SDD Report

# Part A: Progress

## Journal

### 6th March

Reconsidered project to be reversi, instead of checkers, with similar input styles.

Began work on a structure chart

### 7th march

Finished structure chart, with all major functions included

### 13th March

Prepared IDE workspace for writing majority of program, set up files, configured WebStorm to work for typescript, got “Hello World!” code running

### 15th March

Began creating each function in structure chart, with parameters of data input, and returning appropriate data types, though primarily empty variables

Began structuring from main downward, linking functions similarly to structure chart, though reordering some functions, considering multiple outputs of a function can be clunky, as well as the challenge having the player or computer make another back-to-back move if the opponent cannot make one. Began drawing a second draft of structure chart, with these considerations

### 20th march

Finished second draft of structure chart, finished creating outline of each function in main code, linked every function according to structure chart, with fundamental logic, for cases where it does or doesn’t trigger a function.

### 22nd March

Wrote logic for checking endgame function. It can now detect given a board whether the game ends and who wins. Created new function for checking score, updating score, and empty function for displaying score.

### 23rd March

Changed the player order, added capacity for 2 players, instead of just player against computer. Made first player 0, second player 1, and empty tiles 3. Player 1 is also false, and player 2 is true, when expressed as Boolean.

### 24th march

Wrote logic for function that flips tiles in a direction from a placed tile. Uses a while loop to iterate forward to find tiles, and another while loop within to iterate backward and flip tiles

### 25th March

Spent an hour testing the function that flips tiles in a row, which wasn’t working. I used test outputs at various points, and discovered that the issue was that I wrote “not x and not y” instead of “not (x and y)” or “not x or not y”. Wrote code for function that flips tiles around a piece, by iterating the previous function, tested it. Changed some code around in the run turn function because some arguments to if statements were wrong, since I forgot to change it when I initially changed the game to accommodate 2 players.

### 27th March

Wrote logic for checking a position can validly have a piece placed there or not, by comparing a board where a move is made in that square, compared to if the piece was only placed there. Using an iteration of this function, I created the logic for the function that generates all valid moves for a player

### 28th March

Began work on implementing a UI output for the board. I created a HTML file with a script linked to the main file, but without much success. I intend to complete this later.

## Evaluation

### Tasks completed

Overall, I have written code that processes the actual game. This includes all internal logic of placing tiles in the board, given the input positions, updating the board every turn, determining the score and winner of a game.

### Tasks yet to be completed

The code for the camera input has yet to be written, which is the function that takes the image from the camera and determines the position of the input tile. I will need to learn how to access the camera from typescript/javascript, and learn a module that allows me to process the image produced by the camera to identify colours in the image, and positions of the piece, relative to the board. Red dots will be placed in positions in the UI to make it easier to know where things should be.

The displaying function has not been completed, and code that allows UI to function has not been written, as well as code to interact between logic in the typescript and the html/CSS of the site. I will need to learn a relevant module for displaying a simple, but properly formatted board, which can be manipulated by the typescript, as well create cleaner images to use within the UI.

The algorithm that determines the move for the computer to make also has not been written. I need to write the function, which has the parameters of difficulty, which determines how good the turns should be, and the board itself.

