Team 5
Minesweeper-CLI
Software Design Document

ID 1100 CS Module
31st October, 2015

Group Members:

Libin N. George,
Jude K. Anil,
Harsh Yadav,
Anish M M,
Adrian McDonald Tariang

Introduction

Purpose

This Software Document describes the specification to the “Minesweeper-CLI” game we are to develop in the C programming language, similar to the game popularised by Microsoft.

Scope

The “Minesweeper-CLI” is a single-player puzzle video game where the objective is to clear all the hidden mines in a given field, without the detonation of any of them. This is done by analysing the board and clicking on the field units, giving us clues about the number of neighbouring mines in each field. This process is repetitive until and unless the player identifies the position of all the mines, which means, he wins the game, the other possibility being, clicking on a mine and henceforth, losing the game.

The given software is to run entirely on the Command Line Interface (CLI) and where the inputs are to be made through coordinate inputs through the keyboard.

The minefield is auto generated by the machine at the start of the game using random functions to allocate the mines.

Reference Material

“Minesweeper” – A game by Microsoft.

“Minesweeper (Video game)” page on Wikipedia.

Definitions and Acronyms

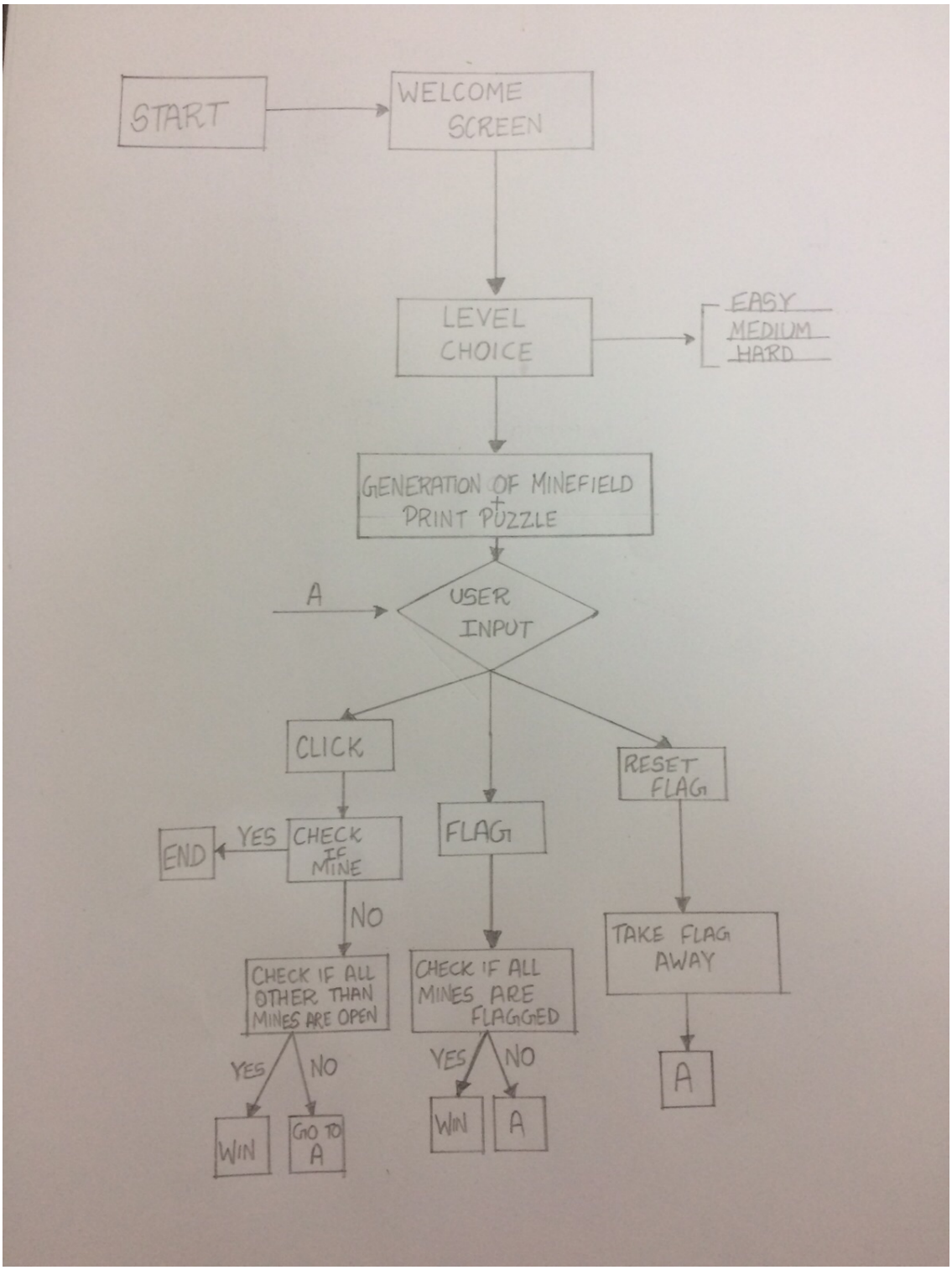
“George” – A gaming enthusiast, one that the world has labelled a GEEK (yes, we are moving in to stereotypes). George plans on cracking and squashing every single puzzle our software throws at him and he still believes that he will get accepted in “Hogwart’s School of Witch-Craft and Wizardry.”

