|  |
| --- |
| 2016/17 |
| **Prison Escape** |
| Adv. Higher Project |

|  |
| --- |
| Colver, Alisdair John |

# Abstract

Prison Escape is a logic based puzzle game that takes inspiration from the game minesweeper. The aim of the game is to locate the warning lights so that you can successfully escape from prison. Once an area has been cleared the user can choose to increase the difficulty or try again at the same difficulty. There will be a high score table to store the fastest times that the three different difficulty puzzles have been solved in.

Due to this being based on minesweeper I originally created a minesweeper game and then made changes so that it would become *Prison Escape*, this is why occasionally in the project I will refer to mines – this should be taken as warning lights.

The end-user group of this project is people over the age of around ten. It’s most likely users are probably young adults that enjoy logic based puzzle games; however there is scope for the application to be enjoyed by a larger group of users.

Prison Escape has an interface appropriate for users that validates all inputs and will also interface with stored data. Prison Escape also includes 2-D arrays, arrays of records, a sort algorithm and recursion

This project has allowed me to apply generic skills such as research, planning, analysis, problem solving and evaluation. The project will be completed in the given time and any health and safety issues, permissions and logistics have already been dealt with. Prison Escape was managable within the timescale I have been given and I had access to adequate expertise, resources and equipment.

# Project Proposal

## Aim

To create a logic and strategy game, based on Minesweeper, that can be played by players of almost any age to improve their logic and reasoning skills.

## End-User Group

Men and women, of age ten and above, who have an interest in logic and puzzle games. It will probably appeal most to users that are young adults.

## Features

* Use of visual basic
* Different game difficulties
* High score tables
* Timer to count how long it takes to complete a grid
* Multilevel navigation

## AH Criteria

* Interface that validates all inputs
* Interfaces with stored data
* 2D arrays
* Arrays of records
* Sorting algorithm
* Recursion

# Feasibility Study

## Technical

This project is technically feasible as I have access to all the software and hardware need to complete the project. I will have to make use of online support forums to ensure that I can successfully write the user control required to create the game and to gain new technical knowledge that I will require to be able to successfully complete the project. Although I have Advanced Higher level programming skills there will be certain things that I will come across in this project that I have not previously come across. For example, I plan to create a user control that can cycle through different designs – this will involve using code to implement drawing methods which I haven’t done before. I will have to make use of online support forums to ensure that I can successfully write the user control required to create the game and to gain new technical knowledge that I will require to be able to successfully complete the project.

## Economic

This project is economically feasible as there will be zero costs throughout the analysis, planning, design, implementation or testing of this project. Visual Studio Community 2013 is the software that I will be using to implement my solution and since this is an open source software package and there will be no costs for download and use of this software. All other software that I will use is already available on school computers so there will be no costs here for licensed software packages.

## Time

This project is feasible in the allowed time of 40 to 50 hours. To ensure that I manage to successfully complete the project in this timescale I will allow time for planning the time for all the different aspects of my project and sort this using a Gantt chart so I know whether I am on time or not. The use of these progress tracking methods should ensure that the project is complete and finished in the allowed time.

## Legal

This project is legally feasible. The Copyright, Design and Patents Act may have some implications on this project since Prison Escape is based on the Microsoft game version of the game Minesweeper – however my version will be different enough so that it will not infringe on any intellectual property rights owned by Microsoft. In terms of media I will be personally creating all graphics myself so there will be no implication of The Copyright, Design and Patents Act in this sense. The project will not have any implications in terms of the Data Protection Act as no personal data that identifies one specific person will be stored.

# Project Plan

This project will consist of all the below stages in the waterfall development method:

* Planning
* Analysis
* Design
* Implementation
* Testing
* Evaluation

In order to ensure that my project can meet the feasibility requirements previously mentioned I will set deadlines for each of the main stages of my project to be completed. This will be broken down more specifically in a Gantt chart to create a clear and complete plan of each individual stage along with how much time each stage should take me. Doing this will ensure that I can successfully meet all of the requirements.

## Project Deadlines

|  |  |  |
| --- | --- | --- |
| Stage | Time (approx.) (hours) | Target Date (approx.) |
| Planning | 4 | 30/01/2017 |
| Analysis | 5 | 02/02/2017 |
| Design | 5 | 16/02/2017 |
| Implementation | 22 | 16/03/2017 |
| Testing | 6 | 24/03/2017 |
| Evaluation | 3 | 31/03/2017 |

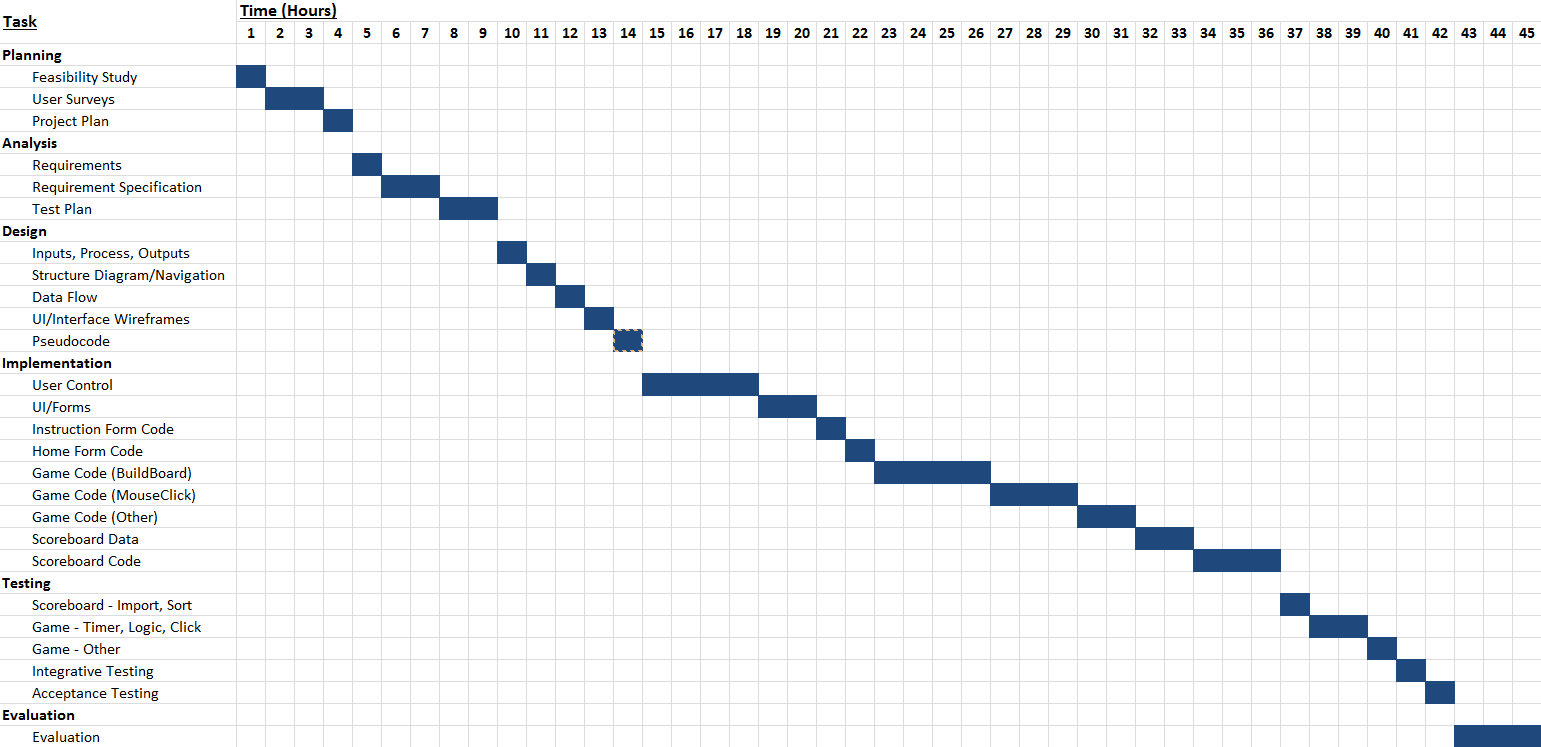
I have allowed 45 hours to complete the project. This allows for a five-hour window on either side so that I have some flexibility in taking more or less time on stages that require it. However, to complete this project on time the final target date must be met; to ensure I manage this I will use my Gantt chart and progress diary to keep track of what I have done and what I still need to do.

## Resources

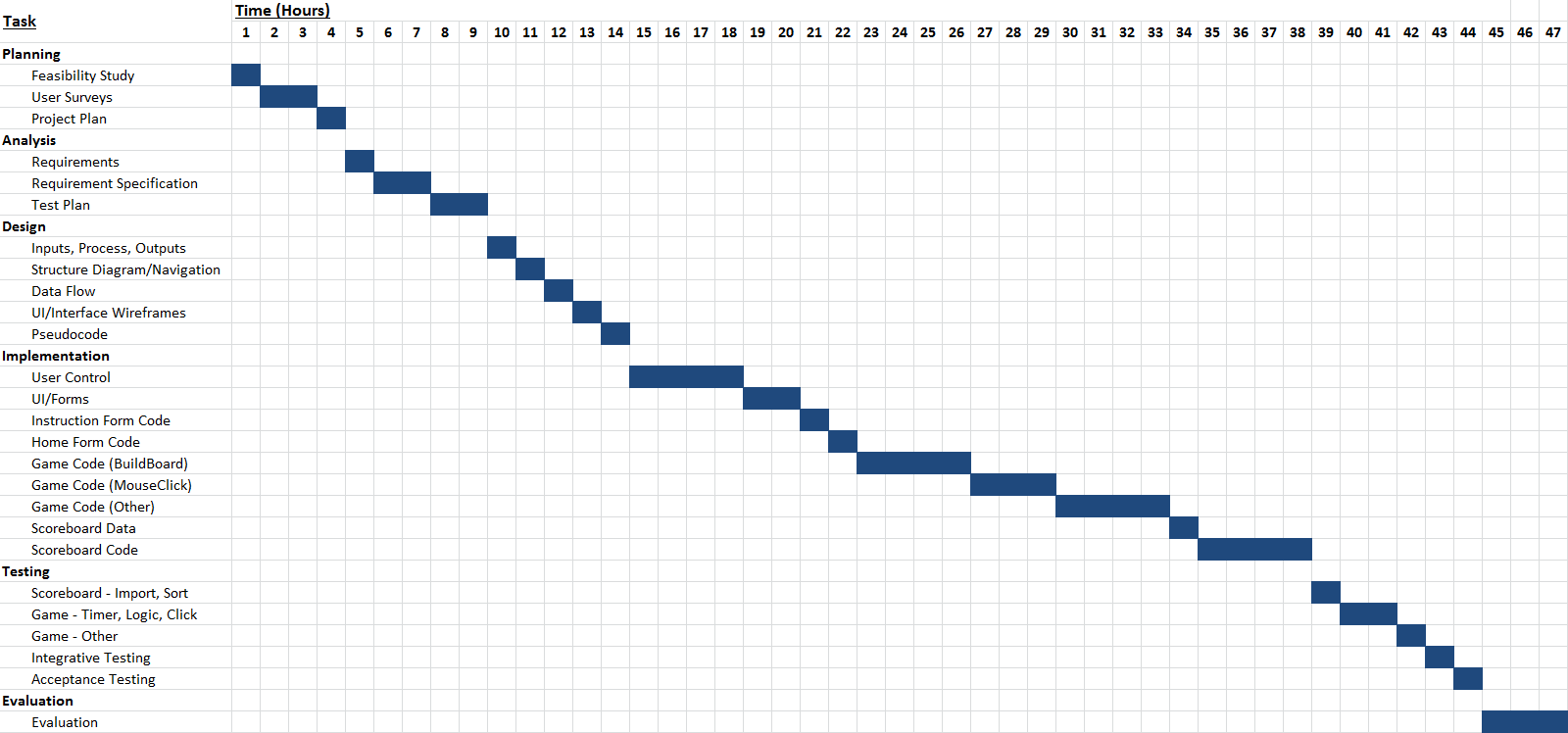
During the development of this project I will require several different human and technical resources. These resources will include:

* Desktop PC
* Visual Studio Community 2013
* Microsoft Excel
* Notepad
* Microsoft Word
* Internet Access for research
* Google forms
* End users
* Personal knowledge and skills

## Gantt Chart



## Gantt Chart – Revised Timings



Due to the implementation stage taking longer than expected due the iterations of development the Gantt chart ended up looking like the above. As you can see the implementation stage took two hours longer than expected.

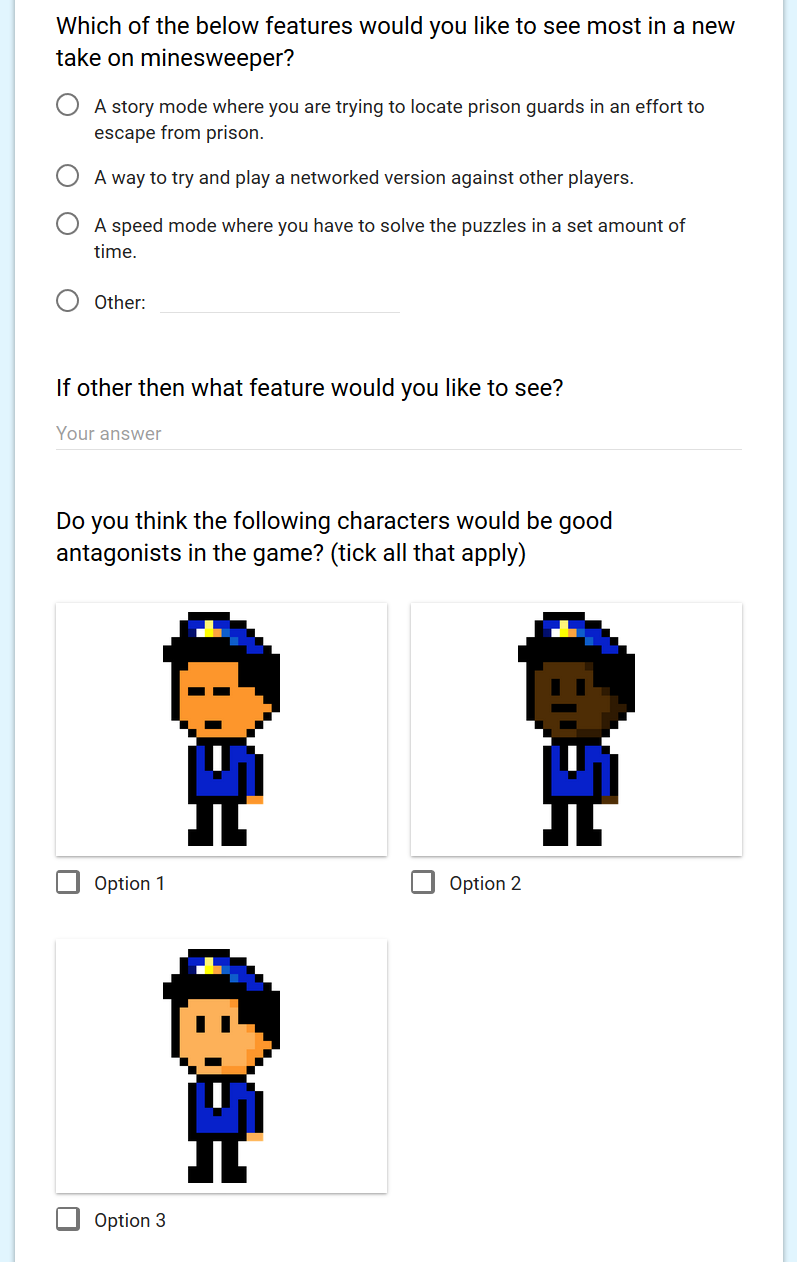
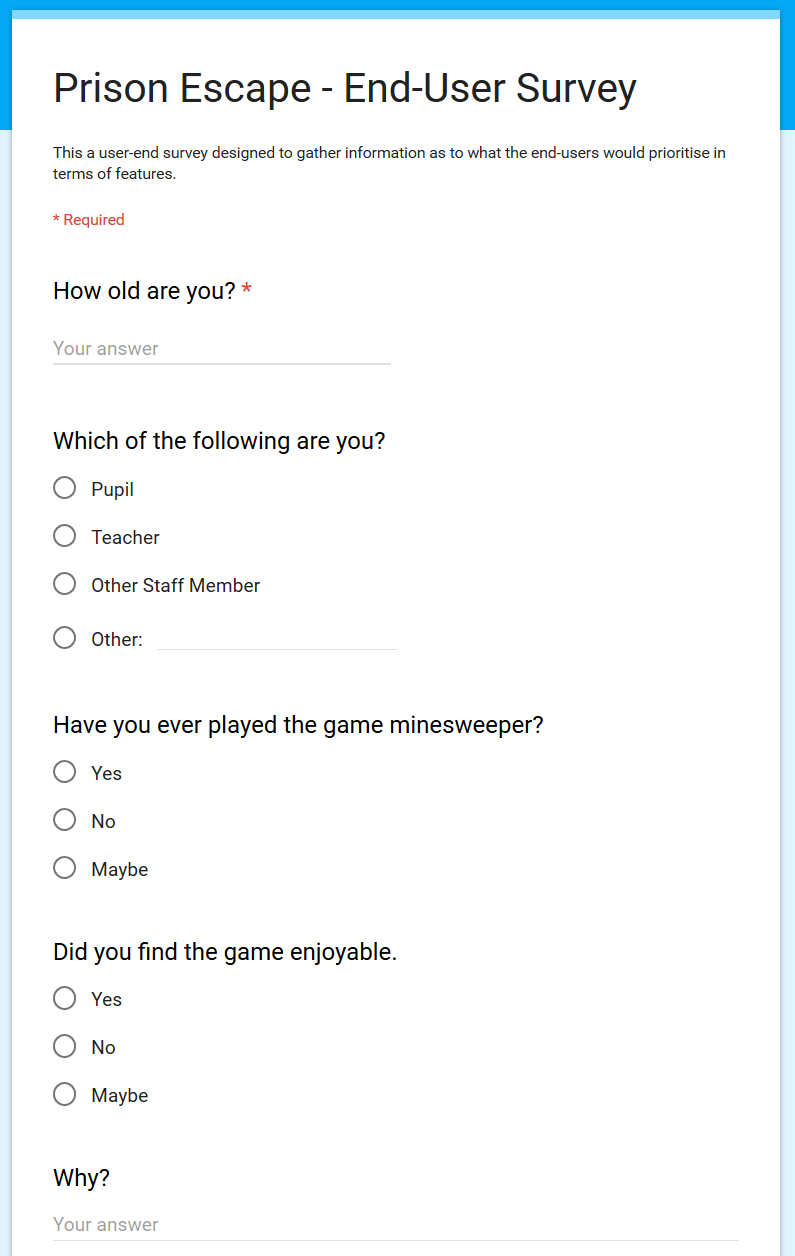
## Gantt Chart – Resources

|  |  |
| --- | --- |
| Task | Resources Required |
| Planning |  |
| Feasibility Study | Microsoft Word |
| User Surveys | Google Forms (end users) |
| Project Plan | Microsoft Word, Internet Access |
|  |  |
| Analysis |  |
| Requirements | Microsoft Word, Completed Surveys from Google Forms (end users) |
| Requirement Specification | Microsoft Word |
| Test Plan | Microsoft Word |
|  |  |
| Design |  |
| Inputs, Process, Outputs | Microsoft Word |
| Structure Diagram/Navigation | Microsoft Word |
| Data Flow | Microsoft Word |
| UI/Interface Wireframes | Microsoft Word |
| Pseudocode Design | Microsoft Word |
|  |  |
| Implementation |  |
| User Control | Microsoft Visual Studio Community 2013 |
| UI/Forms | Microsoft Visual Studio Community 2013 |
| Instruction Form Code | Microsoft Visual Studio Community 2013 |
| Home Form Code | Microsoft Visual Studio Community 2013 |
| Game Code (BuildBoard) | Microsoft Visual Studio Community 2013 |
| Game Code (MouseClick) | Microsoft Visual Studio Community 2013 |
| Game Code (Other) | Microsoft Visual Studio Community 2013 |
| Scoreboard Data | Microsoft Visual Studio Community 2013, Microsoft Excel, Notepad |
| Scoreboard Code | Microsoft Visual Studio Community 2013 |
|  |  |
| Testing |  |
| Scoreboard – Import, Sort | Microsoft Visual Studio Community 2013 |
| Game – Timer, Logic, Click | Microsoft Visual Studio Community 2013 |
| Game – Other | Microsoft Visual Studio Community 2013 |
| Integrative Testing |  |
| Acceptance Testing | End-users, Microsoft Visual Studio Community 2013 |
|  |  |
| Evaluation | Microsoft Word |

## End User Survey – Prison Escape

To collect data from end user I created to following survey using google forms. It allowed end users to suggest features. I also did some casual end-user interviews to further clarify what main features that end-users would like to see. I will then analyse the responses from the end-user interviews and the hard responses from the end-user surveys to try and create a game with enjoyable playability and no usability issues. These user responses will be crucial to deciding what features are and are not desired by end users meaning the created game should be something that is very enjoyable to play.

