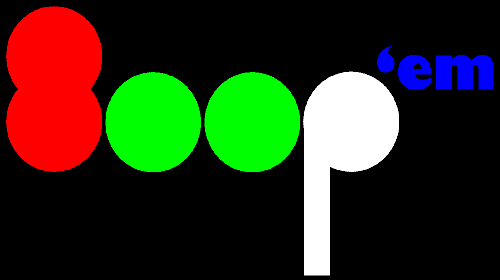
“Boop ‘em” Advanced Higher Computing Project Documentation



By Craig Methven

Contents Page

[Project Proposal: 5](#_Toc507459379)

[Outline of Project: 5](#_Toc507459380)

[End User: 5](#_Toc507459381)

[Knowledge and Skills Needed to Create the Program 5](#_Toc507459382)

[Stay within time frame: 6](#_Toc507459383)

[Ensure I will have the skills required to complete: 6](#_Toc507459384)

[Ensure that I have the resources available: 6](#_Toc507459385)

[Initial Research 7](#_Toc507459386)

[Survey of end users: 7](#_Toc507459387)

[Assessment of end user requirements: 8](#_Toc507459388)

[Copyright: 8](#_Toc507459389)

[Additional resources: 8](#_Toc507459390)

[Additional Techniques needed: 8](#_Toc507459391)

[Analysis of findings: 8](#_Toc507459392)

[Time allocation: 9](#_Toc507459393)

[Intermediate targets: 9](#_Toc507459394)

[Initial Project Plan 10](#_Toc507459395)

[On-going refinement: 10](#_Toc507459396)

[Specification 11](#_Toc507459397)

[Purpose of the Solution: 11](#_Toc507459398)

[Scope and boundaries: 11](#_Toc507459399)

[End users: 12](#_Toc507459400)

[User requirements: 12](#_Toc507459401)

[Functional requirements: 12](#_Toc507459402)

[Input: 12](#_Toc507459403)

[Outputs: 13](#_Toc507459404)

[On-going refinement: 13](#_Toc507459405)

[Testing 14](#_Toc507459406)

[How solution will be tested: 14](#_Toc507459407)

[How results of testing are recorded: 14](#_Toc507459408)

[Input Validation Testing: 18](#_Toc507459409)

[Test plan for end user testing: 18](#_Toc507459410)

[On-going refinement: 18](#_Toc507459411)

[Interface design: 19](#_Toc507459412)

[Wireframes: 19](#_Toc507459413)

[How each screen is related 21](#_Toc507459414)

[Purpose of each Element: 22](#_Toc507459415)

[Expected inputs and outputs: 23](#_Toc507459416)

[On-going refinement: 24](#_Toc507459417)

[Program/data structure Design: 25](#_Toc507459418)

[Variables and Files that I will use: 25](#_Toc507459419)

[Pseudo Code: 26](#_Toc507459420)

[Data flow between sub routines: 30](#_Toc507459421)

[On-going refinement: 30](#_Toc507459422)

[Implementation: 31](#_Toc507459423)

[The Interfaces Matching the Wireframe: 31](#_Toc507459424)

[On-going refinement: 36](#_Toc507459425)

[Final Testing: 37](#_Toc507459426)

[Testing all elements in test plan: 37](#_Toc507459427)

[Testing results: 41](#_Toc507459428)

[End User Testing: 42](#_Toc507459429)

[On-going refinement: 42](#_Toc507459430)

[Evaluation: 43](#_Toc507459431)

[Evaluating if the solution matched the requirement specification: 43](#_Toc507459432)

[Evaluating the final testing: 43](#_Toc507459433)

[Evaluating the end user testing: 43](#_Toc507459434)

[Further development: 44](#_Toc507459435)

[Conclusion about solution: 44](#_Toc507459436)

[Evaluation on development process: 44](#_Toc507459437)

[Evaluation on my performance: 45](#_Toc507459438)

# Project Proposal:

## Outline of Project:

I want to create a grid-based game which involves 2 players trying to remove the other player from the board.

The 2 players take it in turns to place a counter in a square on the 6x6 grid with them only being able to place in squares that have their counter or a blank square. You will be able to tell who owns which square as each player’s counters will have a different colour.

If the number of counters in a square is equal to that of the squares surrounding squares (on the 4 cardinal directions) then the counters spread to those surrounding squares adding one counter to those square. This spread will take over the surrounding boxes even if it is the other players this allows the players to take one another out of the game.

The interface of the main game would be the grid with an image in each square telling the user what player owns the box and how many counters is in it as well as a input for the player to decide which square to go into at the bottom.

There will also be a “high score sheet” which saves the amount of turns it took before a player won automatically. I would do a sort on this to put the highest scores at the top of the list.

## End User:

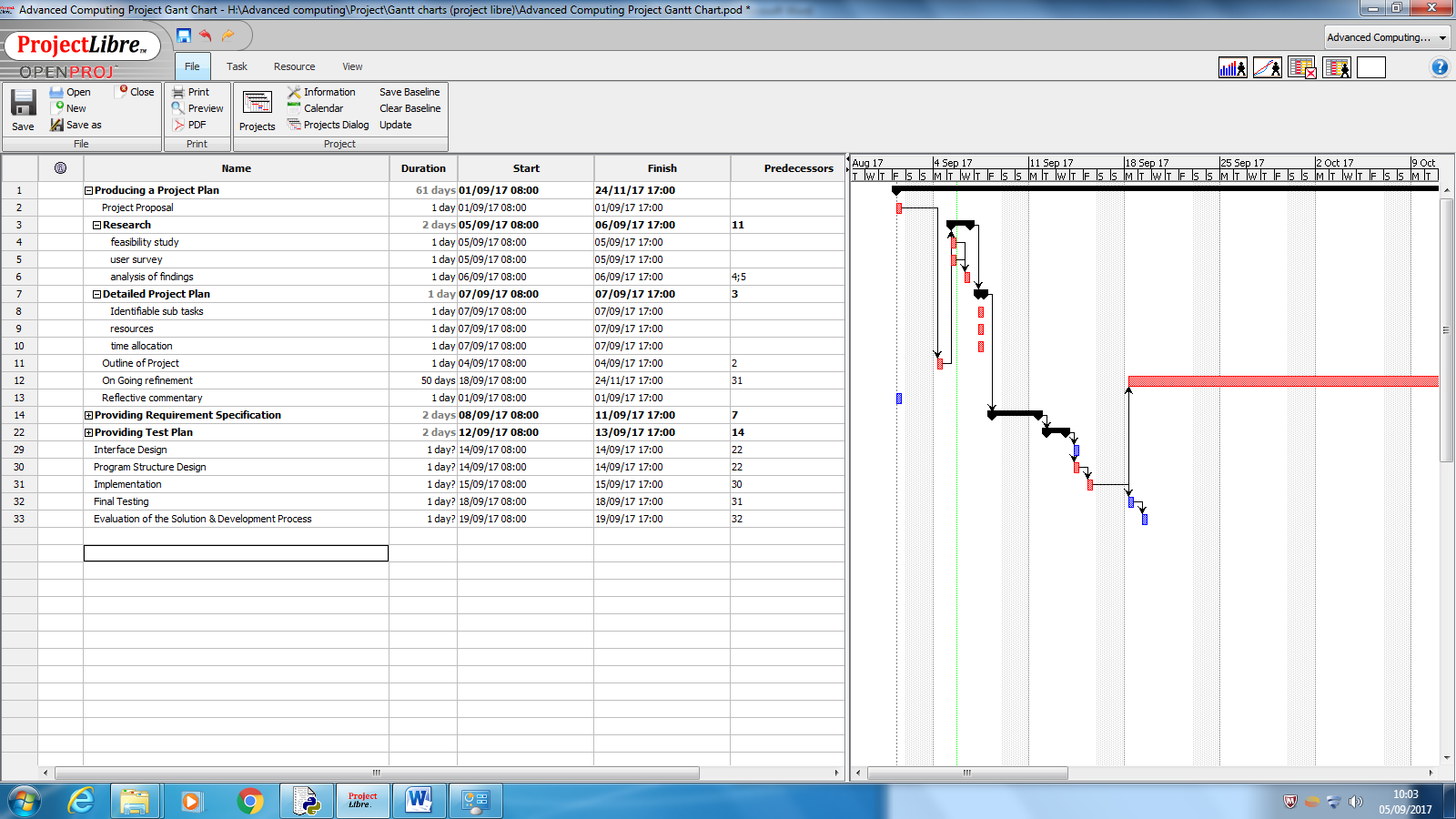
This target market for the game is novice users that have an interest in games. This is due to the simple way of inputting the data into the program (using text boxes), the simplistic style and the fact that it is a game. I think the average age of the users will be 12-18 as that is the age people are in high school so have lots of time to spend with friends which they could use to play this game.