System Architecture

Architectural Design

For the sake of division of labour and modularity, we have decomposed our project into different and dedicated sections which make sense in a workflow.

Flowchart


START
WELCOME SCREEN
LEVEL CHOICE
EASY
MEDIUM
HARD
GENERATION OF MINEFIELD
PRINT PUZZLE
A
USER INPUT
CLICK
FLAG
RESET FLAG
END
YES
CHECK IF MINE
NO
CHECK IF ALL OTHER THAN MINES ARE OPEN
CHECK IF ALL MINES ARE FLAGGED
TAKE FLAG AWAY
YES
NO
YES
NO
WIN
GO TO A
WIN
A
A

Particulars for each module:

1) Start- Opens to a screen which introduces the game and gives George an option to play or leave.

2) Select Level – Gives him an option to either play on 'Easy', 'Medium' or 'Hard'.

3) Create Field – Creates the minefield with specification passed to it and also sets the mines within defined constraints.

4) Display Puzzle- Displays the puzzle after refreshing the screen, giving the user a feeling of dynamic change of the screen's content.

5) User Input – Takes input from the user on how he'd like to solve the puzzle; The user has a choice between opening a minefield unit, marking a unit as a mine or even resetting a marked region.

6) Check for mine- Simply checks if a mine is present, if so, terminates the game by going the Lose module, otherwise, it opens the slot to show the number of mines in its proximity.

7) Win – Congratulates the user on successfully solving the puzzle

8) Lose – Takes the user out of current game and asks him to try again.

Data Design

Data Description:

The earlier stated minefield in the game is a series of two-dimensional arrays which would help the game to keep track of the location of the mines, the user's previous clicks and also the number of mines in the vicinity of each field unit.

List of 2 dimensional arrays:

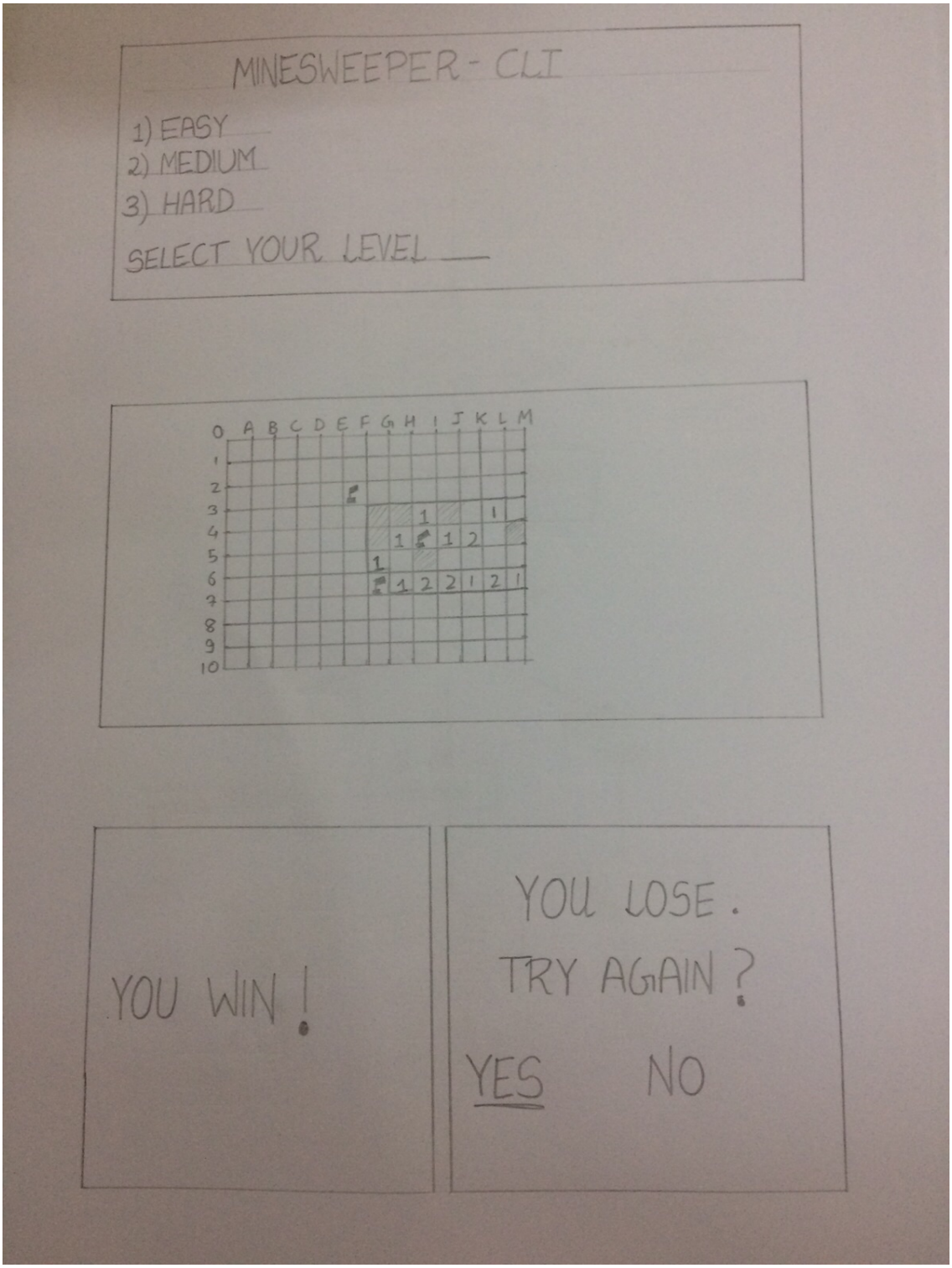
1) Mine map – Stores the locations of the randomly positioned mines, it also stores the number of mines around the particular field.
2) User clicks – Keeps track of the mines the user has triggered this also included the field units flagged as bombs.

