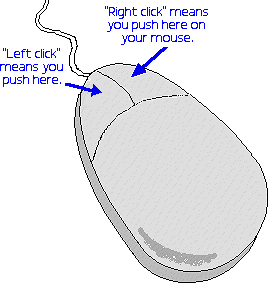
After each game the score is reset back to 0 as when the game ends the player has the choice to enter their name next to their time to complete the game in a high score table. The high score table can vary from something simple such as a small top 10 list (common in offline only versions of Minesweeper) to huge databases of high score information (common in online websites dedicated to the game). The timer is calculated by polling the system clock, and converting from clock cycles to seconds like so:

(clock ticks final – clock ticks initial) / computer clock speed = seconds elapsed.

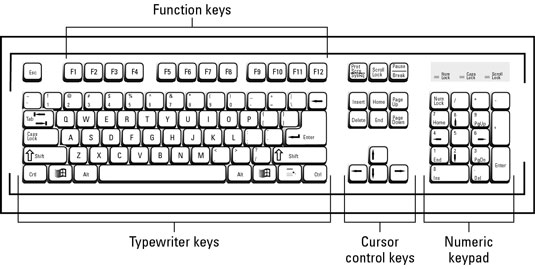
Here we can see that two important factors be kept in mind:

* The clock ticks and clock speed must only be read, and not changed, in order to prevent any changes to the computer system itself.
* The timer and high score table must communicate with each other fluently, meaning that if this code is being written my multiple coders, the input and output from each must be compatible with each other.

Since minesweeper is a single player game, the game is controlled using a single mouse and keyboard. The mouse is used exclusively to play the game. The left mouse button is used to reveal a square on the minefield by clicking a square, and upon release of the mouse button the square (tile) underneath is revealed. The right click button is used to toggle through markings for squares, going in order from flag, question mark, blank. (Where a flag signifies that the user believes that a mine is under that specific squares, the questions marks is a special use marking for the user to use however they feel necessary and blank is the default, unrevealed state of a square.)

(A work-a-round to using a mouse can be to use a keyboard, in order to control the cursor using the arrow keys to point to specific squares, and use other keys such as enter to replace left clicking and space bar to replace right clicking.)

The keyboard control is needed in order to enter a user’s name into the high score table with ease.

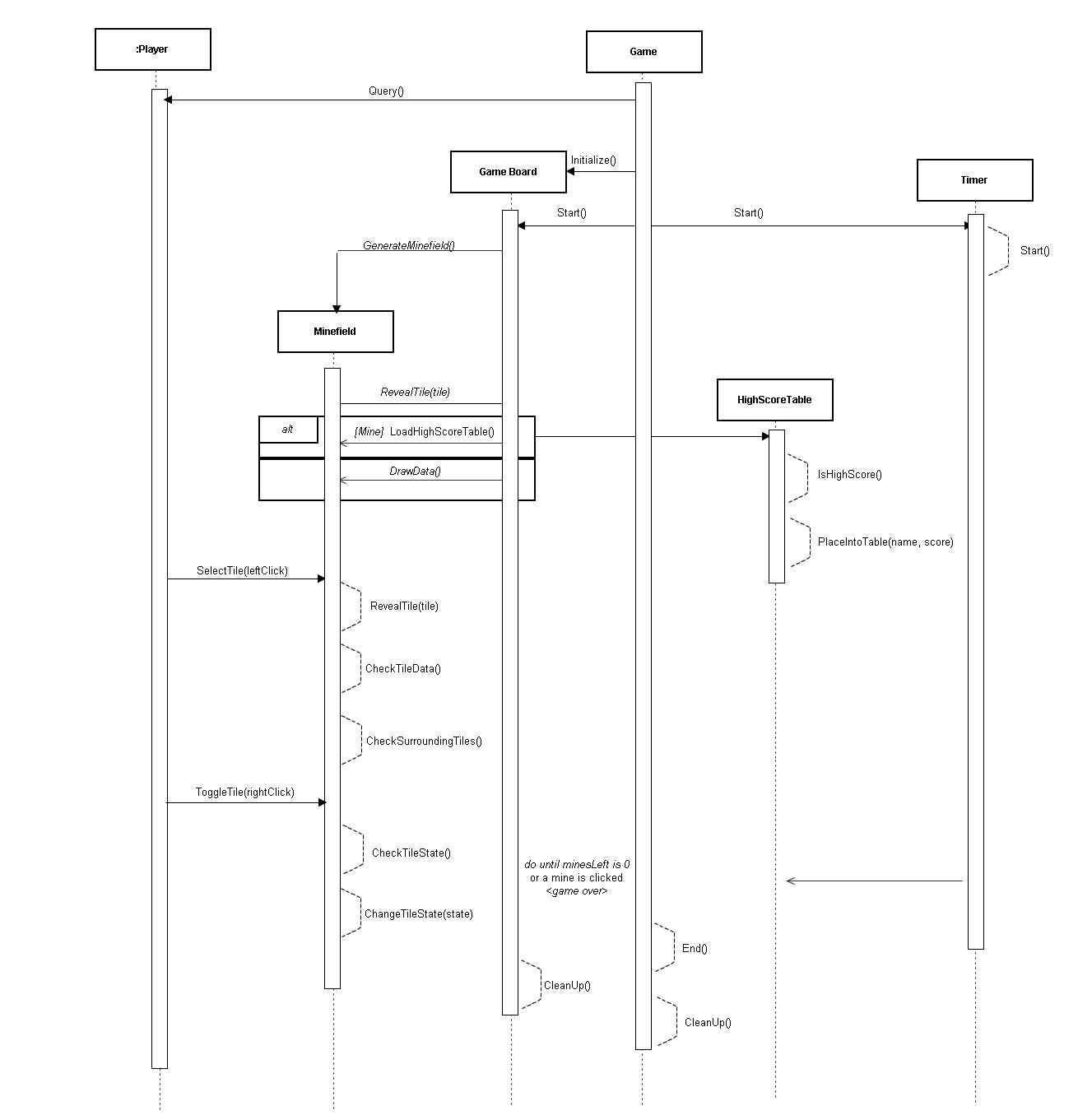
(A work-a-round to not having a keyboard can be to have arrows to cycle through letters to enter a name that way, such as how high score tables in arcades and olden games had only 3 letters to be entered as a name for high score lists.)