## Knowledge and Skills Needed to Create the Program

For the program, I will need to use many programming constructs, algorithms and structured data types such as: reading and writing to files, sorting information in a file, 2D-arrays, recursion, queues and input validation. I will also need to increase my knowledge of Pythons basic code as I will need to use it to display information, collect data and do other basic features. So far, I am not fluent in Python so I will need to dedicate time to learning that in my free periods using online and school resources to help me to understand it. I will also need to do some graphic design for the interface so that the user knows what is happening on the grid. This will need me to create graphics for 1-3 counters of 2 colours (one for each player), I will also need to create a graphic for a grid.

## Stay within time frame:

I will use ProjectLibre (an open source program that allows the user to easily create Gantt charts and therefore monitor time management) to create a chart that informs me of how long I have to spend on doing certain aspects of the project so that I don’t spend too long on some aspects and run out of time for others therefore staying within the time frame. I will update the Gantt chart regularly to keep up with what I need to do and make any changes that need to be made. The chart will give me a clear sign if I need to speed up with what I am doing, which I will do if necessary or allocate more time in my day to work on it so I do not fall behind and miss the deadlines

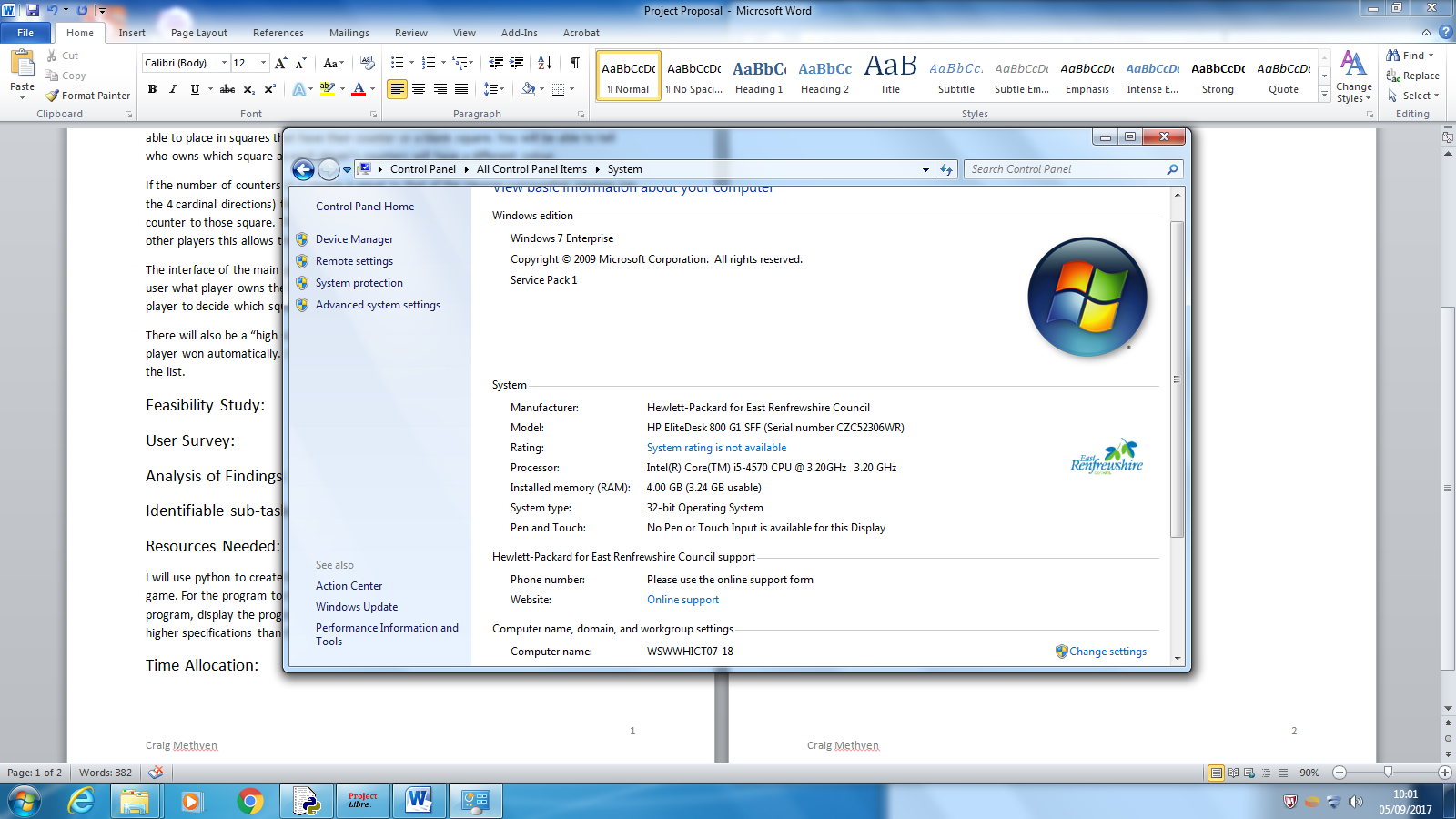


## Ensure I will have the skills required to complete:

I will use my free periods in school and spare time at home where necessary to learn new skills to ensure that I have the skills required to complete the project. This will require me doing research into Python (to become more comfortable with how it works and what it can do, as that is a new programming language for me), how to use Pygame (to create an interface) and how to save two variables to each square in a grid, as well as any other problems which I encounter when creating the program.

## Ensure that I have the resources available:

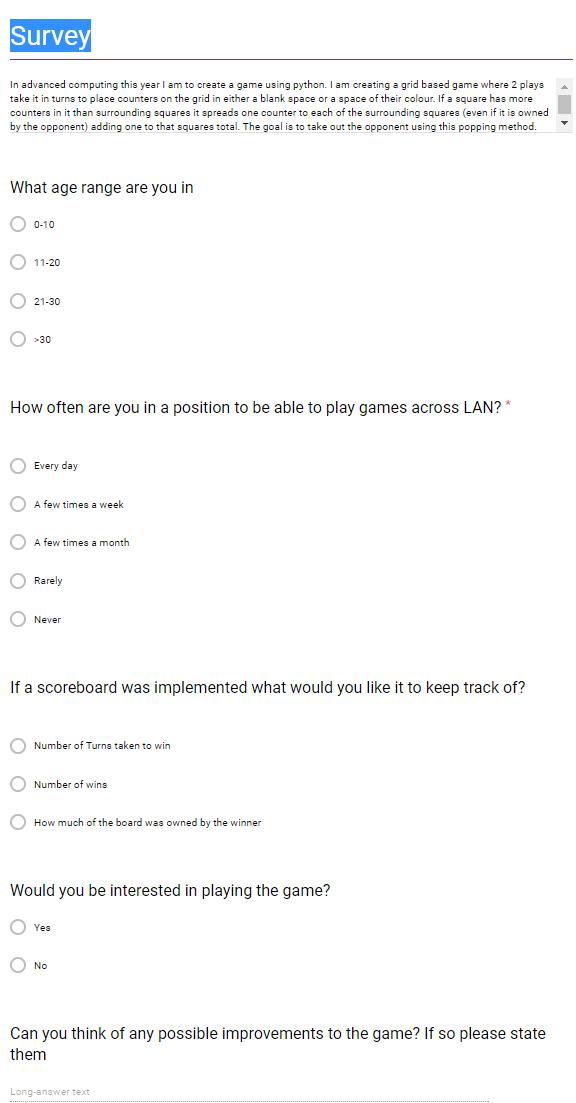
In terms of software, I will need IDLE and Pygame to be able to run my program. Pygame to create the interface and IDLE to create and run the code. IDLE is already on the computers at the school and Pygame if not already on the computers I’m sure I will be able to ask to get them added onto them. In terms of hardware, for the program to run a computer, monitor and keyboard will be needed to run the program, display the program and make inputs respectively. A computer with similar or higher specifications than this would work well. We have all of these at school so have all hardware requirements met.



# Initial Research

## Survey of end users:

I have created a survey using Google Forms which asks them various questions about the game I am proposing. These questions include their age, if they would be interested in playing the game, what they would like a scoreboard to track and how often they are able to play across LAN. The results of the survey are in a spread sheet attached.



## Assessment of end user requirements:

Users will require a computer, keyboard, mouse and monitor to play the game. During testing I am going to use the school computers which will work well with their specifications being stated previously.

## Copyright:

My game is not affected by copy right as it is not going to be stealing any code, sounds or graphics from companies. If I do want to use any copyrighted sounds or graphics later in the project I will ask the company if I am able to use them but I will most likely be making my own if needed. This project does not come under the Education section of the Copyright Design and Patents Act 1998 as that states that the work has to be \*“for the sole purpose of illustration for instruction” which mine is not as it is supposed to be played as a game not teach others how to code.

## Additional resources:

In terms of hardware in addition to a computer, a monitor, a mouse and keyboard I will need a microphone if I want to record any sound for the game and a way to listen to the sound. In terms of software I will most likely need an image editing and creating software which I can use paint for in school to create the graphics.

## Additional Techniques needed:

I have written out the pseudo code for the program that I am wanting to create and found that the only thing that I am not knowing how to do is print images onto the screen and create interfaces. I will have to look into that before I start programming. For this I will look into Pygame.