### Difficulties and problems encountered

1. Over much thought over my development process while planning, I realised that 24 individual checker pieces would create problems for my image analysis code, with many of the same tile around the board, which would make it hard for the computer to correctly interpret for the board. King tiles, or upside down checker pieces would also be needed to be computed differently, which would either mean textured tiles on one side, which would be more work for the computer to analyse, producing major error, or confusing colour schemes for the player. Additionally, I realised that having the player move both for themselves and the computer would provide extra work for the player, and would not be an ergonomically sound idea in practice. I combatted these difficulties by altering the initial plan, using reversi, or Othello instead, and using only 1 input key. This would be both less frustrating for the player to work with, as well as less prone to error of the camera mis-identifying positions of pieces.
2. Setting up a typescript environment presented a challenge which took several hours to configure. I didn’t know how to set up a typescript environment, I tried VSCode for a while, and couldn’t get modules working because they wouldn’t install, and had the same issue with WebStorm. Once installing typescript, I couldn’t properly run it. I rectified this issue by searching online for tutorials on setting it up, which initially didn’t work, but with other online sources, worked out the error was the firewall in the computer. The second part of the problem I solved by reading documentation for WebStorm, and other online sources that taught typescript, and learnt how to set up a compilation file and configure WebStorm to automatically compile the typescript upon editing into javascript to be run.
3. Writing the function for flipping tiles in a specific direction was problematic and took a while to solve. It required me to flip tiles between a given tile and the next of its own tile in a row, meaning it had to iterate both to find how far it can flip, and iterate to flip the tiles themselves. Initially I tried iterating with a while loop, stopping if it found a blank tile or its own tile, then backwards with a while loop, flipping tiles each iteration, till it reached the initial position. However this didn’t work, because it wouldn’t recognise when it had passed the original position. I reconciled this by making the backwards flipping loop a for loop, which iterated as many times as the first loop iterated forwards, using a counter. This solved the problem.

### Foreseeable difficulties

1. Analysing the image will prove to be a challenge to write code for. Outlier pixels are likely to be picked up and cause issues, if I don’t go out of my way to create exceptions for, and make sure to only register the piece as where the majority of pixels are. I also have not used image analysis programming modules before, and it seems more complex than most modules I have used.
2. The part of the computer side of moves that determines the best move based on difficulty is likely to be a problem in the future. A perfect bot would be actually easier to write, since it always picks the best move, based an arrangement of factors contributing to the value of moves. However, this wouldn’t be a good experience for the player, so lower difficulties should not always make the best move. Writing an algorithm that finds a balance between not blundering a move when it is the most obvious best move to take, i.e., taking a corner, but also doesn’t make perfect moves all the time. This balance also has to allow for variable, so it can be adjusted with a difficulty setting.
3. Propping up the camera in a specific location is likely to be another problem. I would have to create a frame that is stable enough to hold a computer webcam over the screen, but not so large that it becomes obstructive to the user. I would either have to find something suitable to prop it up, or create something from scratch to hold it up. The diagram drawn below is one possible solution to how I could hold it up, though that would take time to engineer, and might not be that strong, with only one arm.