I am going to be creating a strategy game for people who enjoy games like minesweeper. I would like to create this using feedback from the people that would play this game.



## End User Survey – Analysis

This survey was completed by 18 students in a senior (fifth and sixth year) higher and advanced higher computing class. These would probably be typical end-users for this solution.

I analysed their responses and the responses from the casual interviews and the following is the result of that analysis. Their responses will be attached after the analysis.

**Design** – Current characters to be used throughout UI on various forms are appropriate – general appearance of the game is appropriate for use. The pixel art style is good for both in game and menu items. Clarity of layout is very important and having spoken to people that had filled in the survey easy, uncluttered navigation was a priority. Due to this the usability of the application as a whole should be sufficient for easy use.

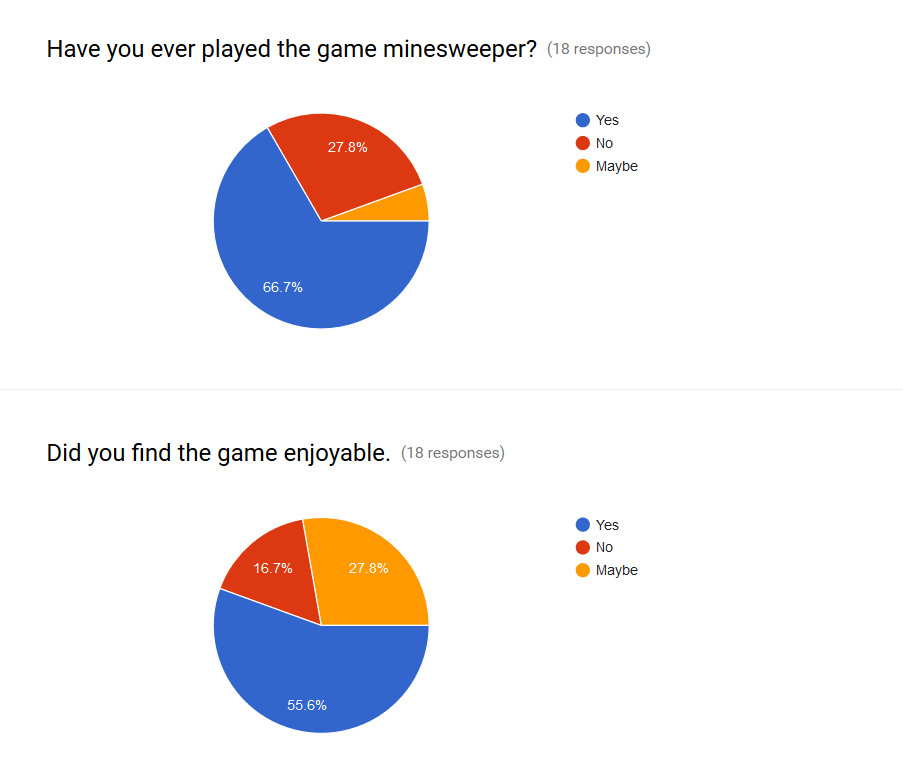
**Gameplay** – Most people had played minesweeper before and of those who had played it the clear majority enjoyed playing the game due to its logical and strategic nature something which will obviously remain in the project.

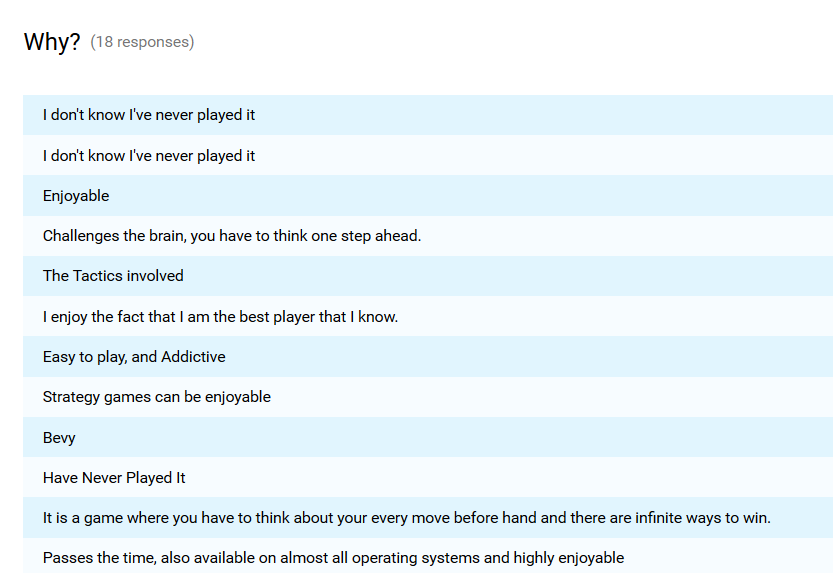
**Features** – The prison escape storyline was the most appealing out of all the optional features. The end users however did suggest some interesting other features such as a ranked system, the higher rank you get the more complex the games, and as you win you climb. You lose you drop. However due to the time restrictions I won’t be able to implement this idea. Another user suggested a versus mode where you play against another player with a choice of abilities to make things easier for you and harder for your opponent – this was another good idea, however, again due to the time restrictions this won’t be able to be implemented. Both ideas could be useful in future versions or refinements of the game.

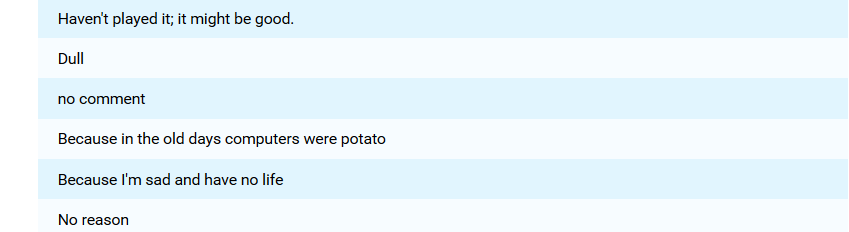
## End User Survey – Individual Responses

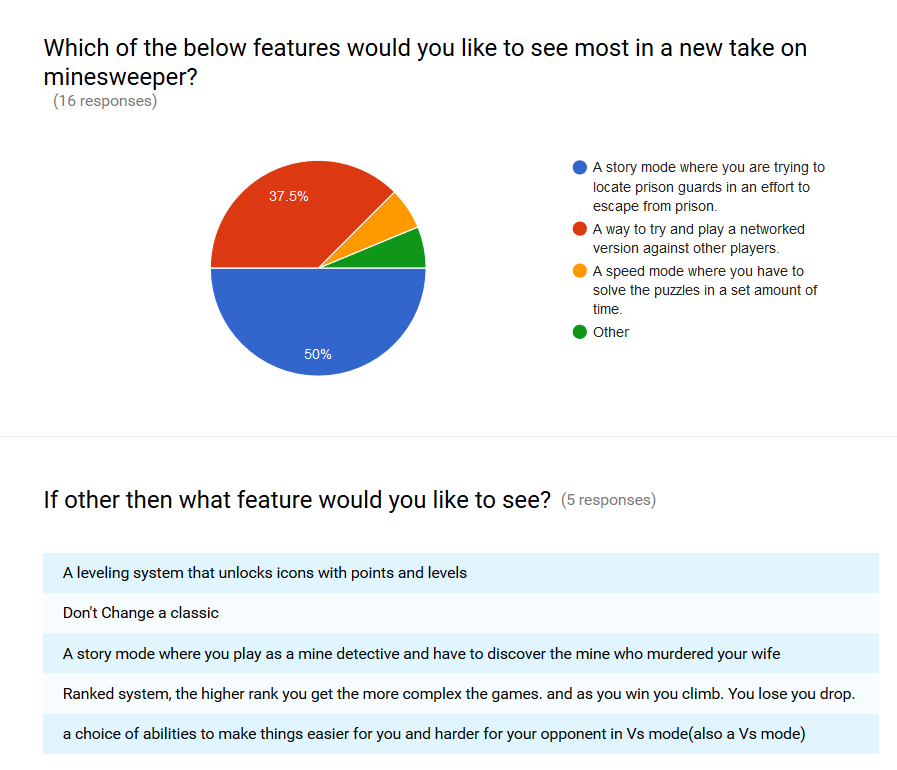
Due to the number of responses and size of the forms I have decided that I will not include individual responses but that I will use the automatically created collation of responses.

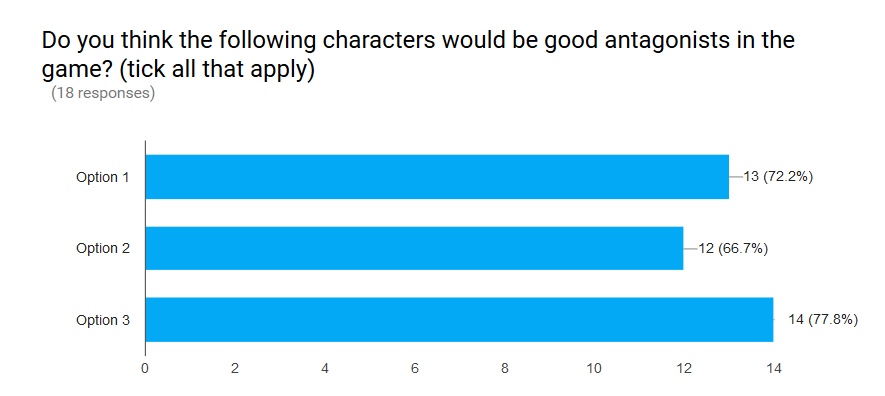
## Screen ClippingEnd User Survey – Collated Responses



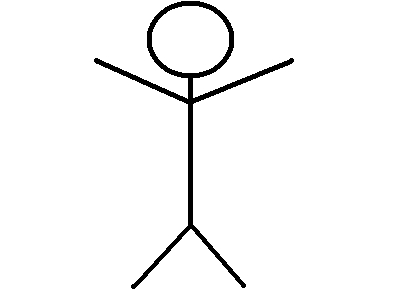
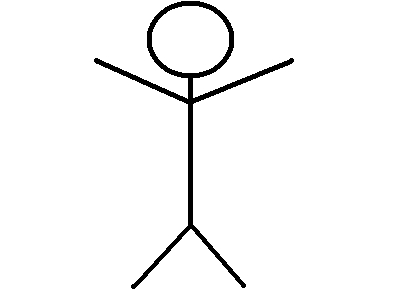








## Use Case Diagram



Scoreboard File

Player

# Analysis

## Initial Requirements – User

* Simple and Understandable UI
* Easy navigation between forms
* Completable game

## Initial Requirements – Functional

* Home screen
* Instruction screen
* High Score table
* Games of varying difficulty
* Return score
* Enter name for high score
* Display high score
* Consistent navigation between forms
* Button to restart finished game

## Initial Requirements – Inputs, Processes, Outputs

### Inputs

Left mouse click, right mouse click, player name, time, data from high score file

### Processes

Uncover cell, cycle through markers, clear empty cells, calculate time taken, read in times from file, sort times into order

### Outputs

Sorted high score table, uncovered cells, cleared empty cells, markers, high score table files

## Initial Requirements – Overview

The above requirements are subject to refinement and change throughout this project due to constraints and possible improvements that the solution will go through.

## Requirement Specification

### Purpose of Solution

The purpose of this solution is to make a fun game for people that enjoy strategy and logic games. It will include different difficulty settings, a clear and concise UI and will attempt to adhere to all the user requirements specified in responses to the end user survey.

### Scope

Internal: Gantt Chart, UI design, Test plan, Form design, Data flow, Pseudocode, Test plan, Requirements specification

External: Game with varying difficulty settings, ability to save score, sorted high score table

### Constraints

Time: Deadline of March 31st 2017 for submission of completed solution

External: Game will not have versus mode, game will only have three predefined difficulty modes

### End-Users

The end-user group of this project is people over the age of around ten. It’s most likely users are probably young adults that enjoy logic based puzzle games; however, there is scope for the application to be enjoyed by a larger group of users. The end-users will directly influence and contribute to my solution by completing user surveys which will be used to make final decisions on design and functionality

### Requirements

All requirements have been numbered so that they can be easily referenced when testing and throughout the rest of this document.

### User Requirements

1. Straightforward and Consistent navigation – On speaking with the end-users a key issue with games is the number of menus and settings. They wanted a streamlined game without a lot of different settings but enough for it not to get boring quickly. They also wanted to ensure that the form design was similar throughout the pregame menus and menus that all link together so that you can see the instructions whilst playing the game and check the high score table as well.
2. Playable Game – The end-users wanted a game that they would be able to complete so to ensure that almost everyone can play a game of the correct difficulty I have settled on choosing three different difficulty modes: beginner, intermediate and hard. Having three difficulties also means that there is an element of progression in the game users could start on the beginner boards and try and work their way up – this approach was suggested in the end-user surveys.
3. High Score table – A prominent idea when speaking with end-users was the ability to have some sort of high score or ranking system. Since this is a game with a set objective where the users either completes the game or doesn’t there was no way to include a points system – instead I plan on measuring the time it takes for users to complete the game and rank them so that the fastest score wins.

### Functional Requirements

1. Home screen – A clear opening screen with options to navigate to all the other possible screens that will be available in the solution. At this point in time this includes: start game, instructions and high-score table.
2. Functional Navigation – End-users wanted there to be an opportunity to navigate between the forms in a clear and easy manner. I thought this would be best achieved by creating a structure where from any one form you can access any other form respectively. They also suggested that they should be able to see the instructions and the high score table whilst they were playing the game so when navigating to these forms from the game screen, the game screen should remain open and running.
3. Instruction screen – A clear request from end-users was a clearly accessible instructions screen, they said it shouldn’t be in a drop down in-game menu, but that it should be clearly accessible from the home screen and be its own dedicated form.
4. Games of varying difficulty - The end users suggested a ranked system, the higher rank you get the more complex the games become. Due to time restraints I won’t have the option to create a game with many layers of difficulty but have decided the best way to include this idea is to create a game with three different difficulty settings – that way there is an element of progression in the game, which was very important to the end-users.
5. Return score – The end users were keen on a player vs. player system where they could play directly against other players. However, due to the restrictions of the game format I couldn’t find a feasible way to include this in my finished solution. I decided instead that users should be timed when trying to solve the puzzles, with the aim to try and solve the puzzle in the shortest possible time. The times will then be saved and returned by the program.
6. Enter name for high score – The users also suggested that they could put in their own nickname for the high score table each time they play. I suggested an account set up so that they can record and track their times but on the whole they thought that this was a lot of hassle for a game that is likely to be played for about 10 minutes at a time. This will be limited to 20 characters so that it can be properly displayed
7. High Score table – Due to the three difficulty settings there will be three different high score tables, one for each difficulty respectively. This means that there will be three individual text files.
8. Button to restart finished game – Another feature was the ability to quickly restart a game due to the nature of the puzzle game and how easy it is to make an error and lose the game. They suggested a clear button to restart the game.
9. Recursive Clear Algorithm to Clear Empty Boxes – The end-users mentioned that they didn’t want to have to clear loads of boxes without any warning lights near them so to counteract this a recursive clear algorithm will be created that clears all empty boxes in an area up to either the edge of the game board or a number.
10. Right Click To Flag Warning Lights – The end-users thought that a flagging mechanism that could mark the mines location may increase the usability and playability of the game.

### Inputs

1. Player name - Entering and storing names and times of players for the high score table
2. Text file – The text file containing the player names and times for the high score table
3. Left mouse click – Uncovering cells to reveal what lies underneath, either a number, nothing or a warning light
4. Right mouse click – Cycling through the different markers of a Cop badge to flag up warning areas, a question for when the user isn’t sure and nothing for if the user wants to remove a marker

### Outputs

1. Sorted High Score Tables – There will be three high score tables, one for each of the difficulties respectively. The will read data in from their respective high score file and sort the times into ascending order so that the shortest time will be at the top
2. Current Game Situation - A message telling users whether a game is in progress or whether they have escaped from prison or been caught trying to escape

## Test Plan

Throughout this project, I will use various types of testing to ensure that all parts and stages of this project are successfully completed and meet the earlier specified requirements – of both the end-users and the project.