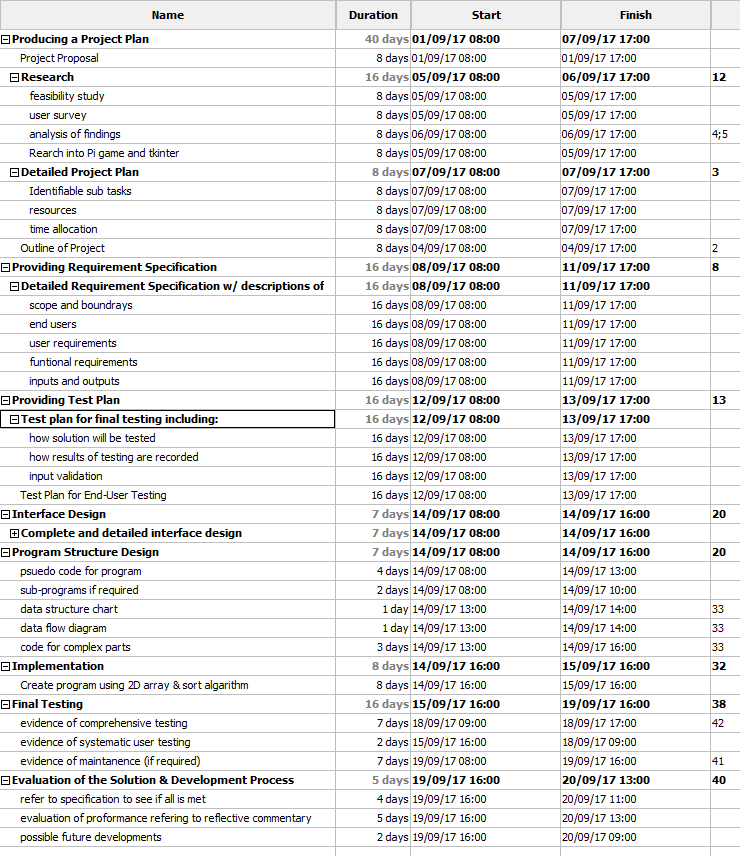
## Analysis of findings:

From the survey, I have found out that players are wanting the scoreboard to keep track of how much of the board is owned by the winner of the game when it ends and they want to be able to play the game with a grid size of their choice. Also, the vast majority of the people who would want to play the game aren’t often in a position to play games across LAN so if this feature is implemented it will be implemented last. Other than that, they are happy with how I described the game and can’t think of any other features they want implemented.

From writing out the pseudo code I found that I am capable of making the game from the knowledge I have learnt at advanced higher as long as I do research into how to create an interface with python and Pygame

## Time allocation:

I have created a Gantt chart using ProjectLibre to keep track of time. I will be updating it with how long it has taken me to do certain tasks to keep me on schedule and make sure I meet final deadlines.



## Intermediate targets:

I have put milestones into my Gantt chart so that I know what time I am do to certain tasks for and keep me on track. These will be updated as I get the dates to do so.



\*reference to “ <http://www.legislation.gov.uk/ukpga/1988/48/part/I/chapter/III/crossheading/education> ”

## Initial Project Plan

|  |  |  |
| --- | --- | --- |
| Task | Time | Target Date |
| Requirement specification | 12 Hours | 06/10/17 |
| Test plan | 4 Hours | 13/10/17 |
| Interface design | 5 Hours | 29/10/17 |
| Program design | 5 Hours | 05/11/17 |
| Implementation | 24 Hours | 22/11/17 |
| Final testing | 4 Hours | 3/12/17 |
| Evaluation | 6 Hours | 21/12/17 |
| Final Deadline | 60 Hours | 23/2/18 |

## On-going refinement:

I have added to the additional techniques needed by adding that I needed to know Pygame to be able to do the interface of the game. I’ve also updated the Gantt chart multiple times adding more milestones and updating it to meet my progress throughout the process. The various versions of the Gantt chart can be seen attached.

# Specification

## Purpose of the Solution:

The purpose of the solution is to entertain the user with a multiplayer game. This game will take in the user’s inputs into an, initially empty, 2D grid allowing the user to place in an empty square or one that they own adding one to that number of counters in that box and making it owned by that player. If the number of counters in the square matches the number of surrounding squares the box the counters will spread to those boxes making any counter in those boxes owned by the player who spread. The goal of the game will be remove the other player from the board. Once this is done it will create a document saving how much of the board is owned by the winning player.

## Scope and boundaries:

Scope: For the program to be complete it must:

* Collect the size of grid wanted and number of players wanted
* Collect and verify positions wanted to play by the user (inputted using the mouse) and display valid ones on a 2D grid and ask for the player to pick another grid squire if that one is invalid
* Check to see if the number of counter in a grid square is equal to that of the number of surrounding squares and if so splitting the counters up into those squares.
* Checking to see if one team has been eliminated and if so displaying the winner
* Create a sorted score sheet that stores that amount of the board owned by the winner of the game

Boundaries:

The main constraint of the program is time as there is a deadline to the program in the near future. I will initially add in all the features stated in the scope although if I have time left over I will add:

* Way to set a RGB colour for each team’s counters
* Way to set screen resolution
* Way to play it over LAN
* Select number of teams/players
* Keep track of multiple things on the score sheet such as the number of turns taken to win and number of wins
* Interactive tutorial of the game
* Noises when counter gets shot out

## End users:

From the user survey I found out that the vast majority of the potential users are in the age range of 11-20 with 50% of the people being in that age range and all of them saying that they would want to play it. Although, also all the respondents in the age range of 21-30 said that they would be interested in playing the game showing that they are also a potential target for the game.

Therefore, the likely end users of the game will be between the ages of 11-20 and like to play video games.

## User requirements:

The final users will require a visual of the 2D grid with the correct positions of each players counters which are easy to identify as each player by a unique colour. Counters will need to be easily placed on the grid. They will also require an options menu to select how many players they are playing with and the size of the grid they want.

## Functional requirements:

For the program to function it will at minimum need a 2D grid, a way for multiple players to enter counters onto the board, verify if a counter can be placed where the user is wanting to place it, a way to check when counters should spread and a way to check who wins.

## Input:

|  |  |
| --- | --- |
| Input | Description |
| Width of Grid | Inputted via the options menu using the keyboard number keys it determines how many squares wide the grid that the user plays on is. It will be within the range of 3-12. |
| Height of Grid | Inputted via the options menu using the keyboard number keys it determines how many squares high the grid that the user plays on is. It will be within the range of 3-12. |
| Square the user places the counter | Inputted via the game screen using a mouse click it places a counter on the grid in the position clicked. It will only place a counter if the player who’s turn it already has a counter in that square or the square doesn’t already have a counter in it. |
| Number of Players | Inputted via the options menu using the keyboard number keys it determines how many players are playing the game. It will be within the range of 2-8. |
| Nickname | Inputted via the winning screen it takes a keyboard input and saves that in the score file along with the score the user got. It will be at least 2 letters long and be up to 5 letters long. |
| Home button | Inputted via the options menu and scores screen using a mouse click it takes the user to the main menu if they click the button. |
| Play game button | Inputted via the main menu using a mouse click it takes the user to the game if they click the button. |
| Options menu button | Inputted via the main menu using a mouse click it takes the user to the options menu if they click the button. |
| Scores button | Inputted via the main menu using a mouse click it takes the user to the scores screen if they click the button. |

## Outputs:

|  |  |
| --- | --- |
| Output | Description |
| Counters positions on the grid | On the games screen the number of counters in each box will be displayed along with the grid that it is on. The counters will also have different colours depending on which team placed them with a unique colour for each team with a maximum of 3 counters per box. |
| The winner of the game | On the winning screen the winning players number will be displayed in the colour of their counters. |
| Scores table | On the scores menu the top 10 scores will be displayed along with the names of who got each score. |
| The grid size | On the options menu the grid size will be displayed showing the user the grid size that will be used if they continue with these options. This is true for both the width and height of the grid. |
| The number of player | On the options screen the number of players will be displayed showing the user how many players will be in the game if they continue with those options. |

## On-going refinement:

I have changed the input and outputs for the game as I forgot that I would need to have an options menu to be able to have the options of changing the grid size and the number of players

I then separated the input and output table into two separate tables and added a descriptions column to show what each of the inputs and outputs do.

# Testing

## How solution will be tested:

I will be running the program and seeing if the correct output is given visually on the screen and if not start using print statements to see what is saved in variables and arrays to see what caused the problem and try to resolve it. I will save the testing that I have done in a table and save screenshots of the testing. I will do this for all actions that could occur in the program: counters spreading from a box surrounded by 2 boxes, 3 boxes and 4 boxes, closing the pages, changing the resolution, changing the size of grid, changing the colour, the options menu is validated correctly, changing the number of teams and input validation of the winner’s name.