# Modelling Documentation

## Storyboard and screen designs

A screenshot of a computer

Description automatically generated with low confidence

A picture containing graphical user interface

Description automatically generatedChart

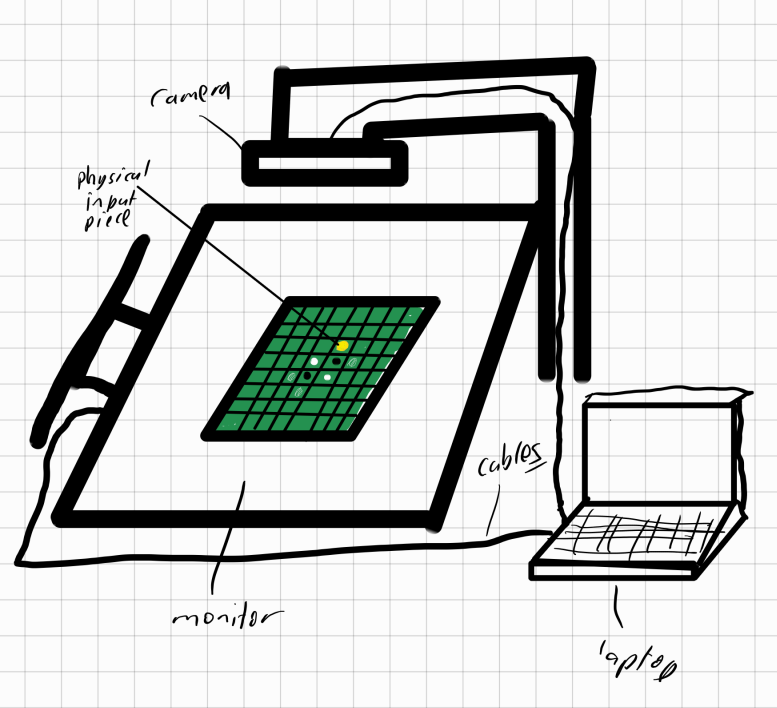
Description automatically generated

Chart

Description automatically generatedA picture containing graphical user interface

Description automatically generated

Diagram

Description automatically generated with medium confidence 

## Structure Chart

Diagram, engineering drawing

Description automatically generated

## Data Dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Type | Variable function | Description |
| gameRunning | boolean | game | the variable that dictates the game running |
| difficulty | number | game | Variable from 0-3 describing the computer difficulty, except 0 means player vs player |
| currentTurn | boolean | game | a boolean that stores which player it is, false for player 1, true for player 2 |
| board | number[][] | game | an array of arrays, it contains an 8x8 grid containing the data for the board. |
| row | number[] | checkEndGame | the row of the board being iterated through |
| tile | number | checkEndGame | the current tile of the board being iterated through |
| player1Score | number | checkEndGame | the current score of the first player |
| player2Score | number | checkEndGame | the current score of the second player |
| output | number | checkEndGame | variable that gets changed to the corresponding output based on the state of the board |
| count | number | checkScore | counts when tiles of the selected player are found, and contain the score of that player at the end to be outputted |
| row | number[] | checkScore | the current tile of the board being iterated through |
| tile | number | checkScore | the current tile of the board being iterated through |
| validMoves | number[][] | runTurn | an array of moves, where each move is defined as an array of length 2, containing coordinates |
| moveToMake | number[] | runTurn | the move to be sent from the player or computer turn making algorithm to be played into the game |
| updatedBoard | number[][] | runTurn | the board after the end of the turn |
| moves | number | getMoves | array to be filled with all valid moves for a position |
| controlBoard | number[][] | checkValid | the board, except the selected move has been placed in its position |
| testBoard | number[][] | checkValid | the board with the selected move computed, which is used to compare to control board |
| xIncrement | number | flipDirection | the amount by which the horizontal should be changed to check in a direction for tiles |
| yIncrement | number | flipDirection | the amount by which the vertical should be changed to check in a direction for tiles |
| xIteration | number | flipDirection | the horizontal coordinate of what is being checked to find how far in a direction tiles may be flipped |
| yIteration | number | flipDirection | the vertical coordinate of what is being checked to find how far in a direction tiles may be flipped |
| count | number | flipDirection | the number of times the algorithm has found flappable tiles, which exists so it can go backward and flip them |
| player1Score | number | updateScore | the score of the first player to be displayed to the board |
| player2Score | number | updateScore | the score of the second player to be displayed to the board |

## Peer comparison

Chosen peer: Marcus

Our projects are vastly different, especially in the structure of the programs. Marcus is running a unity program, with a main program, and multiple objects that interact within the unity engine, and act according to individual scripts.

Most notably, Marcus’ project is event oriented, meaning that it is concurrently running scripts, on different objects, which is strongly contrasted to my strictly sequential program. This is necessary for him, since unity requires each object to have its own properties and ways in relation to time and interaction with other objects, and to update simultaneously. His program even allows him to run the same script, multiple times concurrently. In contrast, in a turn based, web game such as mine, a sequential approach to structure is more suitable. A function may be used multiple times, especially within a loop, but usually with different parameters each time, and never concurrently.

However, his concurrent running of scripts allows more scripts to be run quickly, assuming sufficient processing space, meaning more tasks can get done in a time frame, and are limited by the update rate. However, running too many scripts concurrently may cause issues for the rest of the computer system, using too much memory at once. Running an ordered script runs slightly less quickly, but uses significantly less memory at a time, and should not significantly slow down the computer.

In conclusion, our programs main structural difference is the event oriented, while mine is sequential. Though we both have multiple uses of each function, the way and order they are executed are different, due to my usage of a website compared to his usage of the unity engine.