Human Interface Design

Overview of User Interface

The general user interface used for this project is the Command Line Interface on a terminal where the use and printing of only ASCII characters would define the menu and the minefield of the game.

Screen Images



Research Section

Colored Input in Terminal

```
printf("\033[031m This text is red \033[0m This text has default color\n");
```

The snippet given above is an illustration on how we can change the foreground, General formatting and the background colour to text printed in the terminal.

The Syntax for the following are as follows :-

```
ESC[{attr1};…;{attrn}m
```

Where **ESC,** in our case, is “\033”

[literal opening square bracket.

{attr1}is an attribute for formatting for the foreground, background and general formatting.

The list of possible attributes are:

General Formatting:

0 Reset all attributes

1 Bright

2 Dim

4 Underscore

5 Blink

7 Reverse

8 Hidden

Foreground Colors:

30 Black

31 Red

32 Green

33 Yellow

34 Blue

35 Magenta

36 Cyan

Background Colors:

40 Black

41 Red

42 Green

43 Yellow

44 Blue

45 Magenta

46 Cyan

47 White

Clear Console Screen

We need to include the given header file and then we can use the function `clrscr()` to clear the given screen.

Directional Keys in a console application

For directional inputs, we notice that we had to find a way to enter the key strokes without typing the ENTER key.

We found the **`getch()`** function to help us with this part, which was the perfect tool to meet our requirements.

Now, to implement the use of the directional keys, we use the following snippets of code.

```
#include <stdio.h>
#include <conio.h>

/* System dependent key codes */
enum
{
 KEY_ESC    = 27,
 ARROW_UP   = 256 + 72,
 ARROW_DOWN  = 256 + 80,
 ARROW_LEFT  = 256 + 75,
 ARROW_RIGHT = 256 + 77
};

static int get_code ( void )
{
 int ch = getch();

 if ( ch == 0 || ch == 224 )
```

```
   ch = 256 + getch();

  return ch;
}

int main ( void )
{
  int ch;

  while ( ( ch = get_code() ) != KEY_ESC ) {
    switch ( ch ) {
    case ARROW_UP:
      printf ( "UP\n" );
      break;
    case ARROW_DOWN:
      printf ( "DOWN\n" );
      break;
    case ARROW_LEFT:
      printf ( "LEFT\n" );
      break;
    case ARROW_RIGHT:
      printf ( "RIGHT\n" );
      break;
    }
  }

  return 0;
}
```

Possible Future Improvements

1) Mouse Inputs being used instead of coordinate input; making the game easier for the user.
2) A 'Custom' Level in the game selection menu which allows the user to define their own map size and number of mines in the field.