## How results of testing are recorded:

I will use tables to record the results of the testing. Screenshots will be recorded along with the table. The tables will look as shown:

Spreading of 2 counters:

|  |  |  |
| --- | --- | --- |
| Corner | Does it work as intended? | Screen shot number as proof |
| Top left |  |  |
| Top right |  |  |
| Bottom left |  |  |
| Bottom right |  |  |

Spreading of 3 counters:

|  |  |  |
| --- | --- | --- |
| Edge | Does it work as intended? | Screen shot number as proof |
| Top edge |  |  |
| Right edge |  |  |
| Left edge |  |  |
| Bottom edge |  |  |

Spreading of 4 counters:

|  |  |  |
| --- | --- | --- |
| Square (X cord, Y cord) | Does it work as intended? | Screen shot number as proof |
| 2,2 |  |  |
| 2,3 |  |  |
| 3,2 |  |  |
| 3,3 |  |  |

Closing each page:

|  |  |  |
| --- | --- | --- |
| Page | Does it work as intended? | Screen shot number as proof |
| Game |  |  |
| Instructions |  |  |
| Winner |  |  |
| Main Menu |  |  |
| Scores |  |  |
| Options |  |  |

Changing the resolution:

|  |  |  |  |
| --- | --- | --- | --- |
| Inputted Resolution | Resolution Used | Does it work as intended? | Screen shot number as proof |
| 100x100 |  |  |  |
| 800x400 |  |  |  |
| 800x800 |  |  |  |
| 100x800 |  |  |  |

Changing the grid size:

|  |  |  |  |
| --- | --- | --- | --- |
| Inputted Grid Size | Grid Size used | Does it work as intended? | Screen shot number as proof |
| 3x3 |  |  |  |
| 5x5 |  |  |  |
| 9x5 |  |  |  |
| 12x12 |  |  |  |

Changing the colours:

|  |  |  |  |
| --- | --- | --- | --- |
| Inputted Colours | Colours used | Does it work as intended? | Screen shot number as proof |
| Player 1: Red  Player 2: Orange  Player 3: Pink  Player 4: Purple  Player 5: Light Blue  Player 6: Dark Blue  Player 7: Dark Green  Player 8: Lime Green |  |  |  |
| Player 1: White  Player 2: Dark Blue  Player 3: Lime Green  Player 4: Purple  Player 5: Dark Green  Player 6: Orange  Player 7: Light Blue  Player 8: Yellow |  |  |  |
| Player 1: Yellow  Player 2: Pink  Player 3: Light Blue  Player 4: Orange  Player 5: Red  Player 6: Dark Blue  Player 7: Lime Green  Player 8: White |  |  |  |
| Player 1: White  Player 2: Red  Player 3: Purple  Player 4: Yellow  Player 5: Dark Green  Player 6: Light Blue  Player 7: Orange  Player 8: Lime Green |  |  |  |

Changing the number of teams:

|  |  |  |  |
| --- | --- | --- | --- |
| Inputted Number of Players | Outputted Number of Players | Does it work as intended? | Screen shot number as proof |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |

Validation on the options menu:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Inputted Number | Field inputted into | Output | Does it work as intended? | Screen shot number as proof |
| 2 |  |  |  |  |
| 20000 |  |  |  |  |
| 1 |  |  |  |  |
| 3000 |  |  |  |  |
| 1 |  |  |  |  |
| 100 |  |  |  |  |
| 2 |  |  |  |  |
| 50 |  |  |  |  |
| 0 |  |  |  |  |
| 35 |  |  |  |  |

Validation of winners input:

|  |  |  |  |
| --- | --- | --- | --- |
| Name inputted | What happened? | Does it work as intended? | Screen shot number |
| I |  |  |  |
| +PO |  |  |  |
| Crazy |  |  |  |
| Jeremy |  |  |  |

## Input Validation Testing:

Inputs that need validating:

* Resolution size chosen (equal to or less than 2560 x 1440 but greater than 100x 100)
* Grid width chosen (equal to or less than 12 but greater than or equal to 3)
* Grid height chosen (equal to or less than 12 but greater than or equal to 3)
* Number of teams (greater than 1 and less than or equal to 8)
* The mouse clicks on the games screen (to check that the user either is trying to place a counter in a square that they own or a square that has no counters in it)
* The winner’s username (equal to or longer than 2 characters in length and less that or equal to 5 characters in length).
* The mouse clicks on the scores and options menu (to make sure that the user has clicked on the back button and no where else on the screen)

## Test plan for end user testing:

I will survey people about how they found the game and if they found any errors with the game. I will then look at what people thought and makes improvements and changed based on what they have said. As well as this, from this information I will determine if the game is successful in its goal to entertain the users. Some of the questions I will ask are “Did you enjoy the game?” and “What do you think could be added to improve the game?”. I will try to get at least 10 people to test the game to get a variety of responses and see what people really think about the game.

## On-going refinement:

I added all the tables in to show what values I would be using during each test and specifically what fields I would have in each table. I also changed the upper and lower limits of the input validation as the other ones didn’t make as fun of a game. I also added more parts to input validation as I missed out some vital screens that needed validating. I also changed the test plan for end user testing to more clearly state what I am wanting to find out and how many people I will try to get to test it.

# Interface design:

## Wireframes:

Main Menu:

Title

Start Game button that launches the game

Link to options menu

Instructions

Scores

Options menu:

Title of Page

Resolution wanted (2 input boxes)

Grid size wanted 2 (input boxes)

Number of teams (input box)

Back button (links to main menu)

Player names the colour written in the colour the player currently is

Titles of inputs

Colour selector

Game screen:

Grid centred on screen

Instructions:

Title

Instructions Text

Back Button (links to main menu

Scores Menu:

Title

The top 10 scores

Back Button (links to main menu

Winning Screen:

Number of Player who won

Text input for winning players name (once entered links back to main menu). The name must be longer than 1 letter and less than 6

How each screen is related:

Options Menu

Scores Menu

Main Menu

Instructions

Winning Screen

Game screen

When the program will load the main menu when first ran. From this the user can go to the options menu, instructions, scores menu or to the game screen. The options menu, instructions, and scores menu can all go back the main menu. The game screen needs to go through the winning screen to go back to the main menu.

## Purpose of each Element:

Main Menu:

Title: To say the games name

Link to Game: To allow the user go to the game

Link to Options Menu: To allow the user to go to the options menu

Link to Instructions: To allow the user to go to the instructions menu

Link to Scores: To allow the users to go to the scores menu

Options Menu:

Title: To tell the user where they are

Input Box looking for the number of teams: To allow the user to change the number of teams playing the game

Input box looking for the resolution: To allow the user to play the game at different resolutions

Input box asking for the grid size: To allow the user to play the game with grids of different sizes

Back button: To allow the user to go back to the main menu to play the game

Game Screen:

Centred grid: To allow the user to make inputs and see what the board looks like at any given point

Scores:

Back button: To allow the user to go back to the main menu to play the game

Title: To tell the user where they are

Scores: So that they know what the highest scores are and who has them

Winning Screen:

Text saying winner: To let the users see who won the game

Text input: To allow the user to input the name which the score was under

Instructions:

Titles: So that the user knows where they are

Instructions text: So that they know how to play the game

Back button: To allow the user to return to the main menu

## Expected inputs and outputs:

Title screen:

For inputs I expect mouse clicks on one of 4 positions. No validation is needed as the mouse clicks will only work if on the buttons.

The output of the different titles of the screens you could go to as well as the title.

Options menu:

For inputs I expect mouse clicks on different elements five of which allows the user to type into a text box numbers. The validation will be that they have entered numbers within a certain limit. The limits in the number of teams is that it must be greater than one and less than or equal to 8, for the x resolution it has to be greater than 100 and less than 2500, for the y resolution it has to be greater than 100 and less than 1400 and for grid size each have to be greater than or equal to 3 and less than or equal to 12. If the input clicks on a colour the colour is saved although no validation is needed. The back button doesn’t need validation.

I expect it to output the options that were set before the user started changing them.

Game screen:

I expect only mouse clicks either on the grid or on the back button. If they click on the grid validation is needed to check to see if the grid space clicked on is occupied and if so if it is occupied by them and if not, then the input will not to have an effect. If the grid is empty though or occupied by that players team then the input will go through.

For outputs I expect a counter to appear where the user clicks and if they click on a counter which they own a second counter to appear in that box if it is not on a corner. If it is on a corner then I expect to see an extra counter in the boxes surrounding said corner. If the user clicks on a box with two counters, that they own, I expect to see three counters appear in that box unless it is on an edge then I expect to see one extra counter in each of the surrounding boxes. If the user places a fourth counter in a box then I expect to see an extra counter in each of its surrounding boxes.

Scores:

The only input expected is the user clicking on the back button which will lead them to the main menu. This doesn’t need validated

The output expected is the top ten high scores which will need a sorting algorithm to find the top ten scores, and hence display them.

Instructions:

The input expected is them to click on the back button which would lead them to the main menu

Output of the instructions text.

Winning Screen

The expected input is that the user clicks on the input box, types their username and presses enter. The username will be validated to be a minimum of 2 letters long and max of 5.

The output is the number of the player who won in the colour that their counters were.