The types of testing that I will use to ensure the successful completion of this project are:

* Component
* Integrative
* Acceptance
  + Alpha
  + Beta
* Final Testing

### Component Testing

Component testing will be used through this project to test individual parts of the solution – such as specific algorithms and procedures. It will be used to test every function and requirement of the completed game. Due to the fact that most of the solution builds upon a previous part I will test the components of the program as I go to ensure that I am not building a program on top of code that is incorrect or has bugs.

The modules to be tested are:

* Build Board - Code to make the board
* Mouse Click - Events triggered by mouse click
* Show Blanks - Recursive clear algorithm
* Timer - Timer to calculate game length
* Export Times - Procedure to export player details to file
* Import Data - Importing high score data
* Process Data - Processing high score data
* Display Data - Export high score data
* MineCell Views() - Coding for user control

### Integrative Testing

Integrative testing will be used to ensure that all procedures and files communicate correctly. Due to the fact that there are three external text files with the high score table data I will ensure that the reading and writing to these files is correct and in line with the visual basic data. It will also need to be used to ensure that all the links between forms are in working order and that the instructions and high scores can be viewed whilst a game is being played. Integrative testing will also be used to ensure that the game algorithms communicate with the user control in the right way so that the correct cells are being shown and that there is no issue with the numbers displayed on the cells.

The components that require integrative testing are:

* MineCell - To ensure the game form communicates correctly with the user control ‘MineCell’
* High Score Table - To ensure that the visual basic program communicates correctly with the external text file
* Game Play - To ensure that the procedures pass the correct data between them
* Game Form Links - To ensure that the scoreboard and instructions can be viewed whilst playing the game
* All Form Links - To ensure that all the forms pass data and communicate correctly

### Acceptance Testing

#### Alpha

Once all the component and integrative testing has been completed I will personally test the solution as a whole against all the features and requirements of the program. This will involve testing all form links, the high score tables and the gameplay and logic. I will ensure that I add any features that are not present and also that any requirements that have not been met are met.

#### Beta (Including Usability)

This will be carried out after the completion of the alpha testing. The end-users will have the ability to test out the program on their own systems and they will provide feedback on what they feel needs improved and what is good. I will collect this data using end-user surveys to get a grasp of what the strengths and the weaknesses of the solution are. Once this has been completed the solution will go through its final revision.

I will ask five different users to complete the beta testing. The users will be of different users three will be typical end-users and two will be end-users that are not typical end-users of the solution. These non-typical users will test the game to ensure that it can be played, and understood by people who would not typically play strategy game. Having different types of users testing the program will ensure that game is playable and accessible by almost anyone that wants to play the game.

To test the usability, I will give the testers a series of tasks to complete and then the testers will comment on how straightforward they found the task to be. Comments will also be made on efficiency, reliability, robustness and functionality of the program. I will create the tasks for users to complete after the final stages of internal testing have been completed.

I expect at this final stage for all requirements to be met and for the game to be fully functional.

### Final Testing

After all final the revisions have been made I will do a final test of the game to ensure that all the beta test information has been listened to in order to make sure that all requirements have been met. I will use breakpoints and various other techniques to ensure that the data is being passed correctly.

### Test Plan for Final Testing

|  |  |  |
| --- | --- | --- |
| Requirements | Tasks | Expected Outputs |
| Navigation (1, 5) | 1. Test all links | 1. Users will be able to navigate through application successfully |
| High Scores Input ( 8, 14, 18) | 1. Test data types that are not just integers when reading from file 2. Return score of zero and scores that are very large 3. Delete random entries in text file 4. Ensure that data is kept in the same record | 1. Should sort incorrectly due to the sort procedure sorting numbers not strings 2. Program should run correctly 3. Program may crash due to empty data trying to be assigned to records 4. Data should be kept together |
| High Scores Display (3, 10, 15) | 1. Ensure that regardless of entered times or number of entered times the sort is in ascending order | 1. Sort will be in ascending order |
| Input Validation (9) | 1. Test string of length greater than 20 when inputting name 2. Test different variable types other than strings | 1. An error message should appear asking the user to input a valid string – this should repeat until a valid string is entered 2. The file should be exported correctly and no error message should be displayed – this should also have no effect on the high-score table |
| Game (2, 7, 11, 12, 19) | 1. Try clicking random squares – including squares that have already been clicked after a game has ended 2. Ensure that all lights remain covered unless clicked on or marked – make sure no lights are uncovered by recursive clear algorithm 3. Test restart game button at various points – ensure it restarts a game of the same difficulty 4. Right click to flag mines – try flagging squares that have numbers on them or that have been clicked on and make sure that the warning light count decreases regardless of the value of the cell underneath | 1. Nothing should happen as the game should be suspended after the game is finished with the user needing to restart the game 2. All lights remain covered unless clicked on – empty squares should be empty and other squares should also have the correct values stored in them 3. A new game of the same difficulty should start with the timer beginning after the first cell has been clicked on 4. The cells should be flagged correctly and the user shouldn’t be able to flag cells that have already been click as the code states that only cells with view as buttons can be clicked |
| Gameplay (16, 17) | 1. Right mouse click – Cycling through the different markers of a Cop badge to flag up warning areas, a question for when the user isn’t sure and nothing for if the user wants to remove a marker 2. Left mouse click – Uncovering cells to reveal what lies underneath, either a number, nothing or a warning light | 1. The cells markers should be cycled through correctly, in the same order with it only affecting the remaining mine count if the flag is a cop shield and not a question mark 2. Only cells which have not been uncovered are clickable with uncovered cells not reacting to any user interaction |
| Instructions (6) | 1. Instruction screen – Ensure that it is accessible from every in game screen and out of game screen | 1. The instructions screen should be accessible from all of these. |
| Home Screen (4) | 1. The home screen – check that the home screen can be accessed from every form that comprises the game – make sure that form is readable and clear | 1. The home screen should be accessible from every form and the home screen itself should be clear, understandable and have good general usability. |

# Design

## Inputs, Processes, Outputs

### Inputs

* Left mouse click - To uncover a cell. Three possibilities: mine, number or blank
* Right mouse click - To cycle through the markers: cop shield, question mark or nothing
* Player name and time - To add to the high score file
* Data from high score file - To create and sort the times into ascending order

### Processes

Uncover cell - Inputs Required: Left Mouse Click

Cycle through markers - Inputs Required: Right Mouse Click

Clear empty cells - Inputs Required: Left Mouse Click

Calculate time taken - Inputs Required: Time from timer

Read in times from file - Inputs Required: High Score data file

Sort times into order - Inputs Required: Read in data from file

Calculate number of remaining Mines - Inputs required: Right Mouse Click, Number of Mines

### Outputs

* Uncovered cells - Once a cell has been uncovered it will become either a number, empty or a warning light
* Cleared empty cells - If a cell has no mines adjacent all blank cells up to a cell with something it should be cleared
* Markers - Used by the players to mark the locations where they think the mines are
* Number of Remaining Lights - Number of total lights – number of marked lights
* High score table files - Outputs the data required to create high score table
* Sorted high score table - Using the data from the file a sorted high score table will be created

## Navigational Structure – Initial

Game

Home

Instructions

Scoreboard

## Navigational Structure – Updated

(Pop Up)

(Pop Up)

Scoreboard

Game

Instructions

Home

## Structure Diagram

### Game

Prison Escape Game

Left Mouseclick byVal

Right Mouseclick byVal

Export Results

BoardCols byVal

Player as PE\_Player byVal

boardCols ByVal

boardRows ByVal

mines byVal

Gameover

StartGame

Mineclick

DoLabels

BuildBoard

M as Minecell byVal

ShowBlanks

### Scoreboard

Scoreboard

ByRef BPlayer() as PE\_Player

ByRef x As Integer

ByRef BPlayer()

ByRef EPlayer()

as PE\_Player

ByRef EPlayer()

as PE\_Player

ByVal z As Integer

ByRef EPlayer()

as PE\_Player

ByRef z As Integer

ByRef IPlayer() as PE\_Player

ByRef y As Integer

ByRef BPlayer()

ByRef x As Integer

ByRef IPlayer() as PE\_Player

ByVal y As Integer

ByRef BPlayer()

ByVal x As Integer

ByRef IPlayer()

as PE\_Player

BInput

EInput

IInput

Process

ByRef EPlayer()

ByVal z As Integer

ByRef IPlayer()

ByVal y As Integer

EOutput

IOutput

BOutput

## Data Flow

### Game

BuildBoard(**IN:** ByVal boardCols As Integer, ByVal boardRows As Integer, ByVal minecount As Integer)

MineClick(**IN:** ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs)

ShowBlanks(**IN:** ByVal M as Minecell)

ExportResults(**IN:** ByVal player As PE\_Player, ByVal BoardCols As Integer)

### Scoreboard

BInputs (**OUT:** ByRef BPlayer() As PE\_Player, ByRef x As Integer)

BProcess(**IN:** ByVal x As Integer **OUT:** ByRef BPlayer() As PePlayer)

BOutput(**IN:** ByVal x As Integer, ByRef BPlayer() As PePlayer)

IInputs (**OUT:** ByRef IPlayer() As PE\_Player, ByRef y As Integer)

IProcess(**IN:** ByVal y As Integer **OUT:** ByRef IPlayer() As PePlayer)

IOutput(**IN:** ByVal y As Integer, ByRef IPlayer() As PePlayer)

EInputs (**OUT:** ByRef EPlayer() As PE\_Player, ByRef z As Integer)

EProcess(**IN:** ByVal z As Integer **OUT:** ByRef EPlayer() As PePlayer)

EOutput(**IN:** ByVal z As Integer, ByRef EPlayer() As PePlayer)

### Other

All other procedures are going to be open so that they can access any data in any other part of the program. This will take up more memory and be slightly less efficient than if specific data was being passed, but due to the small amount of data that will be accessible this should make no measurable difference to the user experience – it will also reduce the chance for error and bugs in the code meaning that the overall game feels more usable as a whole.

## Wireframes

### Home

Prisoner Graphic

Title Graphic

Prison Escape

Button

Button

Button

Police Officer Graphic

Score Board

Instructions

Start Game

Instructions (Graphic)

### Instructions

Button

Button

Police Officer Graphic

Start Game

Home

### Scoreboard

Labels

Labels

Button

Listboxes

Button

Listboxes

Home

Score

Board

How

To

Play

Time

Time

Time

Name

Name

Name

Pos.

Pos.

Pos.

Advanced

Intermediate

Beginner

Listboxes

### Game Board

Drop down menu

Button

Label

Panel

(will be resized to fit different sized boards)

Table

Button

Button

Button

Label

Label

Game

Warning Light Count

Timer

Instructions

Home

Scoreboard

Message

### MineCell (User Control)

All text for the different views will be in font ‘Arial’. Button design is template for question and cop shield.

#### Warning Light

Rectangle – Rounded top

Exclamation Mark

!

Circle

#### Cop Shield

“COP” text

COP

Rectangle

Circle

#### Button

Grey fill

Dark Grey bottom and right border

White top and left border

#### Question Mark

?

Question

Different colours are used to increase readability

#### Numbers

8

7

6

5

4

3

2

1

## Pseudocode

### Home

1. Start Game Button Click
2. Instructions Button Click
3. Scoreboard Button Click
   1. Open start game form
   2. Close home form
   3. Open instructions form
   4. Close home form
   5. Open scoreboard form
   6. Close home form

### Instructions

1. Start Game Button Click
2. Home Button Click
   1. Open start game form
   2. Close instructions form
   3. Open home form
   4. Close instructions form

### Scoreboard

1. Form – Opened
2. bInputs
3. bOutput
4. iInputs
5. iOutput
6. eInputs
7. eExport
8. process
   1. Define STRUCURE pe\_player
      1. Position as INTEGER
      2. playerName as STRING
      3. playerTime as INTEGER
      4. difficulty as CHAR
   2. END STRUCTURE
   3. SET Bplayer(999999) As pe\_player
   4. SET Iplayer(999999) As pe\_player
   5. SET Eplayer(999999) As pe\_player
   6. SET x as INTEGER = 0
   7. SET y as INTEGER = 0
   8. SET z as INTEGER = 0
   9. Call bInput(Bplayer, x)
   10. Call Process(Bplayer, x)
   11. Call bOutput(Bplayer, x)
   12. Call IInput(Iplayer, y)
   13. Call Process(Iplayer, y)
   14. Call IOutput(Iplayer, y)
   15. Call EInput(Eplayer, z)
   16. Call Process(Eplayer, z)
   17. Call EOutput(Eplayer, z)
   18. OPEN beginner high score file
   19. READ length of file
   20. SET x TO length of file
   21. FOR i FROM 1 TO x
   22. INPUT data as player( i)
   23. SET player(i).position TO i
   24. NEXT
   25. CLOSE FILE

3.1 FOR i = 1 TO x

3.2 ADD player(i).position TO bpositionlistbox

3.3 ADD player(i).playername TO bnamelistbox

3.4 ADD player(i).position TO btimelistbox

* 1. OPEN intermediate high score file
  2. READ length of file
  3. SET x TO length of file
  4. FOR i FROM 1 TO x
  5. INPUT data as player( i)
  6. SET player(i).position TO i
  7. NEXT
  8. CLOSE FILE

1. FOR i = 1 TO x
2. ADD player(i).position TO ipositionlistbox
3. ADD player(i).playername TO inamelistbox
4. ADD player(i).position TO itimelistbox
   1. OPEN intermediate high score file
   2. READ length of file
   3. SET x TO length of file
   4. FOR i FROM 1 TO x
   5. INPUT data as player( i)
   6. SET player(i).position TO i
   7. NEXT
   8. CLOSE FILE
5. FOR i = 1 TO x
6. ADD player(i).position TO ipositionlistbox
7. ADD player(i).playername TO inamelistbox
8. ADD player(i).position TO itimelistbox
   1. FOR i FROM 1 to (x-1)
   2. FOR j = (i+1) TO x
   3. IF player(i).playerTime > player(j).playerTime THEN
   4. SWAP player(i) AND player(j)
   5. END IF
   6. NEXT
   7. NEXT
   8. FOR k = 1 to x
   9. SET player(k).position TO k
   10. NEXT

### MineCell

### Draw Button

### Draw Cop Shield

### Draw Light

### Draw Numbers

### Draw Question Mark

### 1.1 SET cellcolour TO GREY

### 1.2 SET bottomBorder TO DARKGREY

### 1.3 SET rightBorder TO DARKGREY

### 1.4 SET leftBorder TO WHITE

### 1.5 SET topBorder TO WHITE

### Draw Button()

### 2.2 SET circle TO YELLOW CRect(-0.6, -0.6, 1.2, 1.2)

2.3 SET rect TO YELLOW TRect(-0.6, -0.7, 1.2, 0.7)

2.4 DRAWSTRING “COP” (font = Arial, colour = black, X=0, Y+0.3)

3.1 DRAWCIRCLE(colour = red, x=0, y=0, rad = 1)

3.2 SET shine AS RRect(colour = yellow)

3.3 FOR i FROM 1 to 8

3.4 DRAW shine AT 45° intervals

3.5 NEXT

3.6 DRAWSTRING “!” (colour=white, x = 0, y = 0)

* 1. IMPORT adjacentMines FROM GAME
  2. SELECT CASE adjacentMines
  3. CASE 1
  4. DRAWSTRING “1” (font = Arial, colour = blue, x = 0, y = 0)
  5. CASE 2
  6. DRAWSTRING “2” (font = Arial, colour = green, x = 0, y = 0)
  7. CASE 3
  8. DRAWSTRING “3” (font = Arial, colour = red, x = 0, y = 0)
  9. CASE 4
  10. DRAWSTRING “4” (font = Arial, colour = navy, x = 0, y = 0)
  11. CASE 5
  12. DRAWSTRING “5” (font = Arial, colour = dark green, x = 0, y = 0)
  13. CASE 6
  14. DRAWSTRING “6” (font = Arial, colour = dark red, x = 0, y = 0)
  15. CASE 7
  16. DRAWSTRING “7” (font = Arial, colour = brown, x = 0, y = 0)
  17. CASE 8
  18. DRAWSTRING “8” (font = Arial, colour = black, x = 0, y = 0)
  19. END SELECT

5.1 Draw Button()

5.2 DRAWSTRING “?” (colour = black, font = Arial, x = 0, y = 0)

### Game

1. Initialise
2. BuildBoard
3. startGame
4. MineClick
5. Showblanks
6. Timer
7. DoLabels
8. gameOver
9. Difficulty
10. export\_times
11. homeclick
12. instructionsclick
13. scoreboardclick
    1. SET cellSize As INTEGER = 40
    2. SET boardRows As INTEGER = 9
    3. SET boardCols As INTEGER = 9
    4. SET mineCount As INTEGER = 11
    5. SET seconds As INTEGER = 0
    6. SET markedMines As INTEGER = 0
    7. SET finished As BOOLEAN = FALSE
    8. SET minefield(9, 9) As MineCell
    9. Define STRUCURE pe\_player
       1. Position as INTEGER
       2. playerName as STRING
       3. playerTime as INTEGER
       4. difficulty as CHAR
    10. END STRUCTURE
    11. SET player As pe\_player
    12. BuildBoard()
    13. SET finished TO FALSE
    14. SET seconds TO 0
    15. SET markedMines TO 0
    16. DoLabels()
    17. DISABLE timer
    18. CLEAR lblmessage
    19. REDFINE mineField As (boardRows – 1, boardCols - 1)
    20. FOR rows FROM 0 TO boardRows – 1
    21. FOR cols FROM 0 TO BoardCols – 1
    22. SET c TO NEW MineCell
    23. ADD c TO panel
    24. SET hasmine TO FALSE
    25. SET c.x TO cols
    26. SET c.y TO rows
    27. SET mineField(rows, cols) TO c
    28. NEXT
    29. NEXT
    30. SET RX As NEW RANDOM
    31. FOR i FROM 1 to mineCount
    32. SET x TO 0
    33. SET y TO 0
    34. DO LOOP
    35. x = RX.Next(0, boardCols)
    36. y = RX.Next(0, boardRows)
    37. LOOP UNTIL mineField(y, x) hasmine
    38. SET mineField(Y, X).hasmine TO TRUE
    39. FOR rows FROM 0 to boardRows -1
    40. FOR cols FROM 0 to boardCols -1
    41. IF mineField(rows, cols) DOES NOT have mine
    42. FOR R FROM row-1 TO row+1
    43. FOR C FROM col-1 TO col+1
    44. IF R>=0 AND R < boardRows AND C>=0 AND C boardCols THEN
    45. IF mineField(R, C) Has mine THEN
    46. Minefield(rows,cols).number = minefield(rows,cols) + 1
    47. END IF
    48. END IF
    49. NEXT
    50. NEXT
    51. END IF
    52. NEXT
    53. NEXT
    54. RESIZE panelWidth TO boardCols
    55. RESIZE panelHeight TO boardRows

