COSC1076 Advanced Programming Techniques
Assignment 2 Report

Group: YTB

Rico Wu
Merhawi Minassi
Jialun Darren Huang
Siddhartha Raju Chandra Vani

Trello Link: https://trello.com/b/KGaLq12Y/ytb
Github Link: https://github.com/s3668396/YTB (Private)

Background

On the 11th of September 2019, after missing out on the previous 5 APT labs, a teamless Rico stumbled into class not knowing what to expect. Out of luck, Merhawi, Darren and Sid, who have already formed a group was missing a fourth member, and so, the legendary 'YTB' team was born. Throughout the rest of the semester, apart from the labs, YTB engaged in weekly meetings at RMIT every Monday afternoon to go over any work that was done in the previous week. A list of items to do and its deadlines were made at the end of each meeting and all members of the group were held accountable for their individual tasks. The main method of communication between group members were Discord (where our work was discussed) and Facebook Messenger (where we would discuss where to meet up). A Trello board was also used to track these tasks and deadlines, as well as a Github where all members were able to push their own work to.

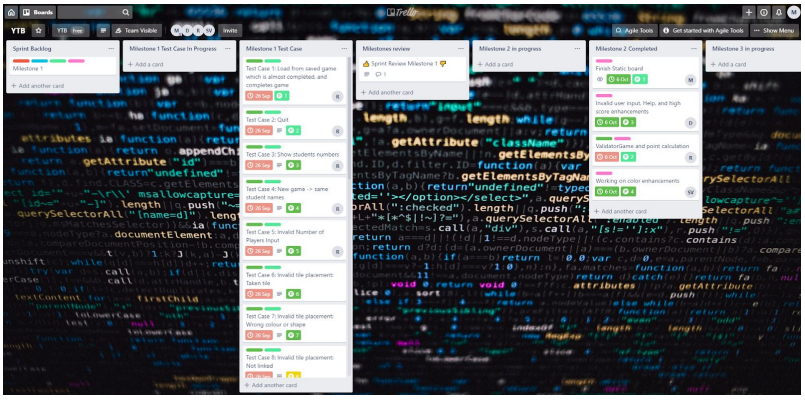

Figure 1: Screenshot of Trello Board

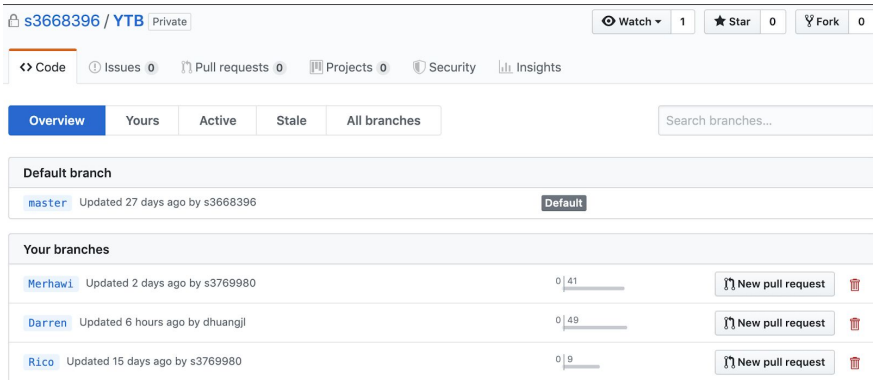

Figure 2: Screenshot of Github Repo

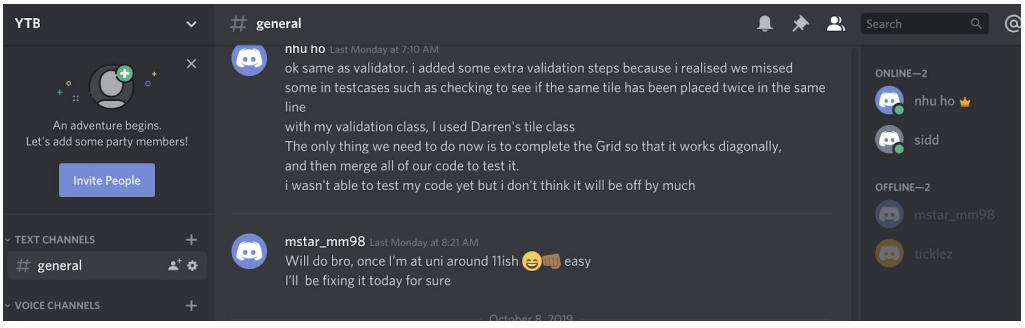

Figure 3: Screenshot of Discord Chat

Game Code/Features

Test Cases:

Before our code was implemented, the group came together and brainstormed a list of scenarios where a test case was required for the game. This was then revisited later on and modified as more features of the game was implemented which changed our original test case. The list of test cases that were created were:

- Load from saved file, complete game [1]
- Quit game [2]
- Show student numbers [3]
- Create a game with the same student names [4]
- Invalid number of players [5]
- Invalid tile placement: taken tile [6]
- Invalid tile placement: wrong colour/shape [7]
- Invalid tile placement: not linked [9]
- Invalid tile placement: tile not in hand [11]
- Tile placed outside gameboard [10]
- Invalid tile placement: same tile in line [12]
- Invalid menu option [13]
- Load game, no saved game [14]
- 3 player game [15]
- 4 player game [16]
- Replace tile [17]
- Qwirkle [18]
- Successful save [19]
- Point Calculation [20]
- TileBag empty & No valid moves [21]
- Game successfully loaded [22]