## On-going refinement:

I added the additional screens (including the instructions menu) and a few that I had forgotten (such as the winners screen and the score screen) about. I also changed the interface of the options menu to look cleaner and also match the other inputs I’m collecting that where in the boundaries.

# Program/data structure Design:

## Variables and Files that I will use:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| teamGoing | variable | The number of the team that is going |
| teamColour | array | The colours used by each player |
| resolution | array | Made up of the X and Y coordinates it defines what the resolution of the screen is |
| gridSize | array | Made up of the X and Y of the grid, it defines what size each aspect of the grid should be |
| grid | 2D array with 3 elements one of which is an array | It stores the X and Y coordinate of each point in the grid and, if a player owns it, the number of the player who owns it and the number of counters in the square. |
| EdgeXStart | variable | A distance calculated for how far away the grid should be from the edge of the screen on the X coordinate |
| EdgeXStart | variable | A distance calculated for how far away the grid should be from the edge of the screen on the Y coordinate |
| Queue | queue | Stores the coordinates of the points that need a counter added to them |
| BoxHW | variable | Once calculated stores the height and width each of the square boxes |
| winner | boolean | Saves if a winner has been found or not |
| spacesOwned | variable | Once a game is one saves the amount of spaces the winner has a counter in. It is used as the score. |
| InputName | variable | Stores the name inputted by the winning player to write it to the document |

## Pseudo Code:

This is the pseudo code for the main sub routines of the program:

Finding the height and width that the boxes should be:

1. Max Box Width = The X coordinate of the resolution divided by the X coordinate of the grid size

2. Max Box Height = The X coordinate of the resolution divided by the X coordinate of the grid size

3. If the Max Box Height is greater than the Max Box Width then do the following

4 Box Width = Max Box Width

5. Else

6. Box Width = Max Box Height

7. End the if statement

Creating the 2D array used to store what team owns each box and how many squares are in each box

1. Define grid as a 2D array with the first component being equal to the number of boxes along the width of the grid and the seconds component being equal to the number of boxes along the height of the grid

2. loop for the number of times equal to the number of boxes along the width of the grid and keep track of the number of times looped using a variable called i

3. loop for the number of times equal to the number of boxes along the height of the grid and keep track of the number of times looped using a variable called counter

4. Save [0,0] to the array grid in the position [i][counter]

5. end loop

6. end loop

Finding the start and end points to display the grid:

1. Save to variable XOverflow the number that is equal to the width of the resolution divided by the width of the grid

2. Save to variable YOverflow the number that is equal to the height of the resolution divided by the height of the grid

3. If XOverflow is equal to YOverflow do the following

4. Save the starting edge of the X to be 0

5. Save the starting edge of the Y to be 0

6. Save the ending edge of the X to be the X part of the resolution

7. Save the ending edge of the Y to be the Y part of the resolution

8. Else

9. Save to the starting edge of the X to be the X part of the resolution minus the X part of the grid multiplied by the box width divided by 2

10. Save the starting edge of the Y to be the Y part of the resolution minus the Y part of the grid multiplied by the box width divided by 2

11. Save the ending edge of the X to the X component of the grid multiplied by the width of the boxes plus the starting edge of the X

12. Save the ending edge of the X to the X component of the grid multiplied by the width of the boxes plus the starting edge of the X

13. End if

Changing the team:

1. Add one to the variable team going which stores which team is going

2. If team going is greater than the number of teams then

3. make team going equal to 1

4. end if

5. if each player has had a turn then

6. if the player who’s turn it is meant to be still has a counter on the board then

7. Use that number as the team going

8. else

9. Run this sub routine again

10. end if

11. end if

The popping mechanism:

1. If the counter being processed was placed of a corner then

2. check to see if that corner now has 2 or more counters in it and if so

3. Add a counter to the 2 surrounding squares and add those squares to the queue to be processed

4. set the number of counters in the square the counter was placed in to 0

5. end if

6. end if

7. If the counter being processed was placed of an edge then

8. check to see if that edge now has 3 or more counters in it and if so

9. Add a counter to the 3 surrounding squares and add those squares to the queue to be processed

10. set the number of counters in the square the counter was placed in to 0

11. end if

12. end if

13. check to see if square the counter has been placed in has at least 4 counters and if so

14. Add a counter to the 4 surrounding squares and add those squares to the queue

15. set the number of counters in the square the counter was placed in to 0

16. end if

Sorting the scores form the scores file:

1. Open the file

2. Loop the same number of times as the length of the file

3. Read a line of the file and save it to a temporary variable

4. Take away the last characters from the temporary variable to remove the /n

5. append the temporary variable to the array Scores

4. end loop

5. loop the same number of times as the length of the file and have the variable counter keep track of how many times it has looped

6. Split apart the string stored under the array Scores in slot counter, and save each part to a 2D array called splitter to have the name and the score separately

7. end loop

8. loop the same number of times as the length of the file and have the variable counter keep track of how many times it has looped

9. Set the variable sortedItem to the number in the array splitter equal to that of the number of times the list has been looped

10. set the variable sorterSpot to counter

11. Loop while the integer saved in the array behind sortedItem is less than sortedItem and sorterSpot is greater than 0

12. Swap name and integer in the array splitter that stores the value of sortedItem and the name and integer behind it.

13. subtract one from sorterSpot

14. End while

15. End loop

16. Loop 10 times keeping track of the number of times it has looped in the variable counter

17. Print the name and integer stored in splitter in the space counter

18. End loop

## Data flow between sub routines:

|  |  |  |
| --- | --- | --- |
| Sub routine name | Inputs | Outputs |
| TheGame | None | None |
| MainMenu | None | Pick |
| Options | None | None |
| Instructions | None | None |
| Scores | None | None |
| makingTheResolutionXSideTheLargerOne | Resolution | Resolution |
| makingTheXSideTheLargerOne | gridSize | gridSize |
| FindBoxHW | Resolution, gridSize | BoxHW |
| Create2DArray | gridSize | Grid |
| StartPoints | EdgeXStart,EdgeYStart,EdgeXEnd,EdgeYEnd | Resolution, BoxHW |
| DrawGrid | BoxHW,EdgeXStart,EdgeYStart,EdgeXEnd,EdgeYEnd,Display,teamColour,teamGoing,gridSize,teams | None |
| Winner | grid,gridSize,turn,teams,teamGoing | Winner |
| ChangeTeam | teamGoing | teamGoing |
| teamChecker | teamGoing, teams, grid, gridSize, turn | teamGoing |
| InputPos | BoxHW,EdgeXStart,EdgeYStart,grid,teamGoing,gridSize | addToQueue |
| QueueAdding | addToQueue, Queue | Queue |
| ChangeGrid | Queue,teamGoing,EdgeXStart,EdgeYStart,BoxHW,Display,grid,teamColour,turn,stop | addToQueu |
| OwnedCounters | gridSize,grid | TotalOwned |
| winnerInput | teamColour, teamGoing | winningName |
| SaveToFile | winningName,spacesOwned | A tab added to the scores output |
| collectInput | workingOn, limit | workingOn |
| writing | txtInput, Size, colour | None |
| readOptionsDoc | None | resolutionX,resolutionY,gridSizeX,gridSizeY,numberOfPlayers,colour |
| writeOptionsDoc | RX,RY,GX,GY,NP,colour | None |

## On-going refinement:

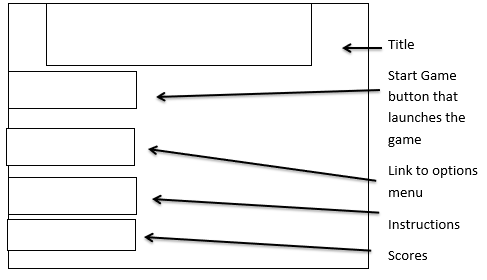
I edited the data flow as I added more sub routines into the program to add more features from the boundaries. I also added more pseudo code as I forgot to do the pseudo code for centring the grid.