3.1 IMPORT Rows, Cols, Mines

3.2 SET boardRows TO Rows

3.3 SET boardCols TO Cols

3.4 SET mineCount TO Mines

3.5 BuildBoard()

* 1. IF finished THEN END SUB
  2. SET M As MineCell TO SENDER
  3. IF leftmouse CLICKED THEN
  4. IF cell has mine THEN
  5. FOR each MC As MineCell in mineField
  6. If MC has mine then MC.View = Mine
  7. NEXT
  8. STOP timer
  9. Lblmessage = “caught”
  10. SET finished TO TRUE
  11. ELSEIF M.Number > 0 THEN
  12. M.View = Number
  13. IF gameover() THEN
  14. STOP timer
  15. Lblmessage = “escaped”
  16. SET finished TO TRUE
  17. Export\_times(player, boardCols)
  18. ENDIF
  19. ELSEIF
  20. M.number = 0 THEN
  21. ShowBlank(M)
  22. IF gameover() THEN
  23. STOP timer
  24. Lblmessage = “escaped”
  25. SET finished TO TRUE
  26. Export\_times(player, boardCols)
  27. END IF
  28. END IF
  29. IF rightmouse CLICKED THEN
  30. SELECT CASE m.view
  31. CASE button
  32. M.view = flag
  33. markedMines = markedMines + 1
  34. DoLabels()
  35. CASE flag
  36. M.view = Question
  37. markedMines = markedMines – 1
  38. DoLabels()
  39. END SELECT
  40. END IF

5.1 M.View = Number

5.2 FOR R As INTEGER = M.y – 1 TO M.y + 1

5.3 FOR C As INTEGER = M.x -1 To M.x + 1

5.4 IF cell not off board THEN

5.5 SET MC As MineCell = mineField(R, C)

5.6 IF MC.View = button THEN

5.7 IF MC.Number = 0 THEN

5.8 ShowBlanks(MC)  
5.9 ELSE

5.10 MC.view = numer

5.11 END IF

5.12 END IF

5.13 END IF

5.14 NEXT

5.15 NEXT

6.1 seconds = seconds + 1

6.2 DoLabels()

7.1 lblClock = seconds

7.2 lblMines = (minecount – markedmines)

8.1 SET tv As BOOLEAN TO TRUE

8.2 FOR EACH mc As MineCell IN mineField

8.3 IF mc DOES NOT HAVE MINE AND mc IS NOT A NUMBER THEN

8.4 SET tv TO FALSE

8.5 END IF

8.6 NEXT

8.7 RETURN TV

9.1 SELECT CASE difficulty

9.2 CASE beginner

9.3 startGame(9, 9, 11)

9.4 CASE intermediate

9.5 startGame(16, 16, 40)

9.6 CASE expert

9.7 startGame(16, 30, 99)

9.8 END SELECT

10.1 SELECT CASE difficulty

10.2 CASE beginner

* 1. SET player.difficulty TO beginner
  2. CASE intermediate
  3. SET player.difficulty TO intermediate
  4. CASE expert

10.7 SET player.difficulty TO expert

10.8 END SELECT

10.9 SET player.playerName TO INPUT FROM KEYBOARD

10.10 IF beginner THEN

10.11 ADD player TO beginner file

10.12 ELSE IF intermediate

10.13 ADD player TO intermediate file

10.14 ELSE IF expert

10.15 ADD player TO expert file

10.16 END SELECT

11.1 OPEN home

11.2 CLOSE me

12.1 OPEN instructions

13.1 OPEN scoreboard

# Implementation

## Home Form

### Design



### Running Form



### Code

Public Class Form1

Private Sub btnHow2Play\_Click(sender As Object, e As EventArgs) Handles btnHow2Play.Click

'Coding to go to instructions

Me.Hide() 'closes this form

My.Forms.Instructions.Show() 'shows instructions

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

My.Forms.scoreboard.Show() 'shows scoreboard

Me.Hide() 'closes this form

End Sub

Private Sub btnStart\_Click(sender As Object, e As EventArgs) Handles btnStart.Click

Me.Hide() 'closes this form

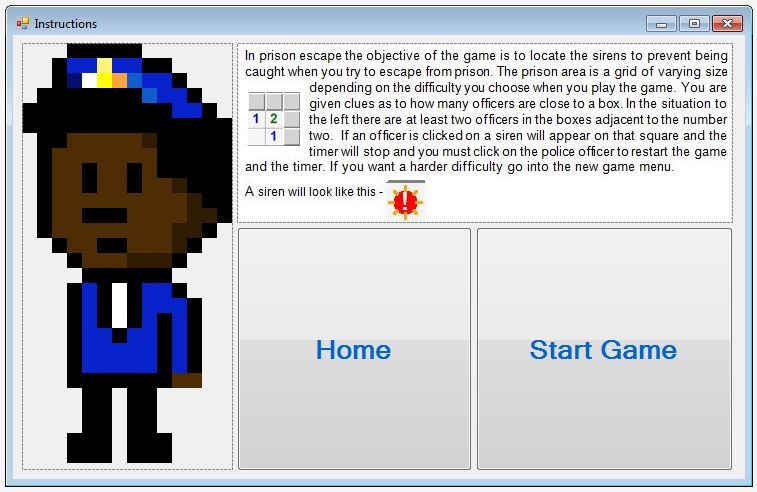
My.Forms.Start.Show() 'opens game form

End Sub

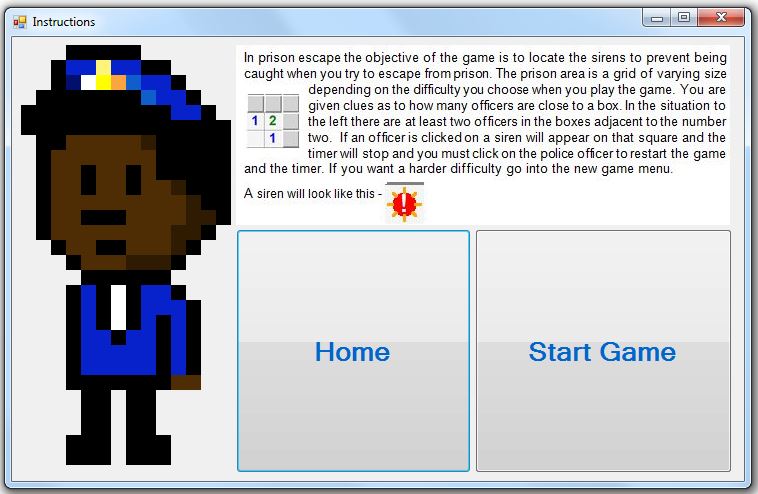
End Class

## Instructions Form

### Design



### Running Form



### Code

Public Class Instructions

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

Me.Close() 'closes this form

My.Forms.Form1.Show() 'opens home

End Sub

Private Sub Button2\_Click(sender As Object, e As EventArgs) Handles Button2.Click

Me.Close() 'closes this form

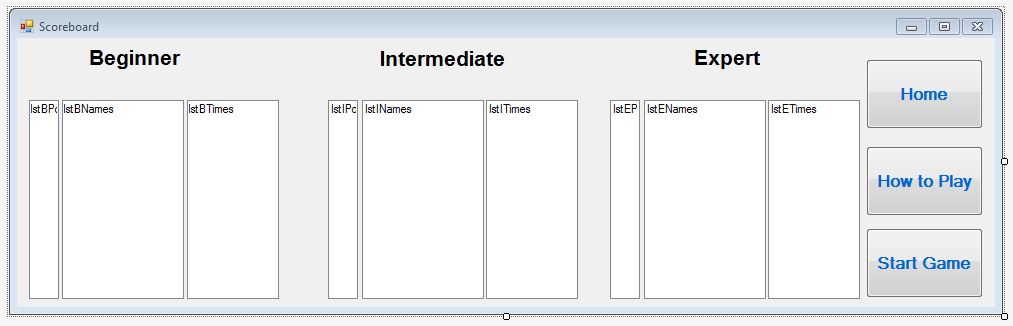
My.Forms.Start.Show() 'opens new game

End Sub

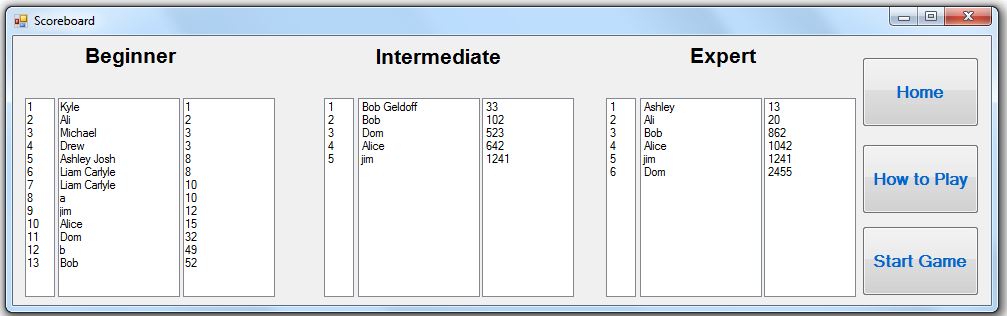
End Class

## Scoreboard

### Design



### Running Form



### Code

Imports System

Imports System.IO

Imports System.Text

Public Class scoreboard

Structure pe\_player 'creates structure

Dim position As Integer 'defines position as integer

Dim playerName As String 'defines player name as string

Dim playerTime As Integer 'defines player time as integer

Dim difficulty As Char 'defines difficulty as char

End Structure

Private Sub scoreboard\_Load(sender As Object, e As EventArgs) Handles Me.Load

Dim Bplayer(9999999) As pe\_player 'defines Bplayer as a high index array of structure pe\_player

Dim Iplayer(9999999) As pe\_player 'defines Iplayer as a high index array of structure pe\_player

Dim Eplayer(9999999) As pe\_player 'defines Eplayer as a high index array of structure pe\_player

Dim x As Integer = 0 'defines x as integer to be used for the file length

Dim y As Integer = 0 'defines y as integer to be used for the file length

Dim z As Integer = 0 'defines z as integer to be used for the file length

BeginnerInput(Bplayer, x) 'Calls subprocedure that inputs beginner high scores

Process(Bplayer, x) 'Calls subprocedure that processes and sorts beginner high scores

BeginnerOutput(Bplayer, x) 'Calls subprocedure that outputs beginner high scores

IntermediateInput(Iplayer, y) 'Calls subprocedure that inputs intermediate high scores

Process(Iplayer, y) 'Calls subprocedure that processes and sorts intermediate high scores

IntermediateOutput(Iplayer, y) 'Calls subprocedure that outputs intermediate high scores

ExpertInput(Eplayer, z) 'Calls subprocedure that inputs expert high score

Process(Eplayer, z) 'Calls subprocedure that processes and sorts expert high scores

ExpertOutput(Eplayer, z) 'Calls subprocedure that outputs expert high scores

End Sub

Private Sub BeginnerInput(ByRef player() As pe\_player, ByRef x As Integer)

Dim path As String = "Z:\S6\Computing Science\PROJECT\Project\BeginnerLeaderboard.csv" 'Sets path to be the file path

Dim leaderboard As StreamReader = New StreamReader(path) 'defines local variable leaderboard as a newstreamreader

Do While leaderboard.Peek() > -1 'loop until end of file

leaderboard.ReadLine() 'read file line

x += 1 'increments value of x

Loop

leaderboard.Close() 'closes leaderboard

FileOpen(1, path, OpenMode.Input) 'opens file with predefined file path

For i = 1 To x 'loop until end of file

Input(1, player(i).playerName) 'inputs player name

Input(1, player(i).playerTime) 'inputs player time

player(i).position = i 'sets position to be the index

Next

FileClose()

End Sub

Private Sub BeginnerOutput(ByRef player() As pe\_player, ByVal x As Integer)

For i = 1 To x

lstBPositions.Items.Add(player(i).position) 'adds position to list box

lstBNames.Items.Add(player(i).playerName) 'adds name to list box

lstBTimes.Items.Add(player(i).playerTime) 'adds time to list box

Next

End Sub

Private Sub IntermediateInput(ByRef player() As pe\_player, ByRef x As Integer)

Dim path As String = "Z:\S6\Computing Science\PROJECT\Project\IntermediateLeaderboard.csv" 'Sets path to be the file path

Dim leaderboard As StreamReader = New StreamReader(path) 'defines local variable leaderboard as a newstreamreader

Do While leaderboard.Peek() > -1 'loop until end of file

leaderboard.ReadLine() 'read file line

x += 1 'increments x

Loop

leaderboard.Close() 'closes file

FileOpen(1, path, OpenMode.Input) 'opens file with path predefined

For i = 1 To x 'loop until file end

Input(1, player(i).playerName) 'input player name

Input(1, player(i).playerTime) 'input player time

player(i).position = i 'set player position to index

Next

FileClose() 'close file

End Sub

Private Sub IntermediateOutput(ByRef player() As pe\_player, ByVal x As Integer)

For i = 1 To x

lstIPositions.Items.Add(player(i).position) 'adds position to list box

lstINames.Items.Add(player(i).playerName) 'adds name to list box

lstITimes.Items.Add(player(i).playerTime) 'adds time to list box

Next

End Sub

Private Sub ExpertInput(ByRef player() As pe\_player, ByRef x As Integer)

Dim path As String = "Z:\S6\Computing Science\PROJECT\Project\ExpertLeaderboard.csv" 'Sets path to be the file path

Dim leaderboard As StreamReader = New StreamReader(path) 'defines local variable leaderboard as a newstreamreader

Do While leaderboard.Peek() > -1 'loop until end of file

leaderboard.ReadLine() 'read file line

x += 1 'increments x

Loop

leaderboard.Close() 'closes file

FileOpen(1, path, OpenMode.Input) 'opens file with path predefined

For i = 1 To x 'loops until end of file

Input(1, player(i).playerName) 'inputs player name

Input(1, player(i).playerTime) 'inputs player time

player(i).position = i 'sets player position to be index

Next

FileClose() 'closes file

End Sub

Private Sub ExpertOutput(ByRef player() As pe\_player, ByVal x As Integer)

For i = 1 To x

lstEPositions.Items.Add(player(i).position) 'adds position to list box

lstENames.Items.Add(player(i).playerName) 'adds name to list box

lstETimes.Items.Add(player(i).playerTime) 'adds time to list box

Next

End Sub

Private Sub Process(ByRef player() As pe\_player, ByVal x As Integer)

Dim i As Integer

Dim j As Integer

Dim temp As Object

For i = 0 To x - 1 'loops through every index

For j = i + 1 To x 'loops until index is at the end of file

If player(i).playerTime > player(j).playerTime Then 'if the player1time>player2time then swap

temp = player(i)

player(i) = player(j)

player(j) = temp

End If

Next

Next

Dim k As Integer

For k = 1 To x

player(k).position = k 'sets position to be index

Next

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

My.Forms.Form1.Show()

Me.Close()

End Sub

Private Sub btnHow2Play\_Click(sender As Object, e As EventArgs) Handles btnHow2Play.Click

My.Forms.Instructions.Show()

Me.Close()

End Sub

Private Sub btnStart\_Click(sender As Object, e As EventArgs) Handles btnStart.Click

My.Forms.Start.Show()

Me.Close()

End Sub

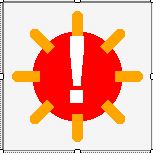
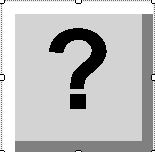
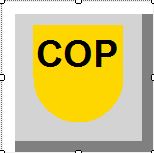
## End Class

## Minecell (UserControl)

### Design

Blank Cell that will be used as a button

Shield to mark the warning light locations



*Figure 1* – Number One

*Figure 2* – Number Two

Warning Light

Question mark to mark the locations where a user is not sure about the contents of a square

*Figure 2*

*Figure 1*

### Code

Public Class MineCell

'Enum for MineCell Views

Public Enum MineCellView

Button

Mine

Number

Flag

Question

End Enum

'Field Values

Private mView As MineCellView

Private mNumber As Integer

Private mHasMine As Boolean

Private myX As Integer

Private myY As Integer

Private mButtonColour As Color = Color.LightGray

'Property Interfaces

Public Property ButtonColour() As Color

Get

Return mButtonColour

End Get

Set(value As Color)

mButtonColour = value

End Set

End Property

Public Property HasMine() As Boolean

Get

Return mHasMine

End Get

Set(ByVal value As Boolean)

mHasMine = value

End Set

End Property

Public Property x() As Integer

Get

Return myX

End Get

Set(ByVal value As Integer)

myX = value

End Set

End Property

Public Property y() As Integer

Get

Return myY

End Get

Set(ByVal value As Integer)

myY = value

End Set

End Property

Public Property Number() As Integer

Get

Return mNumber

End Get

Set(value As Integer) 'Anytime a new number is required redraw is forced

mNumber = value

Me.Invalidate()

End Set

End Property

Public Property View() As MineCellView

Get

Return mView

End Get

Set(value As MineCellView)

mView = value

Me.Invalidate()

End Set

End Property

'The paint routine to draw all the different views

Private Sub MineCell\_Paint(sender As Object, e As PaintEventArgs) Handles Me.Paint

Select Case mView

Case MineCellView.Button 'draws the button

With e.Graphics

.ResetTransform()

.TranslateTransform(Me.Width / 2, Me.Height / 2)

.ScaleTransform(Me.Width / 2, Me.Height / 2)

.Clear(mButtonColour)

'Special Points to shade

Dim TopLeft As New PointF(-1, -1)

Dim TopRight As New PointF(1, -1)

Dim BotLeft As New Point(-1, 1)

Dim BotRight As New Point(1, 1)

'lines for dark colour

Dim mPen As New Pen(Color.Gray, 0.3)

.DrawLine(mPen, BotRight, BotLeft)

.DrawLine(mPen, BotRight, TopRight)

'lines for light colour

mPen = New Pen(Color.White, 0.3)

.DrawLine(mPen, TopLeft, BotLeft)

.DrawLine(mPen, TopLeft, TopRight)

End With

Case MineCellView.Flag 'draws the flag - which in this case is a cop badge

With e.Graphics

.ResetTransform()

.TranslateTransform(Me.Width / 2, Me.Height / 2)

.ScaleTransform(Me.Width / 2, Me.Height / 2)

.Clear(Color.LightGray)

'Special Points to shade

Dim TopLeft As New PointF(-1, -1)

Dim TopRight As New PointF(1, -1)

Dim BotLeft As New Point(-1, 1)

Dim BotRight As New Point(1, 1)

'lines for dark colour

Dim mPen As New Pen(Color.Gray, 0.3)

.DrawLine(mPen, BotRight, BotLeft)

.DrawLine(mPen, BotRight, TopRight)

'lines for light colour

mPen = New Pen(Color.White, 0.3)

.DrawLine(mPen, TopLeft, BotLeft)

.DrawLine(mPen, TopLeft, TopRight)

'Draw Shield

Dim CRect As New RectangleF(-0.6, -0.6, 1.2, 1.2)

Dim CBrush As New SolidBrush(Color.Gold)

.FillEllipse(CBrush, CRect)

Dim TRect As New RectangleF(-0.6, -0.7, 1.2, 0.7)

.FillRectangle(CBrush, TRect)

'Write "cop"

Dim NBrush As New SolidBrush(Color.Black)

Dim myFont As New Font("Arial", 0.5, FontStyle.Bold, GraphicsUnit.World)

Dim SS As SizeF = .MeasureString("COP", myFont)

.DrawString("COP", myFont, NBrush, -SS.Width / 2, ((-SS.Height / 2) - 0.3))

End With

Case MineCellView.Mine 'draws mine which in this case is a cop shield

With e.Graphics

.ResetTransform()

.TranslateTransform(Me.Width / 2, Me.Height / 2)

.ScaleTransform(Me.Width / 2, Me.Height / 2)

.Clear(Color.WhiteSmoke)

'Draw Circle

Dim CRect As New RectangleF(-0.6, -0.6, 1.2, 1.2)

Dim CBrush As New SolidBrush(Color.Red)

.FillEllipse(CBrush, CRect)

'Draw Pegs

Dim IRad As Single = 0.5 'IRad is inside radius

Dim ORad As Single = 0.8 'ORad is outside radius

Dim PPen As New Pen(Color.Orange, 0.15)

PPen.EndCap = Drawing2D.LineCap.Round 'Make ends of light round

For ang As Single = 0 To 1.75 \* Math.PI Step 0.25 \* Math.PI

Dim Inner As New PointF(IRad \* Math.Cos(ang), IRad \* Math.Sin(ang))

Dim Outer As New PointF(ORad \* Math.Cos(ang), ORad \* Math.Sin(ang))

.DrawLine(PPen, Inner, Outer)

Next

'Write "!"