Design & Implementation:

Classes implemented:
GameBoard.cpp, GameEngine.cpp, HighScore.cpp, HighScoreTable.cpp, LinkedList.cpp, main.cpp, Menu.cpp, Player.cpp, Tile.cpp, TileBag.cpp, TileNode.cpp

TileCode.h

TileCode are constants for the Tile. The colours: Red, Orange, Yellow, Green, Blue, Purple are denoted by its starting character. The tile shapes: CIRCLE, STAR_4, DIAMOND, SQUARE, STAR_6, CLOVER are each given an integer from 1 to 6.
Colour code: NORCOLOUR, REDCOLOUR, ORANGECOLOUR, YELLOWCOLOUR, GREENCOLOUR, BLUECOLOUR, PURPLECOLOUR
Colouring Tile: F_RED(x), F_ORANGE(x), F_YELLOW(x), F_GREEN(x), F_BLUE(x), F_PURPLE(x)
These functions takes in a string of tiles and changes its color.
To create colours in our text, we put the Tile in between the colour code.
<<Colour code>><<Tile>><<NORCOLOUR>>

Tile/TileNode/LinkedList

A Tile contains a colour (char) & a shape (int). These colour and int are defined in TileCode.
TileNode contains a Tile Object and a pointer to the next TileNode.
TileNode is used in our LinkedList class.

Player

Player.cpp stores the data of each player playing the game. PlayerNo, PlayerName and PlayerPoints are the essential data that it keeps track of. It also keeps track of the PlayersTurn and tiles in PlayersHand which helps in continuing previously saved game on which players starts the first move.

Menu
This class displays the menu to the players and performs corresponding tasks as per the users input. There exists to some extent, an input validation method that will print out an error code if a wrong input is typed into the console. This also checks to see if the coordinate entered actually exists.

End of EOF
If a user enters Ctrl+D, the game will exit automatically with a “Goodbye” displayed.

GameEngine
The GameEngine is the brain of the code. It invokes all other classes to perform required action to the game based on users input.

Working with the Menu class, it works out what the user entered and displays the expected output. The GameEngine is responsible with keeping the game board, highscore list, list of players and the tile bag.

GameBoard
Our gameboard is a 26x26 array. Not only is this class responsible for the actual board itself, it is also in charge of tile validation and point calculation.
When a tile is placed onto the board, it will automatically run a check to see if that move is valid. The following checks are done to determine whether or not a move is valid or not:

- Is the coordinate empty?
- Are there any tiles surrounding the coordinate?
- Has the same tile been placed on the same line?

If the move is deemed to be valid, the tile is then placed onto the gameboard and then starts to calculate how many points that move is worth. This is done by checking how many tiles there are in a row in each direction, adding up the result and then adding 1. An additional check is also done so that if the sum of opposing directions are equal to 5, then the game will automatically trigger a QWIRKLE!

HighScore/HighScoreTable
This class captures the top 10 scores played in the game. After a game ends, if a player’s score is higher than one of the top 10 scores, it will be added to the high scores list. A user can see the high scores by View HighScores in the main menu option. A save file (file format ‘.high’) is used to store the high score list.

TileBag
TileBag contains all the tiles, selected randomly and stored in a list. Each time a player plays his turn, the tile from top of the list is given to the player.

Minor Enhancement:
The following minor enhancements were implemented:

- Help!
- Better Invalid Input
- Colour
- High Scores

Major Enhancement:
The following major enhancements were implemented:

- 3-4 Player modes

Help!
A user can type “-help” if they are unsure what commands they can enter.
If a user enters something wrong, they will be prompted to enter “-help” to see what commands they can use at the current game menu.

Better Invalid Input
When a user enters an invalid input, the input will be rejected and an error message will be displayed.
The error message helps users identify their mistake. They will also be prompted to enter “-help”.

Colour
Each tile displayed on the game board are colored based on its defined color. The functions defined in TileCode.h are invoked to change the color of each tile before displaying.
This makes is easier to understand the grid.

3-4 Player modes:
We implemented a way that allows the game can be played with more than 2 players. Technically, it can be played with more than that but we have limited the maximum number of players to 4.
All rules still apply.

Discussion/Conclusion

Overall, while the project was able to be completed in the end, it was met with some delays, and bottlenecks along the way. At the start, it was difficult to grasp how to tackle creating test cases without having some sort of implementation already complete. However, it was decided after a group meeting that a list of test case scenarios would be created based on the basic features of the game outlined in the specification, and would then be revisited later on down the track so that it could be modified. Additional test cases were also created by recommendation from our lab demonstrator surrounding different types of point calculations and move validation.

While Implementing, we faced many obstacles. File reading and writing and the algorithm to check if the tile can be placed on the board has been our main challenges. However, with the power of love and friendship, we managed to overcome all odds and figure out the way!