# Implementation:

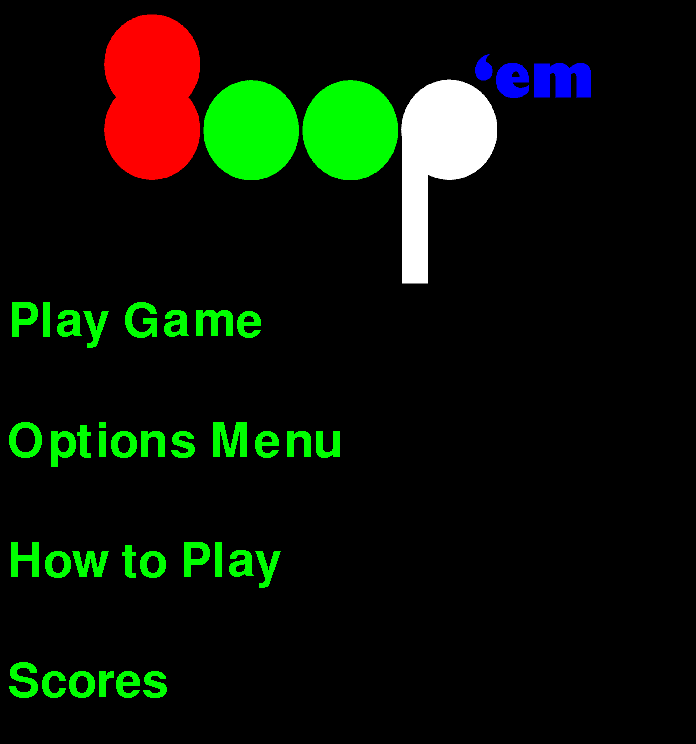
## The Interfaces Matching the Wireframe:

Main Menu:

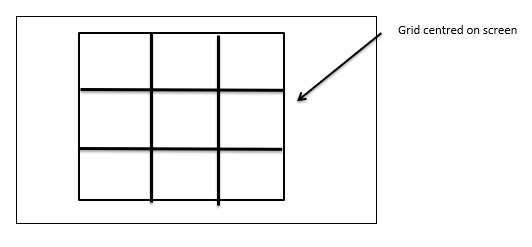
wireframe

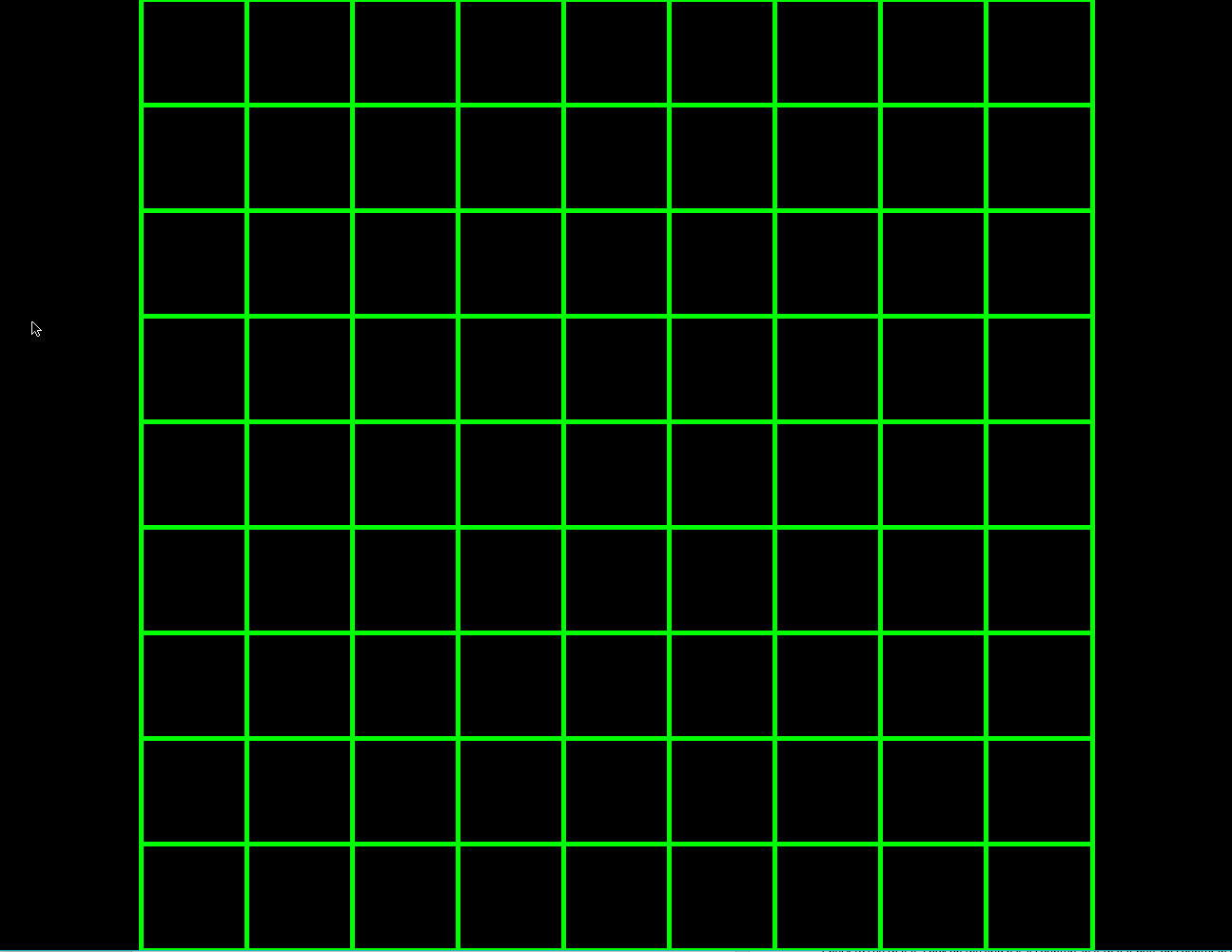


Interface:



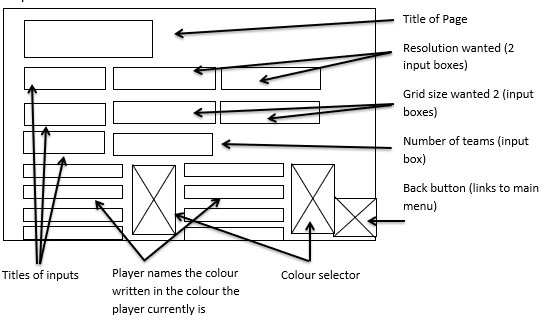
Game:

Wireframe:Interface:

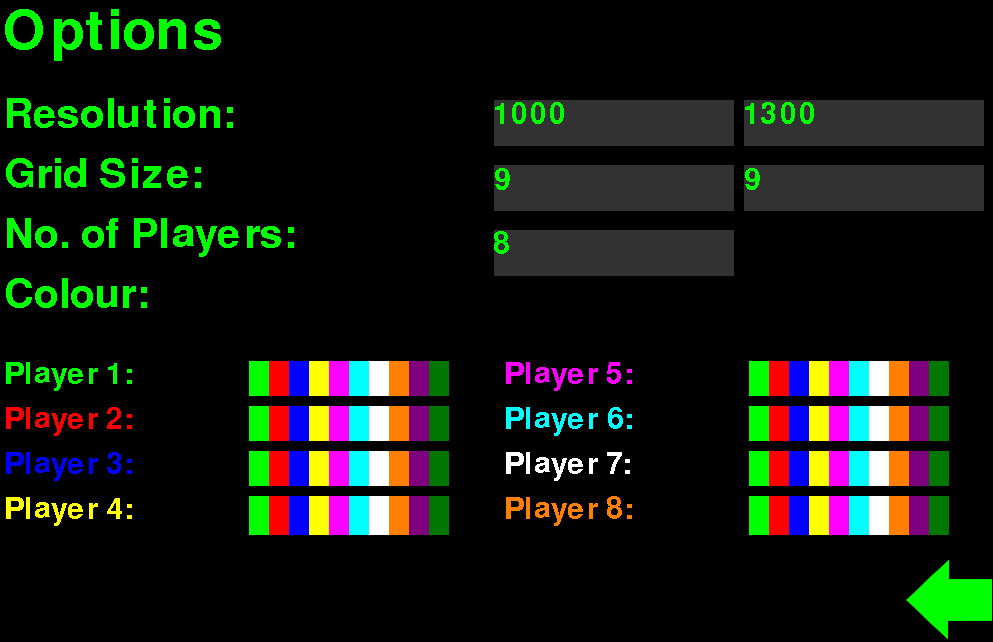


Options Menu:

Wireframe:

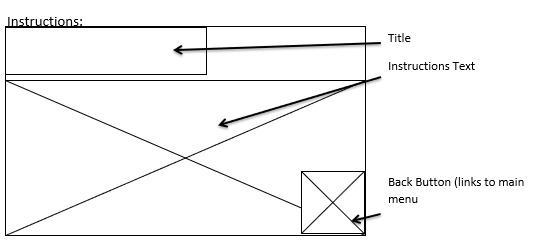


Interface:

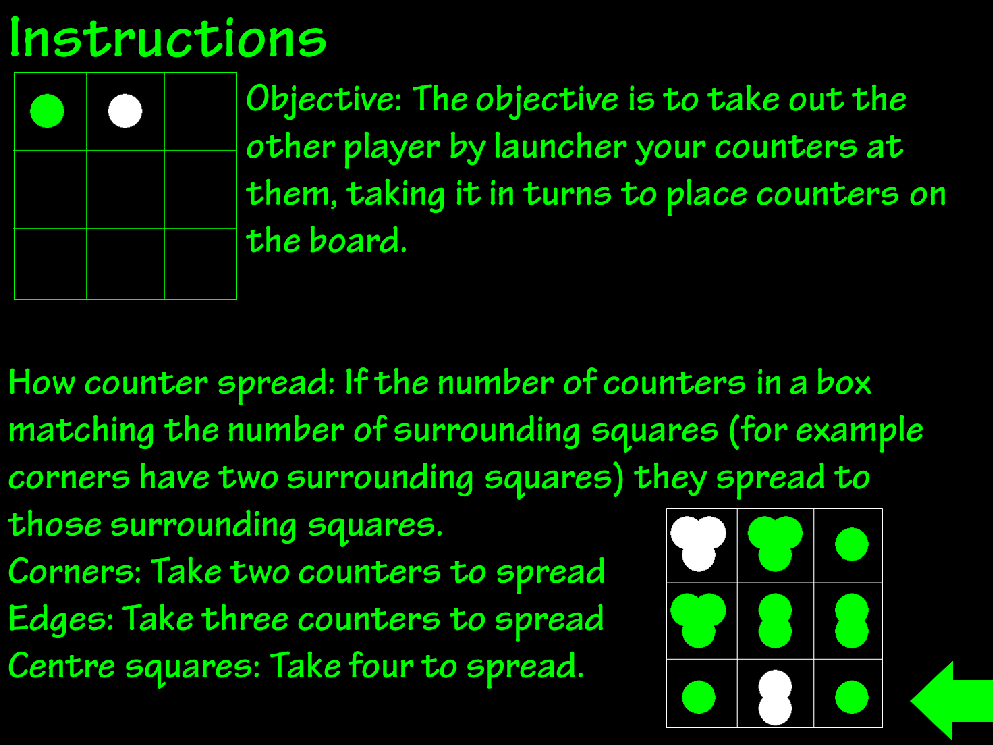


Instructions Menu:

Wireframe:

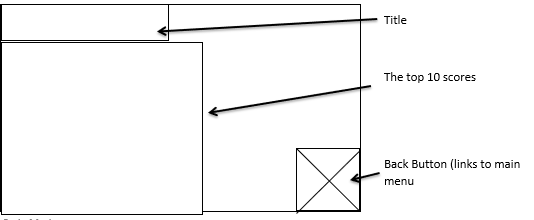


Interface:

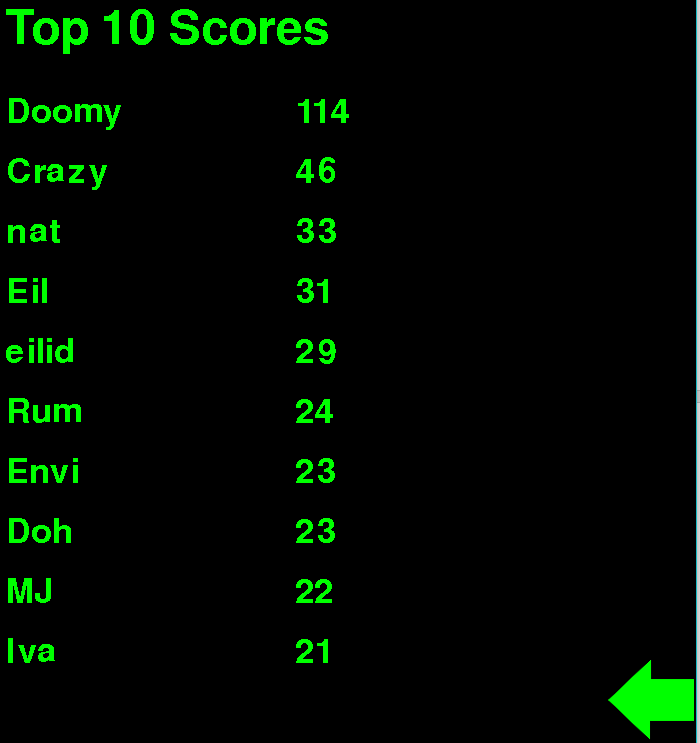


Scores Menu:

Wireframe:

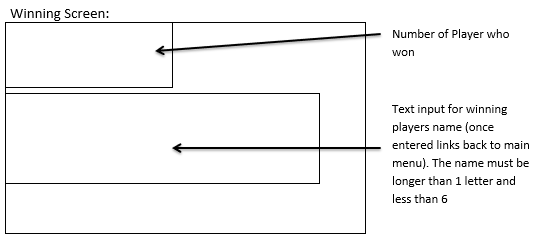


Interface:



Winning screen:

Wireframe:



Interface:



Each of the interfaces match the wireframes correctly and have the functionality required of them.

The code written is attached.

## On-going refinement:

I didn’t change this section as it was done relatively near the end and I was happy with it. The code I have made small changed to though since getting end user responses.

Final Testing:

## Testing all elements in test plan:

Spreading of 2 counters:

Checking to see if the counters on the corners spread correctly, adding one counter to each of the boxes beside them once the corner has 2 counters.

|  |  |  |
| --- | --- | --- |
| Corner | Does it work as intended? | Screen shot number as proof |
| Top left | Yes | 1 |
| Top right | Yes | 2 |
| Bottom left | Yes | 3 |
| Bottom right | Yes | 4 |

Spreading of 3 counters:

Checking to see if the counter on the edges spread correctly, adding one counter to each of the boxes beside them once an edge has 3 counters

|  |  |  |
| --- | --- | --- |
| Edge | Does it work as intended? | Screen shot number as proof |
| Top edge | Yes | 5 |
| Right edge | Yes | 6 |
| Left edge | Yes | 7 |
| Bottom edge | Yes | 8 |

Spreading of 4 counters:

Check to see that if a box on the grid has 4 counters in it that it puts on counter in each of the four surrounding squares. These are a small selection of squares to make sure it works

|  |  |  |
| --- | --- | --- |
| Square (X cord, Y cord) | Does it work as intended? | Screen shot number as proof |
| 2,2 | Yes | 9 |
| 2,3 | Yes | 10 |
| 3,2 | Yes | 11 |
| 3,3 | Yes | 12 |

Closing each page:

Check to see if pressing the X button (close window button) closes the window and program on each screen.

|  |  |  |
| --- | --- | --- |
| Page | Does it work as intended? | Screen shot number as proof |
| Game | Yes | 13 |
| Instructions | Yes | 14 |
| Winner | Yes | 15 |
| Main Menu | Yes | 16 |
| Scores | Yes | 17 |
| Options | Yes | 18 |

Changing the resolution:

Check to see if the resolution inputted into the options menu is the one used in the game

|  |  |  |  |
| --- | --- | --- | --- |
| Inputted Resolution | Resolution Used | Does it work as intended? | Screen shot number as proof |
| 100x100 | 100x100 | Yes | 19 |
| 800x400 | 800x400 | Yes | 20 |
| 800x800 | 800x800 | Yes | 21 |
| 100x800 | 100x800 | Yes | 22 |

Changing the grid size:

Check to see if the grid size inputted into the options menu is the one used in the game

|  |  |  |  |
| --- | --- | --- | --- |
| Inputted Grid Size | Grid Size used | Does it work as intended? | Screen shot number as proof |
| 3x3 | 3x3 | Yes | 23 |
| 5x5 | 5x5 | Yes | 24 |
| 9x5 | 9x5 | Yes | 25 |
| 12x12 | 12x12 | Yes | 26 |

Changing the colours:

Changing the colours for each player and seeing if those colours are used in the game

|  |  |  |  |
| --- | --- | --- | --- |
| Inputted Colours | Colours used | Does it work as intended? | Screen shot number as proof |
| Player 1: Red  Player 2: Orange  Player 3: Pink  Player 4: Purple  Player 5: Light Blue  Player 6: Dark Blue  Player 7: Dark Green  Player 8: Lime Green | Player 1: Red  Player 2: Orange  Player 3: Pink  Player 4: Purple  Player 5: Light Blue  Player 6: Dark Blue  Player 7: Dark Green  Player 8: Lime Green | Yes | 27 |
| Player 1: White  Player 2: Dark Blue  Player 3: Lime Green  Player 4: Purple  Player 5: Dark Green  Player 6: Orange  Player 7: Light Blue  Player 8: Yellow | Player 1: White  Player 2: Dark Blue  Player 3: Lime Green  Player 4: Purple  Player 5: Dark Green  Player 6: Orange  Player 7: Light Blue  Player 8: Yellow | Yes | 28 |
| Player 1: Yellow  Player 2: Pink  Player 3: Light Blue  Player 4: Orange  Player 5: Red  Player 6: Dark Blue  Player 7: Lime Green  Player 8: White | Player 1: Yellow  Player 2: Pink  Player 3: Light Blue  Player 4: Orange  Player 5: Red  Player 6: Dark Blue  Player 7: Lime Green  Player 8: White | Yes | 29 |
| Player 1: White  Player 2: Red  Player 3: Purple  Player 4: Yellow  Player 5: Dark Green  Player 6: Light Blue  Player 7: Orange  Player 8: Lime Green | Player 1: White  Player 2: Red  Player 3: Purple  Player 4: Yellow  Player 5: Dark Green  Player 6: Light Blue  Player 7: Orange  Player 8: Lime Green | Yes | 30 |

Changing the number of teams:

Checking to see if the number of teams inputted into the options menu matches the number of players in the game

|  |  |  |  |
| --- | --- | --- | --- |
| Inputted Number of Players | Outputted Number of Players | Does it work as intended? | Screen shot number as proof |
| 2 | 2 | Yes | 31 |
| 3 | 3 | Yes | 32 |
| 4 | 4 | Yes | 33 |
| 5 | 5 | Yes | 34 |
| 6 | 6 | Yes | 35 |
| 7 | 7 | Yes | 36 |
| 8 | 8 | Yes | 37 |

Validation on the options menu:

Check to see if the maximum and minimum values in the option menus work on the values you input from the keyboard (resolution x and y coordinate, grid size x and y coordinate, number of players). Negative numbers cannot be tested as it doesn’t accept any input other than numbers.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Inputted Number | Field inputted into | Output | Does it work as intended? | Screen shot number as proof |
| 2 | Resolution X | 100 | Yes | 38 |
| 20000 | Resolution X | 2560 | Yes | 39 |
| 1 | Resolution Y | 100 | Yes | 40 |
| 3000 | Resolution Y | 1440 | Yes | 41 |
| 1 | Grid Size X | 3 | Yes | 42 |
| 100 | Grid Size X | 12 | Yes | 43 |
| 2 | Grid Size Y | 3 | Yes | 44 |
| 50 | Grid Size Y | 12 | Yes | 45 |
| 0 | Number of Players | 2 | Yes | 46 |
| 35 | Number of Players | 8 | Yes | 47 |

Validation of winners input:

Check to see if the user names the user inputs are validated correctly (being at least 2 letters long and no longer than 5 long)

|  |  |  |  |
| --- | --- | --- | --- |
| Name inputted | What happened? | Does it work as intended? | Screen shot number |
| I | Didn’t accept it as it is too short | Yes | 48 |
| +PO | Accepted it | Yes | 49 |
| Crazy | Accepted it | Yes | 50 |
| Jeremy | Only saved Jerem as it only takes the first 5 characters | Yes | 51 |

## Testing results:

Spreading of 2 counters: With all of the tests showing that the program worked as intended the spreading of 2 counters works correctly

Spreading of 3 counters: With all of the tests showing that the program worked as intended the spreading of 3 counters works correctly

Spreading of 4 counters: With all of the tests showing that the program worked as intended the spreading of 4 counters works correctly

Closing each page: The tests showed that all of the close buttons worked as intended; stopping the program from running and closing the screen

Changing the resolution: All of the tests showed that the program worked as intended showing that changing the resolution works correctly

Changing the grid size: All of the tests showed that the program worked as intended showing that changing the grid size works correctly

Changing the colour: For all colour combinations tried the colours worked as intended displaying each correctly so this feature works

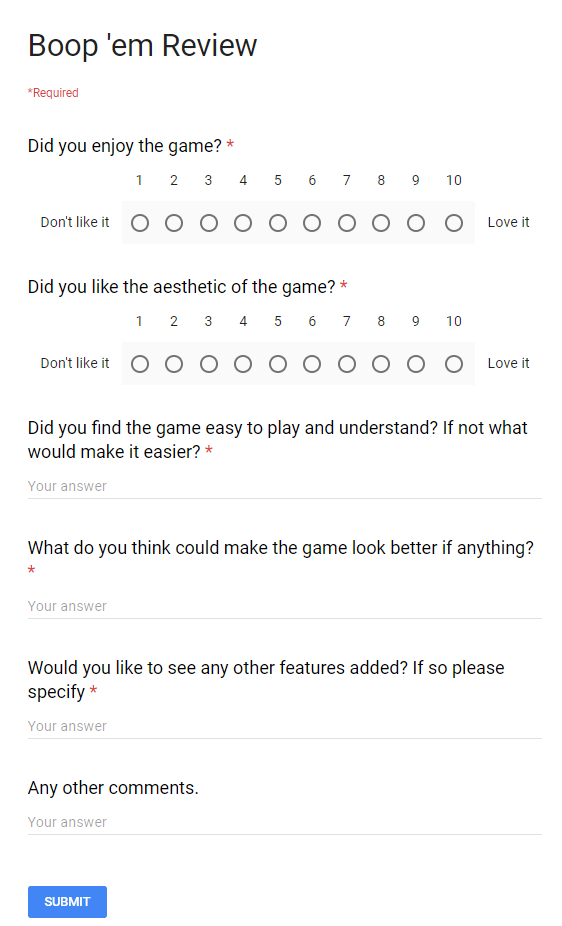
Change the number of teams: For all of the different numbers of teams it worked as intended so changing the number of teams works as intended

Validation on the options menu: For all numbers tested the options menu worked as intended including the extreme and exceptional numbers (although negatives couldn’t be accepted as it only reads numbers) so the validation on the option menu works properly.

Validation on the winners input: For all the tests it worked as intended so the input validation works correctly

## End User Testing:

I got a variety of different users to play the game and then fill out a survey to see what they thought of the game, how easy it was to use and possible improvements. I used google forms to get peoples responses in one place and not have to use paper. The questions I asked were “Did you enjoy the game?”, which was on a scale from 1 to 10; “Did you like the aesthetic of the game?”, which was on a scale from 1 to 10; “Did you find the game easy to play and understand? If not, what would make it easier?”, which they could write a few sentences to answer; “What do you think could make the game look better if anything?”. Which they could write a few sentences for; “Would you like to see any other features added? If so please specify”, which they could write for and finally “any other comments.” Which they could use to write any other feedback including bugs. It looked like the following

I got a total of 12 responses which can be found in the spreadsheet attached

## On-going refinement:

I changed the total amount of responses I had as more came in. I didn’t make many changed to this as it was near the end so I knew the quality of work that I was striving for.

# Evaluation:

## Evaluating if the solution matched the requirement specification:

My program successfully achieves the purpose of the solution by entertaining the user and allowing the user to play the game as intended.

The program successfully achieves all of the scopes by: using the grid size allocated in the options menu, allowing the users to take it in turns to place counters on a 2D grid, splitting the counters when necessary, not letting players who are out of the game place more counters, displaying the winner on the screen, having a sorted score board that shoes the top 10 scores and having the number of players entered in the options menu playing the game.

It also meets some of the boundaries allowing the players to pick which colour they are, the screen resolution and having an instructions menu.

I have managed to meet all the user requirements and functional requirements also.

Therefore, my program meets all of the requirements in the requirement specification as well as some of the boundaries making it a program that meets the specification and works as intended.

## Evaluating the final testing:

The final testing showed that all of the features of the program were working as intended showing that the program made didn’t have bugs that broke functionality and that the game can be played without problems, which is what I set out to achieve.

## Evaluating the end user testing:

The end user testing was helpful for example it showed me that I had made a successful game with me having an average of 8 ¼ out of 10 on how much the end users enjoyed playing the game. Therefore, I feel like I have successfully created an enjoyable game which was one of my outcomes.

1/6 people however found the game difficult to understand and they suggested having better instructions so that is something that could be worked on, also someone suggested making the grid lines thicker so that it was easier to see on smaller screens, and therefore know who’s turn it is, so I made that change and made the grid lines 5x thicker. Overall people found the game easy to understand though which is good as that is one of my outcomes.

Under aesthetics I got two responses asking for the animation of the popping could be slowed down to I did that and in future I would add a way to change that in the options. People also said that it may look better with 3D textures for the counters which I could look into in the future.

For features added people asked for a back button, scores based off of the size of the grid so no matter the grid size people could compete and online multiplayer. Multiple people suggested a back button so in future I may look into that, and the score is easily changeable. Overall, I feel like the only feature needed to be added is a back button which is a over sight and could be corrected.

Other things that were suggested in end user testing was a better name for the game which I could change if I think of a better one but I don’t feel like it is a giant problem.

The end user feedback as told me that people enjoyed the game but the features of a back button could be added as well as some minor changed so I see the game as a success in terms of the end user’s satisfaction.

## Further development:

If I was going to proceed developing the game there are only minor items that I feel should be changed or added:

* A back button in the game so that if a player makes a mistake they can go back to the last turn
* Better textures for the counters making them 3D so that they look nicer
* An option to change the animation speed of the popping
* A skip animation button in case someone doesn’t want to watch all the popping animations
* Online multiplayer so people don’t have to be together to play the game

## Conclusion about solution:

The solution was a success as it met all of the specifications set out at the start of the development process, as shown from the testing, as well as all outcomes for the project as a whole.

## Evaluation on development process:

I would have changed how I did the developing of the program if I was to do it again. I would change it so that I used a AGILE methodology rather than a waterfall as it was hard to do the documentation before doing the program as I was new to python so I didn’t know what was possible. I also would have learnt python and Pygame at the very beginning before I started the documentation so that I understood the limits and how much time it would take to complete the code. Other than I think that the development process went well as everything was few alterations being needed to made to the Gantt chart. I believe it went smoothly as I followed the Gantt chart roughly so knew the timings for everything so I never got thrown completely off track.

## Evaluation on my performance:

I believe that I have done well as the solution has a lot of the features I wanted at the start and is working correctly for all the features implemented as well as completing the documentation for the program. However, I do believe that I could have followed the Gantt chart more closely so that the task didn’t take as long.