Dim NBrush As New SolidBrush(Color.White)

Dim myFont As New Font("Arial", 1.25, FontStyle.Bold, GraphicsUnit.World)

Dim SS As SizeF = .MeasureString("!", myFont)

.DrawString("!", myFont, NBrush, -SS.Width / 2, -SS.Height / 2)

'Draw Border

Dim BRect As New Rectangle(-1, -1, 2, 2)

Dim BPen As New Pen(Color.Gray, 0.05)

.DrawRectangle(BPen, BRect)

End With

Case MineCellView.Number

Dim NColors() As Color = {Color.Blue, Color.Green, Color.Red,

Color.Navy, Color.DarkGreen, Color.DarkRed,

Color.Brown, Color.Black}

With e.Graphics

'Set Scale

.ResetTransform()

.TranslateTransform(Me.Width / 2, Me.Height / 2)

.ScaleTransform(Me.Width / 2, Me.Height / 2)

.Clear(Color.WhiteSmoke)

If mNumber > 0 And mNumber <= 8 Then

'Draw a number

Dim NBrush As New SolidBrush(NColors(mNumber - 1))

Dim myFont As New Font("Arial", 1.5, FontStyle.Bold, GraphicsUnit.World)

Dim SS As SizeF = .MeasureString(mNumber.ToString, myFont)

.DrawString(mNumber.ToString, myFont, NBrush, -SS.Width / 2, -SS.Height / 2)

End If

'Draw Border

Dim BRect As New Rectangle(-1, -1, 2, 2)

Dim BPen As New Pen(Color.Gray, 0.05)

.DrawRectangle(BPen, BRect)

End With

Case MineCellView.Question

With e.Graphics

.ResetTransform()

.TranslateTransform(Me.Width / 2, Me.Height / 2)

.ScaleTransform(Me.Width / 2, Me.Height / 2)

.Clear(Color.LightGray)

'Draw question mark

Dim NBrush As New SolidBrush(Color.Black)

Dim myFont As New Font("Arial", 1.5, FontStyle.Bold, GraphicsUnit.World)

Dim SS As SizeF = .MeasureString("?", myFont)

.DrawString("?", myFont, NBrush, -SS.Width / 2, -SS.Height / 2)

'Special Points to shade

Dim TopLeft As New PointF(-1, -1)

Dim TopRight As New PointF(1, -1)

Dim BotLeft As New Point(-1, 1)

Dim BotRight As New Point(1, 1)

'lines for dark colour

Dim mPen As New Pen(Color.Gray, 0.3)

.DrawLine(mPen, BotRight, BotLeft)

.DrawLine(mPen, BotRight, TopRight)

'lines for light colour

mPen = New Pen(Color.White, 0.3)

.DrawLine(mPen, TopLeft, BotLeft)

.DrawLine(mPen, TopLeft, TopRight)

End With

End Select

End Sub

End Class

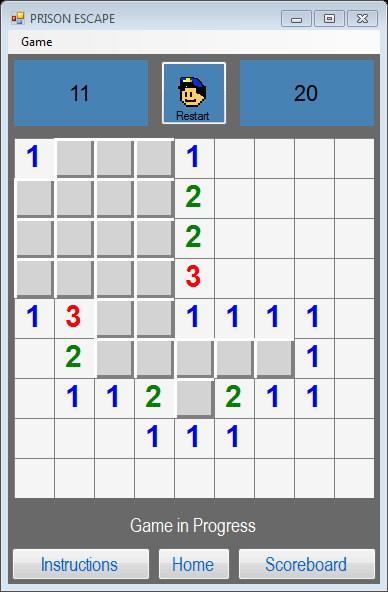
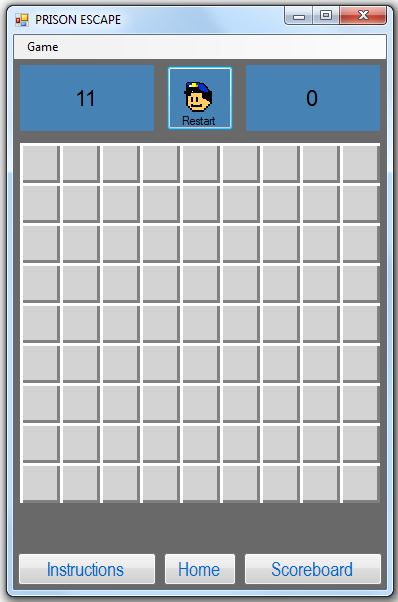
## Game

### Design

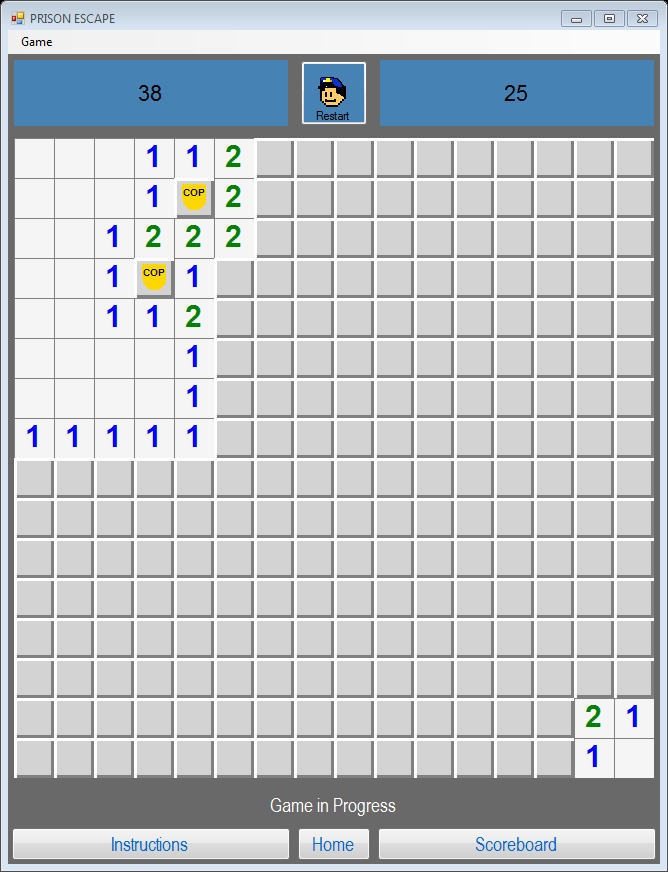


### Running Form

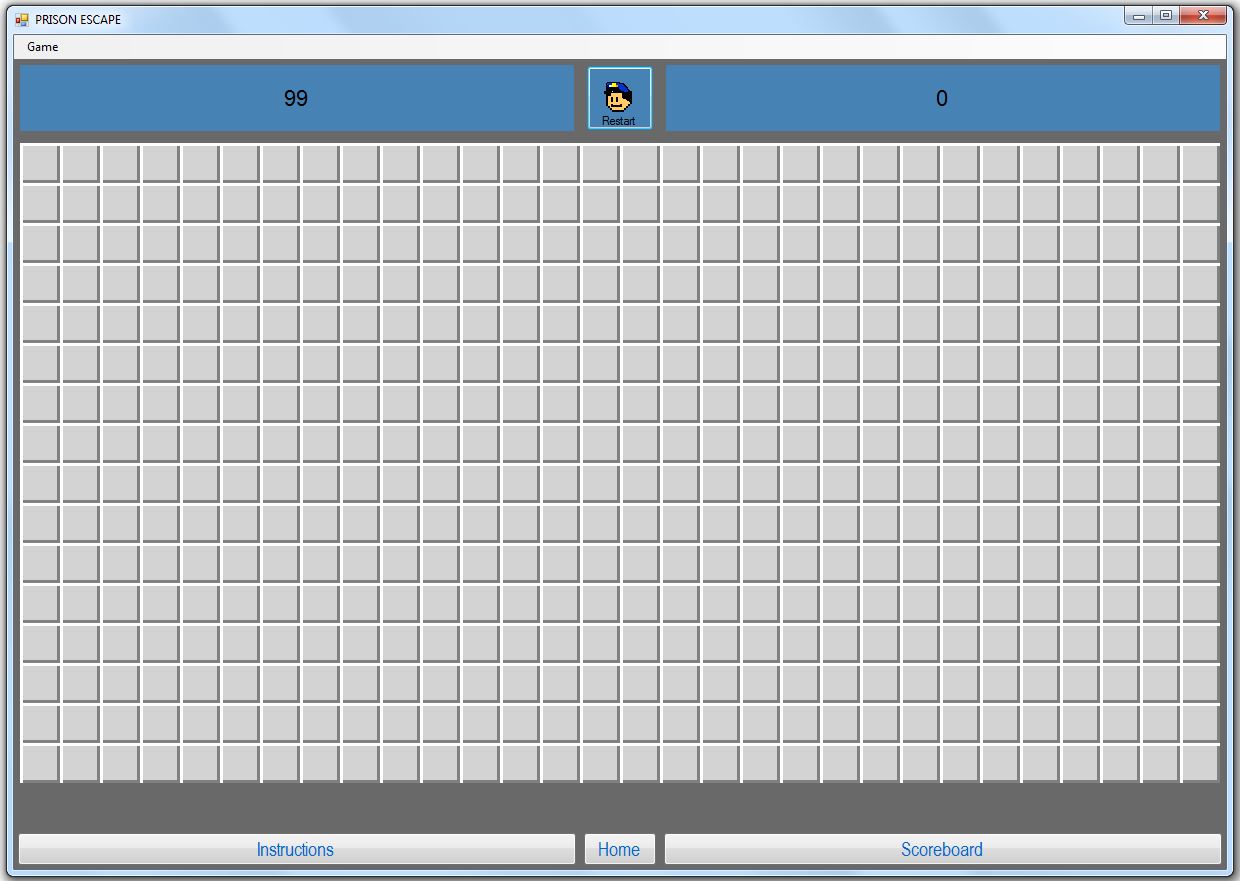
#### Beginner

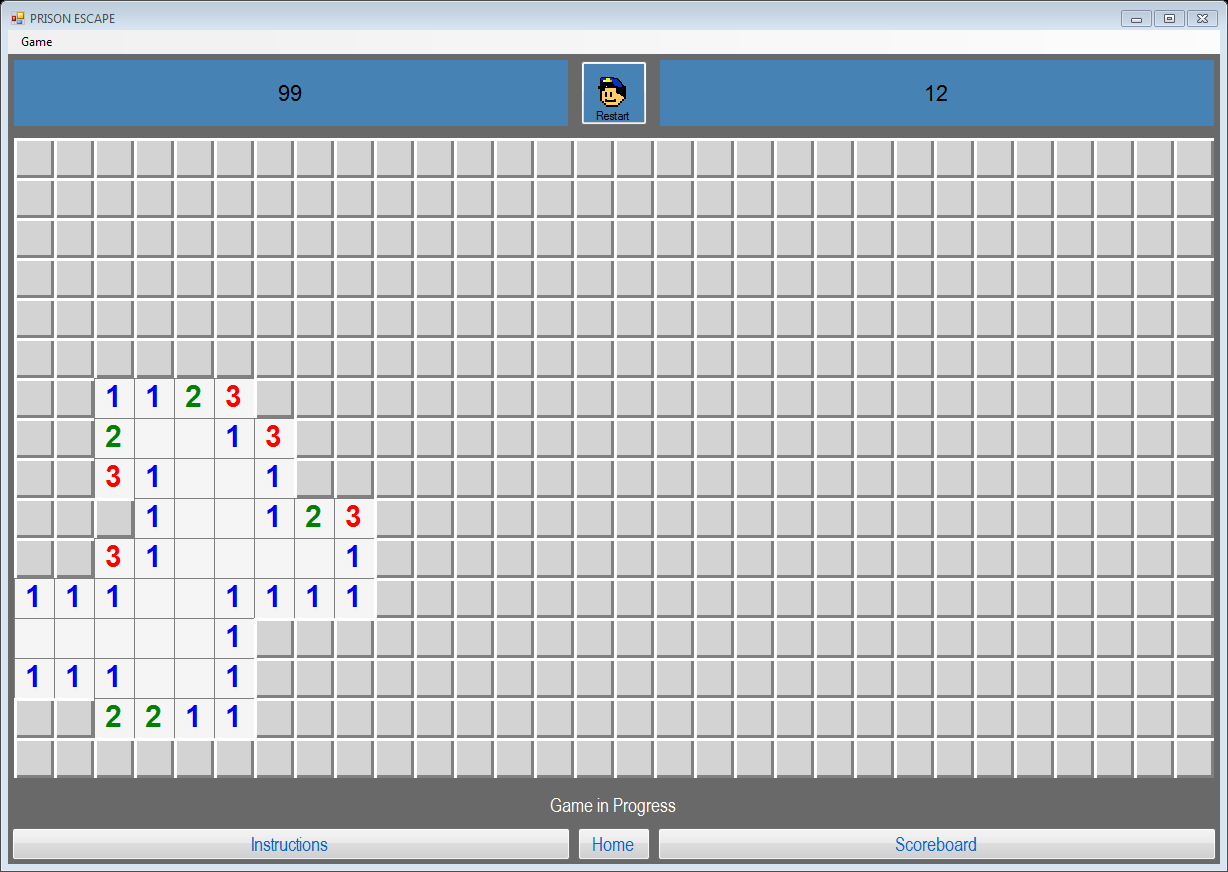


#### Intermediate



#### Expert





### Code

Imports Prison\_Escape.MineCell.MineCellView 'imports usercontrol 'minecell.minecellview' so that it can be refered to shorthand (ie as flag) instead of as Prison\_Escape.MineCell.MineCellView.flag

Public Class Start

Dim cellSize As Integer = 40 'Size of each minecell

Dim boardRows As Integer = 9 'Initial height of board

Dim boardCols As Integer = 9 'Initial width of board

Dim mineCount As Integer = 11 'Initial number of mines

Dim seconds As Integer = 0 'Initial value of time taken to solve puzzle

Dim markedMines As Integer = 0 'number of sirens that have been flagged

Dim finished As Boolean = False 'whether the game has completed (either click a mine, located all the mines or clicked all the "safe" spaces

'Initialising 2D Array to act as a grid

Dim mineField(9, 9) As MineCell

'creating a structure to store the player data so it can be stored and sorted

Structure pe\_player 'creates structure prisonescape player

Dim position As Integer 'leaderboard position

Dim playerName As String 'name of player

Dim playerTime As Integer 'time taken to complete game

Dim difficulty As Char 'what difficulty they were playing (beginner, intermediate, expert)

End Structure

Dim player As pe\_player 'defines player as the user defined structure pe\_player

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles Me.Load

'when the form loads go to sub procedure Buildboard()

BuildBoard()

End Sub

Private Sub BuildBoard() 'defines sub procedure as buildboard

finished = False 'sets the game over to false

seconds = 0 'resets initial variables

markedMines = 0

DoLabels() 'updates the labels

Timer1.Enabled = False 'turns the timer off

lblMessage.Text = String.Empty 'clears the message label

Me.Hide() 'hides the form until sirens have been placed

'Loop through the rows and columns

ReDim mineField(boardRows - 1, boardCols - 1)

pnlMine.Controls.Clear()

For Row As Integer = 0 To boardRows - 1 'Loop through the rows

For Col As Integer = 0 To boardCols - 1 'Loop through the columns

Dim C As New MineCell 'defines c as a new mincell so that all the cells can be placed

pnlMine.Controls.Add(C) 'adds minecell to panel

C.Left = cellSize \* Col 'sets location of cell (from left) to the cellsize times the column number

C.Top = cellSize \* Row 'sets location of cell (from top) to the cellsize times the row number

C.Width = cellSize 'sets the width to be the predefined cellsize

C.Height = cellSize 'sets the height to be the predefined cellsize

C.HasMine = False 'the default value is for there to be no mine in a cell

C.Number = 0

C.x = Col 'x coordinate to the column

C.y = Row 'y coordinate to the row

mineField(Row, Col) = C

AddHandler C.MouseClick, AddressOf MineClick

Next

Next

'Generate Random mine Locations

Dim RX As New Random

For i As Integer = 1 To mineCount 'loops through until predetermined number of mines have been placed