Using a simple interface such as this, the game is very simple and intuitive to control, thus showing the usefulness of encapsulating the controls to simply show results to the user, without them actually having to know how everything works behind the scenes.

A potential issue with having a default setup assuming that the user will be using both a keyboard and mouse can be that if not both a mouse and keyboard are present the user may have difficulty figuring out the alternate control schemes. A work-a-round this can be to detect the current hardware plugged in at launch of the game, and any time that a change in state of the current hardware occurs, and display an appropriate help screen when necessary if needed. A new issue arises with this however that these help menus may bother the user, so an option can be added to the game to allow the user to never view the help screens ever again when viewing the help screens.

Keeping all of the above in mind, the last potential issues that can occur is that if the user wants to play this game on a different system. The current design is compatible with Windows and Linux computer systems. If additional computer systems need to be implemented, it should not be too difficult as the code can be easily converted if any difference in system settings, requirements, or syntax conditions exist.

## UML DiagramF:\Semester 4\VGP112 UML & Technical Documentation\Assignments\Final Assignment\MinesweeperUMLDiagramv6.jpg

Sequence Diagram

# Section 4: Benefits, Assumptions, Risks/Issues

*Top 5 Benefits to my choice of design:*

1. A top 10 high score table versus a top 1 high score screen provides a greater feeling of accomplishment when the player gets close to beating the top high score, and allow for multiple users to see their progress compared to others who play the game.
2. Guaranteeing that the user can not lose on the first click removes the negative feelings that may be unnecessary in a game that should exist purely for enjoyment and learning’s sake.
3. Allowing multiple control methods allows users to play minesweeper when they can not play other games due to hardware restrictions.
4. The design retains enough simplicity to allow ports to additional devices.
5. Having all adjacent blank squares be revealed when revealing a single blank square prevents boredom for users by completing a simple task for them which does not affect the difficulty of the game at all.

*Known Risks & Issues:*

* Hardware issues such as a mouse only having a single button or a keyboard that does not have a compatible key system.
* Game data can easily be deleted by users.
* Game code has no security, so the game code can be stolen or changed easily.
* Royalties may need to be paid to the original creators of minesweeper.

*Assumptions:*

* The user’s computer is powerful enough to run all calculations quick enough.
* The user’s computer allows writing to the disk, in order to save data.
* The user’s display has a graphics renderer, which can display images in windows.

*Summary:*

* Minesweeper is a single player puzzle video game, in which the user must use a process of elimination in order to solve the puzzle.
* The puzzles are randomly generated, and the player’s first move is always safe.
* The player deduces which square to reveal by looking at the state of other nearby squares.
* A timer counts up starting upon the first reveal, which is used to determine the score of a player.
* The player’s high score, if fast enough, is saved in a high score list, broken into categories based on difficulty.
* The game is fast, educational and a bringer of entertainment.

# Section 5: Appendix & Bibliography

**Research**

<http://en.wikipedia.org/wiki/Minesweeper_(video_game)>

<http://en.wikipedia.org/wiki/Minesweeper_(Windows)>

**UML Diagram and Sequence Diagram**

<http://www.gliffy.com/>