Dim X As Integer = 0 'sets x as (columns) a number equal to zero

Dim Y As Integer = 0 'sets y as (rows) a number equal to zero

Do

X = RX.Next(0, boardCols) 'Generates random x coordinate

Y = RX.Next(0, boardRows) 'generates random y coordinate

Loop Until Not mineField(Y, X).HasMine 'if it has a mine then this will be true

mineField(Y, X).HasMine = True 'sets the value of hasmine to true meaning that the siren location has been marked

'mineField(Y, X).View = MineCell.MineCellView.Mine this shows me where the mines are when i am testing

'mineField(Y, X).ButtonColour = Color.Red this changes the colour of the squares where there are mines to red

Next

'Count the mines

For Row As Integer = 0 To boardRows - 1 'this procedure counts the number of mines adjacent to a square with numbers in it

For Col As Integer = 0 To boardCols - 1

If Not mineField(Row, Col).HasMine Then

For R As Integer = Row - 1 To Row + 1

For C As Integer = Col - 1 To Col + 1

If R >= 0 And R < boardRows And C >= 0 And C < boardCols And Not (Row = R And Col = R) Then

If mineField(R, C).HasMine Then 'if there is a mines

mineField(Row, Col).Number += 1 'add one to the number in the square

End If

End If

Next

Next

End If

If mineField(Row, Col).Number > 0 Then

'mineField(Row, Col).View = MineCell.MineCellView.Number this shows me the numbers around the mines when testing

'mineField(Row, Col).ButtonColour = Color.Green this changes the colour of the cells with numbers in to green which is useful when testing

End If

Next

Next

'Resize to fit

Me.Width = boardCols \* cellSize 'changes the size of the form to the correct width

Do Until Me.pnlMine.Width = boardCols \* cellSize

Me.Width += 1

Loop

Me.Height = boardRows \* cellSize

Do Until Me.pnlMine.Height = boardRows \* cellSize 'changes the size of the form to the correct height

Me.Height += 1

Loop

Me.Show()

End Sub

Private Sub startGame(Rows As Integer, Cols As Integer, ByVal Mines As Integer)

boardRows = Rows

boardCols = Cols

mineCount = Mines

BuildBoard() 'goes to buildBoard procedure

End Sub

Private Sub MineClick(sender As Object, e As System.Windows.Forms.MouseEventArgs)

If finished Then Exit Sub

Dim M As MineCell = sender

Timer1.Enabled = True

lblMessages.Text = "Game in Progress" 'once the timer is on the game is playing

If e.Button = MouseButtons.Left Then 'if the left button is clicked then the following applies;

If M.HasMine Then 'if there is a mine in the cell then

For Each MC As MineCell In mineField

If MC.HasMine Then MC.View = Mine 'shows all mines

Next

Timer1.Enabled = False 'the timer stops

lblMessages.Text = "You have been caught!" 'the message updates to tell the user they have lost the game

finished = True

ElseIf M.Number > 0 Then 'if there is a number then

M.View = Number 'show the number

If gameOver() Then 'if game finish criterea has been met then

Timer1.Enabled = False 'the timer stops

lblMessages.Text = "You have successfully escaped from prison" 'the message updates to tell the user they have won the game

finished = True 'the game is now finished

export\_times(player, boardCols) 'the export\_times procedure is called

End If

ElseIf M.Number = 0 Then 'if there are no mines adjacent then

ShowBlanks(M) 'go to procedure showblanks until all the blanks have been cleared up to either the edge of the grid or a border of numbers

If gameOver() Then 'if this means they have beaten the game then

Timer1.Enabled = False 'the timer stops

lblMessages.Text = "You have successfully escaped from prison" 'the message updates to tell the user they have won the game

finished = True 'the game is now finished

export\_times(player, boardCols) 'the export\_times procedure is now called

End If

End If

ElseIf e.Button = MouseButtons.Right Then 'if the right mouse button is clicked

Select Case M.View 'selects to compare the current view of the cell

Case Button 'if the cell is a button then

M.View = Flag 'make it a flag

markedMines += 1 'increase the number of marked mines

DoLabels() 'update labels to show this

Case Flag 'if the cell is a flag then

M.View = Question 'make it a question mark

markedMines -= 1 'reduce number of marked mines

DoLabels() 'update labels to show this

Case Else

M.View = Button 'if it is currently a question make it a button

End Select

End If

End Sub

Private Sub ShowBlanks(M As MineCell) 'recursive algorithm to clear blank cells

M.View = Number

For R As Integer = M.y - 1 To M.y + 1 'for the rows one above and one below of the clicked box

For C As Integer = M.x - 1 To M.x + 1 'for the columns one to the left and one to the right of the clicked box

If R >= 0 And R < boardRows And C >= 0 And C < boardCols Then 'if cell is on board

Dim MC As MineCell = mineField(R, C) 'sets the value of mc as current minecell

If MC.View = Button Then 'if there is a button ie hasn't already been cleared

If MC.Number = 0 Then 'if the number underneath the button is zero ie there are no mines adjacent to cell

ShowBlanks(MC) 'then go to this procedure with this cell as the actual parameter - this will repeat until only cells with mines adjacent are left

Else

MC.View = Number 'if there is a number then show that number

End If

End If

End If

Next

Next

End Sub

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

seconds += 1 'adds one to the number of seconds every second

DoLabels() 'goes to the update labels procedure to update this

End Sub

Private Sub DoLabels()

lblClock.Text = seconds.ToString 'writes the current time value to label

lblMines.Text = (mineCount - markedMines).ToString 'writes current number of sirens to label

End Sub

Private Function gameOver() As Boolean

Dim TV As Boolean = True 'set game over to be true

For Each MC As MineCell In mineField 'for every cell in grid

If Not MC.HasMine And Not MC.View = Number Then 'if there isnot a mine or a number then

TV = False 'set game over to be false

End If

Next

Return TV

End Function

Private Sub BeginnerToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles BeginnerToolStripMenuItem.Click

'starts game with 9 rows, 9 columns and 11 sirens

startGame(9, 9, 11)

End Sub

Private Sub IntermediateToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles IntermediateToolStripMenuItem.Click

'starts game with 16 rows, 16 columns and 40 sirens

startGame(16, 16, 40)

End Sub

Private Sub AdvancedToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles AdvancedToolStripMenuItem.Click

'starts game with 16 rows, 30 columns and 99 sirens

startGame(16, 30, 99)

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles NewToolStripMenuItem.Click, Button1.Click

BuildBoard()

End Sub

Private Sub export\_times(ByVal player As pe\_player, ByVal boardCols As Integer)

Select Case boardCols 'this is to determine what difficulty the user was playing

Case 9

player.difficulty = "B" 'if there are 9 columns then beginner

Case 16

player.difficulty = "I" 'if there are 16 columns then intermediate

Case 30

player.difficulty = "E" 'if there are 30 columns then expert

End Select

player.playerName = InputBox("Please enter you name. Note it must be less than 20 characters to ensure it is properly displayed on the scoreboard") 'sets player name to be defined by the user

If Len(player.playerName) > 20 Or Len(player.playerName) < 1 Then 'input validation

MsgBox("Name must be less than 20 and more than 0 characters to ensure it is properly displayed on the scoreboard")

End If

player.playerTime = seconds 'takes the time taken to complete the game from the timer

Dim export As String

Dim file As System.IO.StreamWriter

Select Case player.difficulty 'exports to different files depending on difficulty

Case "B"

file = My.Computer.FileSystem.OpenTextFileWriter("Z:\S6\Computing Science\PROJECT\Project\BeginnerLeaderboard.csv", True) 'open file with this path

export = (player.playerName & ", " & player.playerTime) 'set export to be current player stats

file.WriteLine(export) 'write current player stats to file

file.Close() 'close the file

Case "I"

file = My.Computer.FileSystem.OpenTextFileWriter("Z:\S6\Computing Science\PROJECT\Project\IntermediateLeaderboard.csv", True) 'open file with this path

export = (player.playerName & ", " & player.playerTime) 'set export to be current player stats

file.WriteLine(export) 'write current player stats to file

file.Close() 'close the file

Case "E"

file = My.Computer.FileSystem.OpenTextFileWriter("Z:\S6\Computing Science\PROJECT\Project\ExpertLeaderboard.csv", True) 'open file with this path

export = (player.playerName & ", " & player.playerTime) 'set export to be current player stats

file.WriteLine(export) 'write current player stats to file

file.Close() 'close the file

End Select

End Sub

Private Sub btnHome\_Click(sender As Object, e As EventArgs) Handles btnHome.Click

My.Forms.Form1.Show() 'shows home screen

Me.Close() 'closes game from

End Sub

Private Sub btnInstructions\_Click(sender As Object, e As EventArgs) Handles btnInstructions.Click

My.Forms.Instructions.Show() 'opens the instructions

End Sub

Private Sub btnScoreboard\_Click(sender As Object, e As EventArgs) Handles btnScoreboard.Click

My.Forms.scoreboard.Show() 'opens the scoreboard

End Sub

End Class

## References

<https://msdn.microsoft.com/en-us/library/system.windows.forms.keys(v=vs.110).aspx>

Syntax for keyboard and mouse inputs

<https://msdn.microsoft.com/en-us/library/system.io.streamreader.peek(v=vs.100).aspx>

Reading until end of file

<https://msdn.microsoft.com/en-us/library/system.io.file.appendtext(v=vs.110).aspx>

How to append text files

<https://msdn.microsoft.com/en-us/library/aa302342.aspx>

Creating a user control

<https://msdn.microsoft.com/en-us/library/mt186350.aspx>

Adding icon to button in game

<http://stackoverflow.com/questions/42695054/where-is-my-error-in-this-visual-basic-bubblesort>

Help from developers finding error in the bubblesort

# Testing

## Component

I tested the components of the solutions as I went along to ensure that they were working individually and on their own all the individual elements of the solution were working correctly however due to the nature of the solution different functions and procedures needed to be tested together due to each procedures dependency on another procedure.

### Place Warning Light Procedure

To test this, I just ran the game but told edited the code so that it showed me if a cell had a warning light in it and then counted the number of warning lights.

#### Beginner

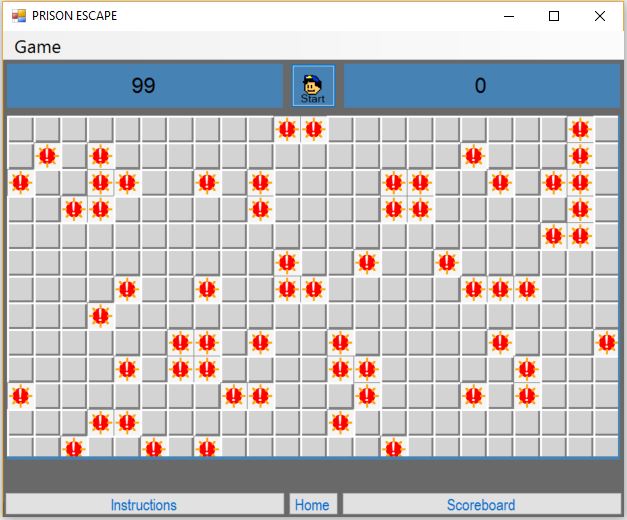
|  |  |
| --- | --- |
| Test Number | Correct Number of Warning Lights Placed (11) |
| 1 | Yes |
| 2 | Yes |
| 3 | Yes |
| 4 | Yes |
| 5 | Yes |

#### Intermediate

|  |  |
| --- | --- |
| Test Number | Correct Number of Warning Lights Placed (40) |
| 1 | Yes |
| 2 | Yes |
| 3 | Yes |
| 4 | Yes |
| 5 | Yes |

#### Intermediate

|  |  |
| --- | --- |
| Test Number | Correct Number of Warning Lights Placed (99) |
| 1 | Yes |
| 2 | Yes |
| 3 | Yes |
| 4 | Yes |
| 5 | Yes |



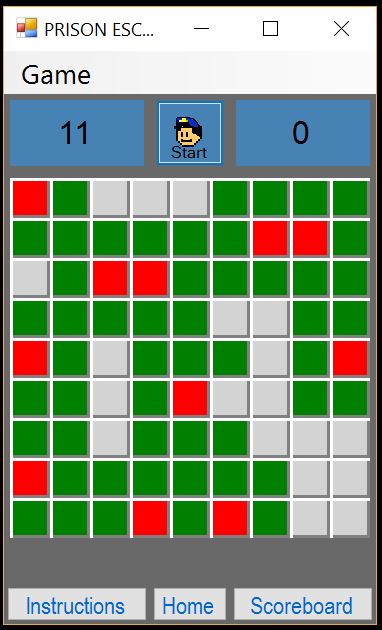
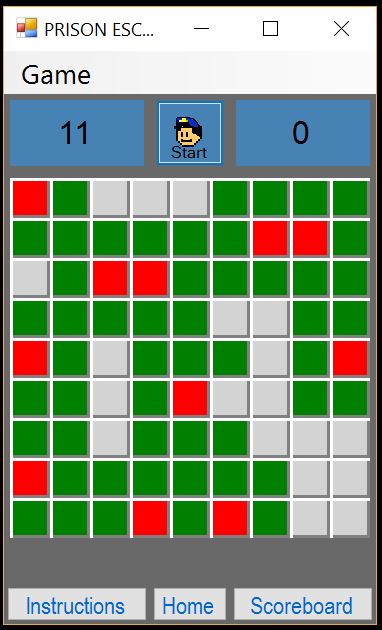
## Integrative

### Numbers Next to Warning Lights

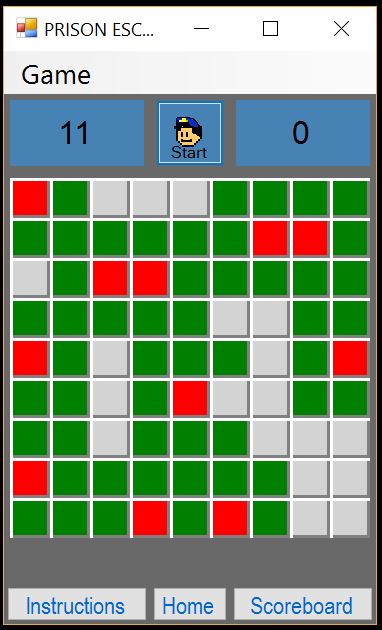
In the following screenshots and table I tested multiple forms to ensure that all cells adjacent to warning lights had numbers in them. The red cells are lights with the green ones being numbers

|  |  |  |
| --- | --- | --- |
| Difficulty | Test Number | Numbers next to Warning Lights |
| Beginner |  |  |
|  | 1 | Yes |
|  | 2 | Yes |
|  | 3 | Yes |
| Intermediate |  |  |
|  | 1 | Yes |
|  | 2 | Yes |
|  | 3 | Yes |
|  | 4 | Yes |
| Expert |  |  |
|  | 1 | Yes |
|  | 2 | Yes |
|  | 3 | Yes |
|  | 4 | Yes |

#### Beginner Intermediate



#### Expert



### Recursion Algorithm

To test the recursion algorithm, I got the program to tell me where the cells with neither a warning light or a number were and then checked what cells should cleared when I was to click on an empty cell. For the example, I will draw a cross on the cells that should be cleared (also all green cells within these cells with crosses should be cleared) when any of the green cells within that area are clicked.

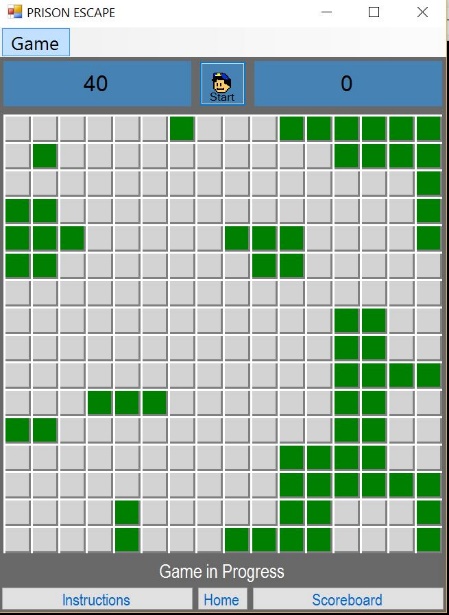
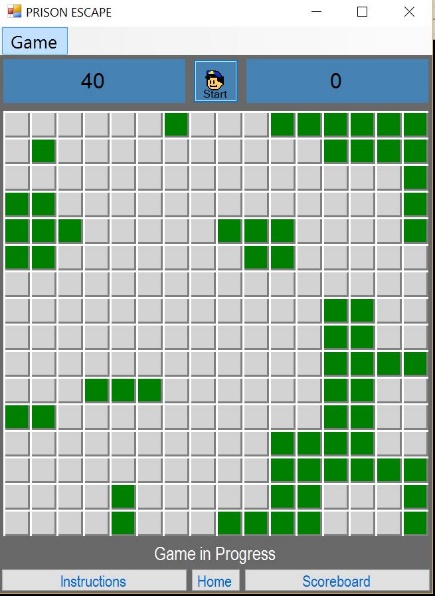
#### Beginner

#### 



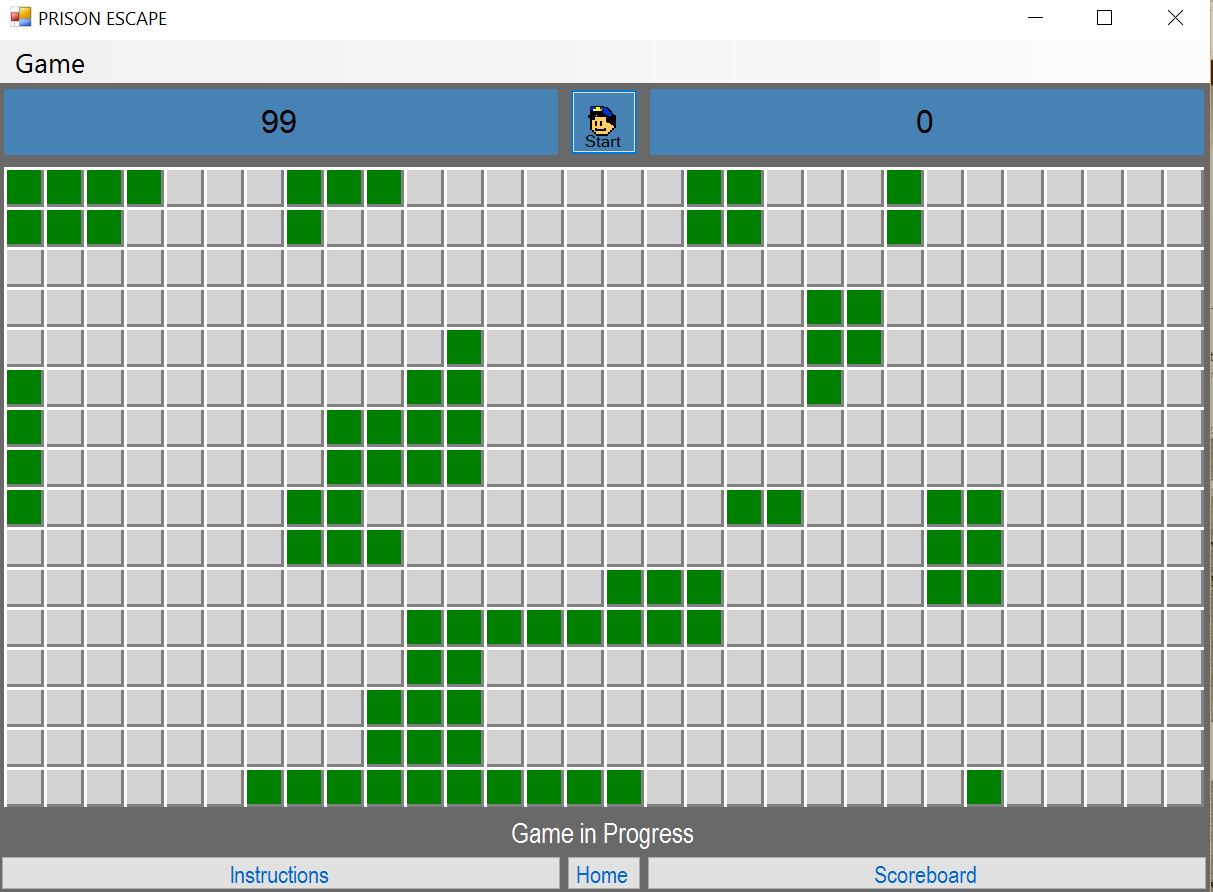
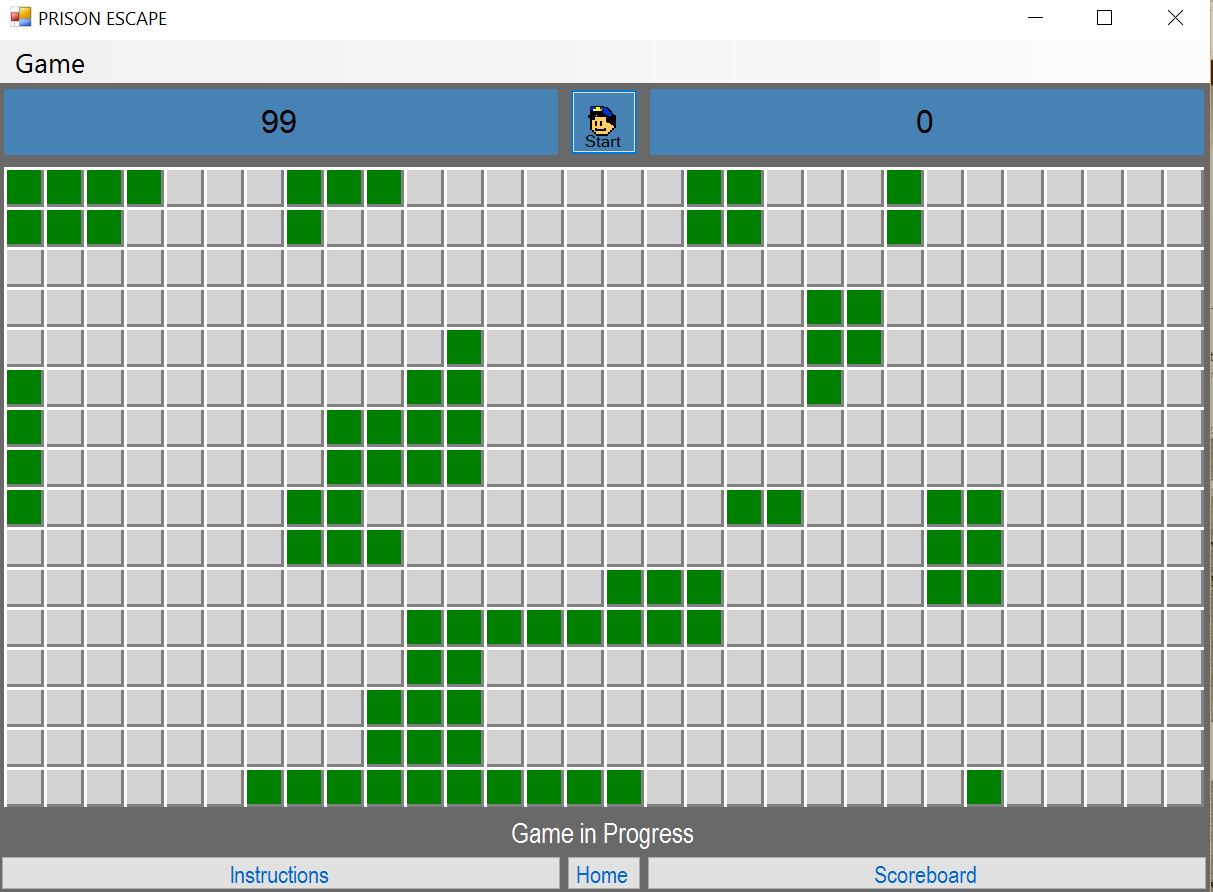
|  |  |  |
| --- | --- | --- |
| Test Number | Number of Cells to Clear | Correct Cells Cleared |
| 1 | 14 | Yes |
| 2 | 32 | Yes |
| 3 | 31 | Yes |

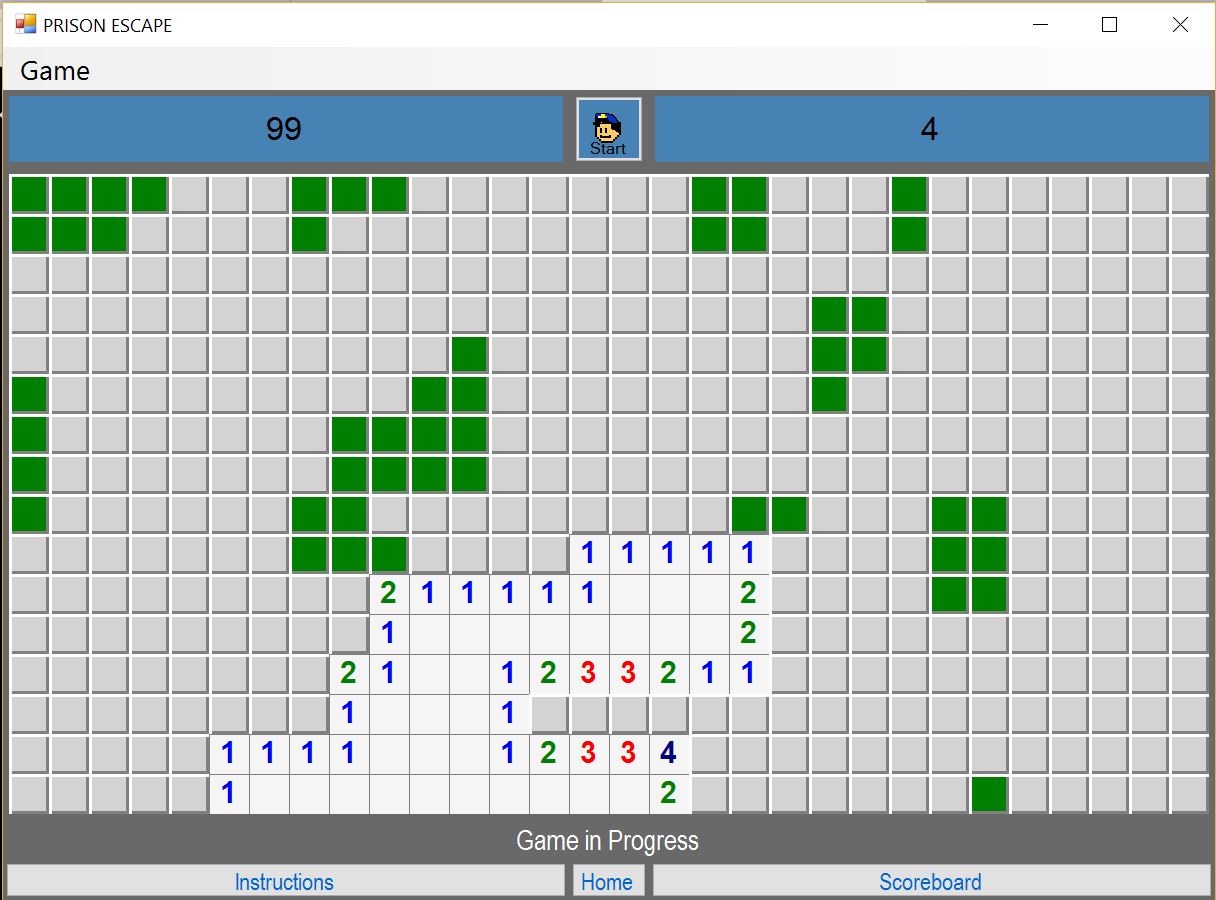
#### Intermediate



|  |  |  |
| --- | --- | --- |
| Test Number | Number of Cells to Clear | Correct Cells Cleared |
| 1 | 35 | Yes |
| 2 | 41 | Yes |
| 3 | 27 | Yes |

#### Expert





|  |  |
| --- | --- |
| Test Number | Correct Cells Cleared |
| 1 | Yes |
| 2 | No |

Due to there being an error I searched the code and found a logic error. The original and updated code is shown below.

##### Original

Private Sub ShowBlanks(M As MineCell)

M.View = Number

For R As Integer = M.y - 1 To M.y + 1

For C As Integer = M.x - 1 To M.x + 1

If R >= 0 And R < boardRows And C>=0 And C < boardRows Then

##### Updated

Private Sub ShowBlanks(M As MineCell)

M.View = Number

For R As Integer = M.y - 1 To M.y + 1

For C As Integer = M.x - 1 To M.x + 1

If R >= 0 And R < boardRows And C>=0 And C < boardCols Then

As the initial boards are square the boardRows and the boardCols values were the same however when the board became rectangular for the expert level the algorithm wasn’t going to the edge of the board.

|  |  |
| --- | --- |
| Test Number | Correct Cells Cleared |
| 3 | Yes |
| 4 | Yes |
| 5 | Yes |

### Other Game Testing

The right click to mark mines worked and functioned correctly along with the game status, marked lights and timer algorithms. There is no in depth description of the testing for these as the testing matches the test plan with the outputs being as expected.

*Requirements tested in previous section: 4, 5, 6, 7, 11, 12, 13, 14, 16, 17, 19*

### Input Validation

*Requirements 9, 14*

|  |  |  |
| --- | --- | --- |
| Input | Expected Output | Real Output |
| Abcdef1234 | Accepted (<20, >0 characters) | Accepted |
| Bob | Accepted (<20, >0 characters) | Accepted |
| 12345678oasdafasfhsdrklhsrt | Not Accepted (27 Characters) | Not Accepted |
| abcdefghijklmnopqrstu | Not Accepted (21 Characters) | Not Accepted |
| Abcdefghijklmnopqrst | Accepted (20 Characters) | Accepted |
| A | Accepted (1 Character) | Accepted |
|  | Not accepted (0 characters) | Not Accepted |

### Scoreboard

*Requirements 8, 9, 10, 15, 18*

#### Sort

|  |  |  |
| --- | --- | --- |
| Input | Expected Output | Real Output |
| Bob, 2  Alex, 10  Tom, 15 | 1 Bob 2  2 Alex 10  3 Tom 15 | 1 Alex 10  2 Tom 15  3 Bob 2 |

This error was caused by a variable type error in the record structure. The original and updated code is shown below.

##### Original

Structure pe\_player

Dim position As Integer

Dim playerName As String

Dim playerTime As String

Dim difficulty As Char

End Structure

As you can see the player time is stored as a string so it is getting sorted by the ASCII value of the first character. To fix this it just required being changed to an integer.

##### Updated

Structure pe\_player 'creates structure

Dim position As Integer 'defines position as integer

Dim playerName As String 'defines player name as string

Dim playerTime As Integer 'defines player time as integer

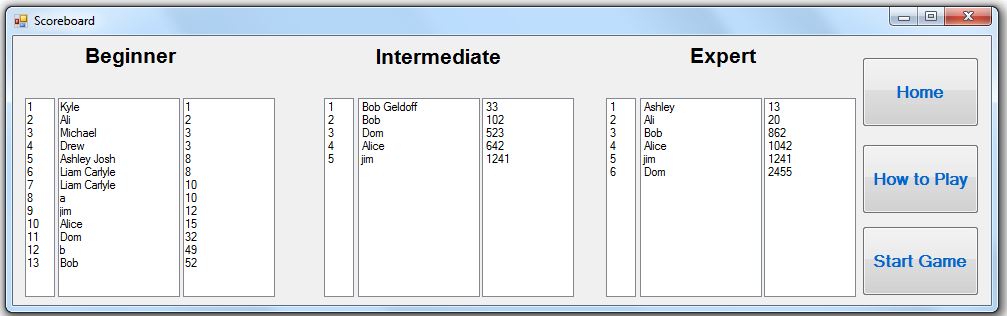
Dim difficulty As Char 'defines difficulty as char

End Structure

|  |  |  |
| --- | --- | --- |
| Input | Expected Output | Real Output |
| Bob, 2  Alex, 10  Tom, 15 | 1 Bob 2  2 Alex 10  3 Tom 15 | 1 Bob 2  2 Alex 10  3 Tom 15 |
| Bob, 10  Alex, 5  Tom, 2 | 1 Tom 2  2 Alex 5  3 Bob 10 | 1 Tom 2  2 Alex 5  3 Bob 10 |
| Bob, 3  Alex, 3  Tom, 4 | 1 Bob 3  2 Alex 3  3 Tom 4 | 1 Bob 3  2 Alex 3  3 Tom 4 |

The reason the final test input doesn’t make both Bob and Alex first place is due to the fact the due to the way the data is stored and appended Bob will have achieved his time before Alex achieved this so I feel that differentiating in the way is the best way to add another competitive element to the game.

##### Example Output



## Beta Testing (Acceptance and Usability)

Requirements 1, 2, 3 and all functional requirements.

To gauge the opinion of beta testers I created the following form for them to fill in and use.

|  |  |
| --- | --- |
| Task (Complete in Order) | Comments (Usability, Robustness, Functionality) |
| In your opinion, what sort of computer experience and what sort of strategy game experience do you have? | * Computer – * Strategy Game – |
| Navigate to Instructions from Home |  |
| Navigate to Game and then Scoreboard from Home |  |
| Navigate from instructions to scoreboard |  |
| Click on Start game |  |
| Right Click on any cell to cycle through markers |  |
| Right Click all cells on board |  |
| Left Click any cell on board |  |
| Change Difficulty to Intermediate |  |
| Change Difficulty to Expert |  |
| Play a game on Beginner |  |

## Completed Responses

|  |  |
| --- | --- |
| Task (Complete in Order) | Comments (Usability, Robustness, Functionality) |
| In your opinion, what sort of computer experience and what sort of strategy game experience do you have? | * Computer – not very much * Strategy Game – A fair amount |
| Navigate to Instructions from Home | Clicked on instructions and was taken there |
| Navigate to Game and then Scoreboard from Home | I just click on the text with what you were asking and ended up in the right place |
| Navigate from instructions to scoreboard | No issue with this either – this is good as I usually struggle with the navigation |
| Click on Start game | A board has loaded |
| Right Click on any cell to cycle through markers | The markers all seem to be working |
| Right Click all cells on board | All cells were clickable |
| Left Click any cell on board | An area became blank and some numbers appeared |
| Change Difficulty to Intermediate | Since I was on the instruction I knew this was located in a drop-down menu. A larger square board appeared |
| Change Difficulty to Expert | This was also located in drop down menu which is easy to use. |
| Play a game on Beginner | I managed to complete the game on beginner which was good since I don’t use a computer very often. I liked this game and it was easy enough to play. |

|  |  |
| --- | --- |
| Task (Complete in Order) | Comments (Usability, Robustness, Functionality) |
| In your opinion, what sort of computer experience and what sort of strategy game experience do you have? | * Computer – A lot * Strategy Game –A lot |
| Navigate to Instructions from Home | This was very straightforward |
| Navigate to Game and then Scoreboard from Home | This was also very straightforward – I like the way the scoreboard pops up so it doesn’t close your game |
| Navigate from instructions to scoreboard | Again, very simple |
| Click on Start game | Done |
| Right Click on any cell to cycle through markers | Yes, all markers were present and no errors encountered when cycling through |
| Right Click all cells on board | Cells are of a good size so that they can be easily clicked on |
| Left Click any cell on board | The cell became a number |
| Change Difficulty to Intermediate | This was accessible from the game drop down menu – I liked this as it did not distract from an in-progress game. |
| Change Difficulty to Expert | Game board became rectangular – I like the way the board changes to the right size automatically |
| Play a game on Beginner | I completed the game and it asked me to input my name which there was no problem with. I then checked the high score table and my name was there along with my time. Overall very usable and playable. |

|  |  |
| --- | --- |
| Task (Complete in Order) | Comments (Usability, Robustness, Functionality) |
| In your opinion, what sort of computer experience and what sort of strategy game experience do you have? | * Computer – loads * Strategy Game – Not a lot |
| Navigate to Instructions from Home | Very straightforward and simple |
| Navigate to Game and then Scoreboard from Home | Once again, no real trouble in doing this |
| Navigate from instructions to scoreboard | Very simple task. Forms are easily navigated. |
| Click on Start game | Game board was loaded |
| Right Click on any cell to cycle through markers | Markers seemed to be cycled through correctly |
| Right Click all cells on board | All cells allowed me to right click |
| Left Click any cell on board | I clicked on a light and all the lights were revealed |
| Change Difficulty to Intermediate | In drop down menu |
| Change Difficulty to Expert | Again, in drop down menu – nice use of space |
| Play a game on Beginner | This was a very playable game and although I did not complete it I think that it was very straightforward. |

## Beta Test Feedback

All of the users found the game as a whole to be usable and playable. The beta testers found no errors in the solution however they did suggest a few improvements that could be made to the solution - most of their suggestions were similar to the ones initially made by end-users in the surveys and so could be added to any further developments or refinements of the solution in order to improve the overall user experience of the game.

## Final Testing

Due to the positivity of the beta tester feedback I felt that there was no need to alter the solution. I would have liked to add some more extra features but due to the time constraints placed on me I didn’t have time to add any more features.

## Overall Testing

Throughout my project, I have been testing components and testing the communication between different modules of my program. This allowed me to spend less time fixing errors at the end of the implementation stage. Having now completed all staged of testing I can now confirm that my solution meets all the requirements, user and functional, set out in the requirement specification.

# Evaluation

Overall my completed solution meets all the requirements specified in the coursework guidelines. It included the Advanced Higher practical skills of implementing recursion, 2D arrays, arrays of records, and sort algorithms. It also included the required interfacing with stored data and the application only has one out of game input, which is validated. The game *Prison Escape* also meets the requirements set by the end-users to be an enjoyable and playable version of the game minesweeper, with a prison escape related theme. *Prison Escape*,and the other parts of the project, have been completed by the deadline and *Prison Escape* is suitable for the end-users of the game.

## Requirements

In terms of end-user requirements and project requirements I believe this solution has met all of the requirements above. The opinions and thoughts gathered in the user surveys have been listened to and implemented in the solution.

The requirement specification set out a range of user and functional requirements that the solution must adhere to. The end user requirements have all been met as the solution has a working and functional navigational structure – all the links are clear and easily navigable throughout the solution. The game also meets the second end-user requirement of being playable – the game has not been found to have many bugs meaning that the user has a good experience. The final user requirement, and the one most sought after by end-users, was the high score table to add a competitive element to the game. The high score table sorts and displays the fastest times that the different difficulties have been solved in.

The larger range of functional requirements has also been met. There is clear and consistent layout throughout the entirety of the application along with the working links and navigation which make this game usable and playable, even by people who are not used to playing strategy games. The ability to mark cells the user believes has warning lights in it is also functional and working correctly.

There is a button to restart a finished game, along with a clear instructions screen that is always accessible from anywhere in the application. The high score table is functional and correctly sorts the range of times from the fastest to the slowest along with this the game clears all blank cells in an area and allows users to mark mines.

Through comprehensively testing this solution I have found that the application is robust and reliable enough to handle incorrect data types along with a large range of data values. Due to the minimal input from the user before the game – the only things inputted are button clicks the entirety of the out of game process meets the requirements. The only text based user input is their name, this is limited to 20 characters so that it can be properly displayed on the scoreboard and this was working correctly throughout the testing process and no errors were noted. There is a possibility that the high score files could be interfered with in their current form so I would probably store them in a secure location when publishing the solution.

All inputs and outputs are stored correctly and have the correct amount of validation to balance user experience. The use of three high score files increased the overall amount of code required in the project due to the fact the one procedure would only read one file, meaning that some lines of code needed to be repeated to read the three separate files. The use of the sort algorithm however, was used to sort each of three tables, reducing the amount of repeated lines of code and increasing the overall readability of the code. Due to myself building minesweeper and then editing the code and files to add a prison theme into the game, as suggested through the user surveys, in the code the word mine is sometimes referred to, on publication of the solution I would ensure that all the procedure and variable names are more relevant – as the names were already meaningful and explained in comments I felt no reason to change them whilst completing my solution but acknowledge that these names should be edited if the solution was to be published.

## Testing

### General

Due to the fact that I tested the implementation as I went along the initial stages of the testing were successful with all components being in working order with no bugs or errors being found in the initial stage. Deciding the test the components as I went along allowed me to spend less time on the testing and implementation as I was able to ensure that I was not writing code on top of code that had logic, or any other type of errors in it.

There were a few bits of corrective maintenance when I tested the integrated solution in order to fix small data passing errors and a couple of logic errors that were causing the solution to output incorrect results. I think it was beneficial to locate and correct these errors before the end-user testing was carried out as it allowed their testing to focus on the usability of the solution. This was helpful as to ensure that the game is usable it needs to be tested by users other than myself as although I may find the solution usable an end-user may not so any errors there need to be picked up and fixed.

### End-User Beta (Acceptance and Usability)

To test the solution completely I decided to ask users of different abilities to test out the solution meaning that I got a broad range of views as to how the solution works in practise. I asked users with different levels of both strategy computer game experience and computer experience as a whole. This allowed me to ensure that my solution would be usable for all potential users.

The beta testers found no errors in the solution however they did suggest a few improvements that could be made to the solution. However, due to the time constraints I did not have time to add any of the features to the solution at this time. Most of the suggestions were in line with the ones initially made by end-users in the surveys and so could be added to any further developments or refinements of the solution in order to improve the overall user experience of the game.

I think that choosing this test group for the beta testing has allowed me to get a good feel as to the overall quality of my solution and allowed me to gauge the opinions of a wide range of possible end-users. Getting the feedback from three different end-users allowed me to see a variety of perspectives on the overall quality of the game.

### Final Testing

During the final testing of the solution I made sure that the final submitted solution met all of the requirements specified in the requirements specification. As the end-user testing, had already occurred at this stage the usability testing had already been carried out so it allowed me time to ensure that the solution met the functional requirements.

At this stage I found there to be no errors and no maintenance of any sort was carried out in this stage.

## Further Developments and Refinements

Most further developments would be adding more features to *Prison Escape*. I would probably try and incorporate more of the features suggested by end users that I couldn’t add into the project due to time constraints. The features such as a player versus player mode would add an even more competitive element to the game rather than just the high score table. Other things I would have liked to add are more animation features so that if a user clicks on a light all the warning lights are revealed and then flash – this would act to improve the user experience. Also, a ladder level system where users are trying to complete more levels may add to the user experience. The system would work that players gradually built their way up from small boards to very large boards however if they made a mistake they would have to begin again from the first level. This feature was also mentioned in the end-user surveys and a lot of end-users were keen to see this feature implemented.

Initially there was supposed to a level/story mode included, where users try to escape from a prison by working their way through different rooms and locations in the prison. As the deadline approached I realised that I would not have time to include such a feature due to time constraints caused by the implementation stage taking longer than expected to complete, this was due to the numbers of errors and bugs that had to be ironed out in order to make the solution complete and fit for purpose. If I had accounted for the length of time taken to research and learn about user controls. If I had factored all of these into my planning I would have been more successful in adding this feature to *Prison Escape*.

## Development Process

The development process as a whole was in my view successful. The initial stages of planning and analysis were completed on time and gave me a good idea of what I needed to implement and how long each stage could take in order to complete the project in the timescale given. Due to the fact that my project was completed within the given timescale and before the deadline for submission I feel that these first stages were successful.

The design and implementation stages I found were the most important stages in terms of keeping the project on time and being able to submit a full and completed solution. The design stage allowed me to plan every aspect of the development from the navigation and data flow structures to the design of the form and the pseudocode. This allowed for a successful implementation stage and although it did take two hours longer than initially allowed for I would still say that these stages were completed, in general, successfully.

The testing stage was also very helpful in locating any minor bugs and faults in my solution. The fact that I was testing my solution as I went along allowed me to enter the final solution without knowing that there would be a requirement to perform corrective maintenance to my solution in order to fix major bugs. This meant that heading into the testing stage I was confident my solution would be able to cope with the testing specified in the test plan and also any testing that the beta testers carried out.

Due to the iterative nature of the development method I chose to complete this project the solution was continually refined throughout the development of the project. This iteration allowed for a completed project to contain all the functional and user requirements and also be carried out in the allowed timescale.

## Performance and Learning

During the process of creating *Prison Escape* I tracked my advancement using a progress diary and noted down any changes from the design and the reasons for them in it. This along with a Gantt Chart and project plan ensured that *Prison Escape* was completed with the correct requirements and within the timescale given to me by my assessor. These were effective ways to keep track throughout the development of the project as they allowed me to reflect on changes made earlier in the project and apply them to the current problem in order to find a solution quickly.

*Prison Escape* was completed on time meaning that my time management and project planning was of a good standard. The use of a Gantt Chart to map out a firm project timeline was a very helpful addition in terms of me ensuring that *Prison Escape* was completed on time. Towards the end of the implementation process I had to move on and leave behind the previously mentioned story mode feature – this was always going to be an additional feature and not one that was mentioned in the requirements specification due to the fact that it was not required by end-users or by the project criteria. The entirety of the development process was planned meticulously and this resulted in a solution being completed before the deadline for submission of the project.

Initially I was not sure where to begin in designing my solution but I was advised to break up the design into many different sections so that I knew exactly what my finished solution was going to look like. In the design stage I also wrote the pseudocode meaning that I just needed to adapt the pseudocode into the specific syntax I was using to implement my solution. This saved time in the implementation stage as I could see what procedures and variables needed used in the section of code. The data flow and structure diagrams also made the development progress much quicker due to the fact I knew what data needed to be passed between each procedure. This made me appreciate the value of the design stage even more than I previously did, I realised that a successful design stage would allow for the overall project to be completed in a faster and more complete way.

I set out to create *Prison Escape* as it the project was unlike anything I have taken on before. It allowed me to learn new skills and apply skills I had already learned in the past, in new and original ways. I had to learn and gain knowledge about user controls and passing data between forms. These new skills will have relevant applications in my future learning and usage of programming. This was also the first time I had applied complex recursion in a real-life context which further added to my understanding of the language and my understanding of the logical process of programming languages as a whole.

This project also taught me the best ways to find and track errors of all types. I already had an idea of the debugging and error fixing that software developers need to be able to do, but I don’t think I anticipated quite how much it would take. I think that spending more time in the planning and design stages would have enabled me to refine potential solution paths so that less errors would occur. I think if I was to undertake a project like this again I would also allow more time for the implementation of my solution by allocating it a larger proportion of project time in order to ensure that all desired features could be implemented and added to the solution successfully. Allowing for more time in the implementation stage would also allow me to spend more time finding and fixing errors in the code.

Overall, I would say I performed well throughout developing this solution and throughout the project as a whole. My project meets all the end-user and functional requirements and was found to be usable by all beta testers. I completed all the required sections of the project on time and followed a development process with all relevant stages of development to implement my solution.

## Implications

### General

*Prison Escape* is intended to be used as an open source strategy puzzle game. The game is intended for personal use, although the game could have some educational purposes to teach people about strategy and logic.

### Environmental

There are some environmental implications here, as there is with any software development due to the required energy and resources need to manufacture and run computers. These processes have a large carbon footprint so there are implications in that sense. I don’t believe that this solution has any other reasonable environmental implications.

### Ethical

There a few general ethical implications. The main one being that in this game police officers are the antagonists and act against the user with prisoners being the protagonists, this could cause a debate as to whether it is ethical to portray criminals as heroes. In terms of computing however there are no reasonable ethical implications.

### Social

Due to the individual nature of the game there should not really be any social implications. The game does have the ability to be played in teams so there could be an element of competition between users but other than this the solution has no social implications. Other than the aforementioned there are no other reasonable social implications.

### Economic

This game will be free and open source so there is no additional costing needed to play the game other than a working computer running windows. If I was to publish the software online an internet connection would also be required to access and download the software.

### Legal

The name was originally going to be “*Prison Break”* however, the name “Prison Break” is the intellectual property of 20th Century Fox Television and Original Film meaning that due to copyright infringement I had to change the name from “Prison Break” to “Prison Escape” due to the breach of The Copyright, Design and Patents Act.

The Copyright, Design and Patents Act may have some other implications on this project since Prison Escape is based and built on the Microsoft game version of the game Minesweeper – however my version ended up being different enough from the Microsoft version that it doesn’t infringe on any copyright or intellectual property owned by Microsoft. In terms of media I will created all the graphics myself so there was no implication of The Copyright, Design and Patents Act in this sense.

The project did not have any implications in terms of the Data Protection Act as no personal data that identifies one specific person is stored. The Computer Misuse Act should also not have any implications in this solution.

If the high-score table was networked that would have allowed the users to track their progress against their friends, colleagues or the world. This however would cause some legal complications due to the fact accounts would need to be stored and the Data Protection Act would come into play.

Other than the above my solution does not have any reasonable legal implications.

# Progress Diary

|  |  |
| --- | --- |
| Time | Work Carried Out |
| Hour 1 | Today I wrote my project proposal and carried out my feasibility study. This has allowed me to confirm that this project is manageable with the timescale and other constraints that have been placed upon me. |
| Hours 2 & 3 | Today I used my time creating and getting responses to the end-user survey. I feel like it is important to spend a bit of time on it to ensure that I understand the feedback that the end users give me so that I can build a project that is complete, functional and serves the end users. I analysed the user feedback from the surveys and also spoke to some of the users that completed the surveys to get a further understanding of what features they desired |
| Hour 4 | I created my project plan and my Gantt Chart. This allowed me to try and set some deadlines for different parts of the project to be completed in. Hopefully this will allow me to ensure that all aspects of my project are completed on time |
| Hour 5 | I used this past hour to draw up some initial functional and user requirements. Although these may change slightly when I do my requirements specification tomorrow I felt it was important to try and get a rough concise idea of what the outline requirements of my solution should be. |
| Hours 6 & 7 | Today I took the time to write up my requirements specification. There were a few additions to the initial requirements as I felt the initial requirements were not quite giving a complete overview of what my solution should be do. I also added some outputs to my initial requirements as I left them yesterday. |
| Hours 8 & 9 | I used this time to create a plan for how I should ensure that my project meets all the requirements set out in the requirements specification. Due to the fact that the project has not yet been designed this could vary slightly with the way the solution ends up being tested but should give an outline of what I need to ensure the solution can do. |
| Hour 10 | Today I mapped out the rough outlines of what the main inputs, processes and outputs of the solution should be. The will save me time when it comes to the implementation stage as I should already have a rough idea of what I should be inputting. |
| Hour 11 | I used this time to create the navigational structure and the procedural structure of the application. I decided that all forms should be accessible from all other forms. The structure diagram was created to illustrate the relationships between the procedures in the main two forms. |
| Hour 12 | Today I created the data flow – mapping out specifically what data should be passed between what procedures. I also decided that my navigational structure was not the most efficient or logical way for users to navigate the application so I decided to add some pop-up windows so that the game wouldn’t close when a user wanted to check the instructions or see the scoreboard. |
| Hour 13 | I created the design of the forms and user control today. I wanted to ensure that they were usable and easily navigated so I tried to make the buttons a good large size so that the users would be able to read them clearly. |
| Hour 14 | Today I mapped out the pseudocode for my solution. I did this for all four forms and the user control however only the game, scoreboard and user control were written in depth due to the lack of function of the other two forms. |
| Hours 15, 16, 17 & 18 | Over the past four hours I have been creating the user control. This contains all the possible designs for the cells and a number of these will make up the form. This was not to complicated as it mainly involved drawing simple shapes using methods and many of the lines of code were the same if not similar. |
| Hours 19 & 20 | I have spent the past couple of hours design the forms. I created some pixel art to make the forms visually appealing and also wrote the instructions. Due to the way visual basic works it was fairly straightforward to match my solution to my earlier created design. |
| Hour 21 | Today I finished off the instruction form design and wrote the code for the instruction form which was just links to the home and start game forms. |
| Hour 22 | In the past hour, I have been finishing the home screen design – I decided to make the title out of pixel art this is a minor deviation from the design but I did this in order to keep the design consistent throughout the application. I also wrote the code for the links from the homepage. |
| Hours 23, 24, 25 & 26 | I have been working on writing the algorithm that creates the board and allocates the location of the warning lights. I followed the design and pseudocode for this so there was not really any deviation from the design. I found that using the 2D array for the board was the most efficient. |
| Hours 27, 28 & 29 | Today I spent three hours creating the algorithm to decide what to do based on the two possible mouse clicks. This involved a little bit of research as I didn’t know the specific syntax of addressing button clicks however once I found this out it was relatively straightforward. |
| Hours 30, 31, 32 & 33 | Over the past couple of days, I have been working on the recursive clear algorithm and the other procedures (mainly export results) the recursive clear algorithm although not very long in terms of lines of codes requires a lot of thought due to the number of nested loops. I did eventually get this and the other procedures working. This allowed me to run the first version of the game and as I was testing the procedures as I went the game built and seemed to run fine. |
| Hour 34 | Today I created the files that the high scored tables would be stored in. I populated them with some example high scores so that I could check the algorithm as I tested. |
| Hours 35, 36, 37 & 38 | I spent the last four hours coding the scoreboard form. I spent the initial time trying to input the data from the files in the same procedure using different parameters but I could not get this to work so I just kept it the same as the design and had three inputs procedures and three output procedures. The reason this procedure took so long to write was due to an error in the sorting algorithm that I came across that I could not find – I then tried it with some different data and it worked fine so I assumed it was just some sort of execution error. |
| Hour 39 | I spend the first hour testing the scoreboard. I again, with the first set of data I tried found an error in the sort so I went through the code again to try and locate the source of the error. I then finally found the source of the error when I realised that I had the data type of player.playerTime to be a string instead of an integer. This was a simple error but it caused what I thought was a logic error so I spent a lot of time trying to fix this in the past few days. |
| Hours 40 & 41 | Today I tested the game. I made sure that the boards were the right size and that the correct number of mines were being placed. These tests were fine and no errors were found. I also tested the numbers and no errors were found here. The recursive clear algorithm was also tested and I found an error when using it on the far side of the board. This was due to a logic error – I was trying to minimise the amount of data being passed into the procedure by only checking the rows instead of the rows and columns but due to the fact that the expert board is a rectangle there are more columns than rows. This was easy to find and fix. |
| Hour 42 | I ensured that the rest of the games features were working correctly, such as the right click and restart button – all of these features were working correctly. |
| Hour 43 | I used this time to test the input validation and ensure that all data entered was reasonable. I found no errors with the input validation. |
| Hour 44 | In this final testing hour, I wrote the beta testers response form. Here I was mainly ensuring that the application was usable as I had been extensively testing the game to ensure that there were no errors. The end-users gave positive feedback on the usability of the form. I was also planning on doing some final testing as I thought I would have to make minor changes to my solution after the beta testing but this was not the case due the positivity of the responses that I received. |
| Hours 45, 46 & 47 | I have spent the past three hours doing the evaluation and pulling together the different sections of my project. I evaluated the solution and showed how it met the requirements set out in the requirements specification and also the quality of the development process as a whole. I evaluated my performance and the things I have learned throughout this process. The testing process was also evaluated and I found the testing to be quite successful. I also listed the possible implications of my project in all relevant areas however, there should not be to many implications of the project. |