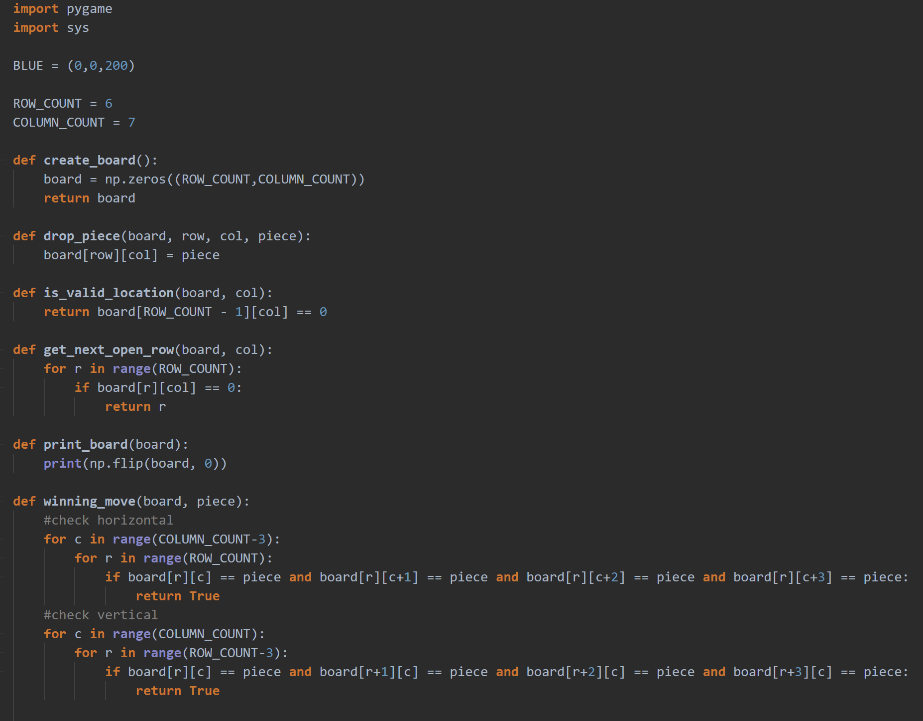
Mr. Lai

Software Design and Development

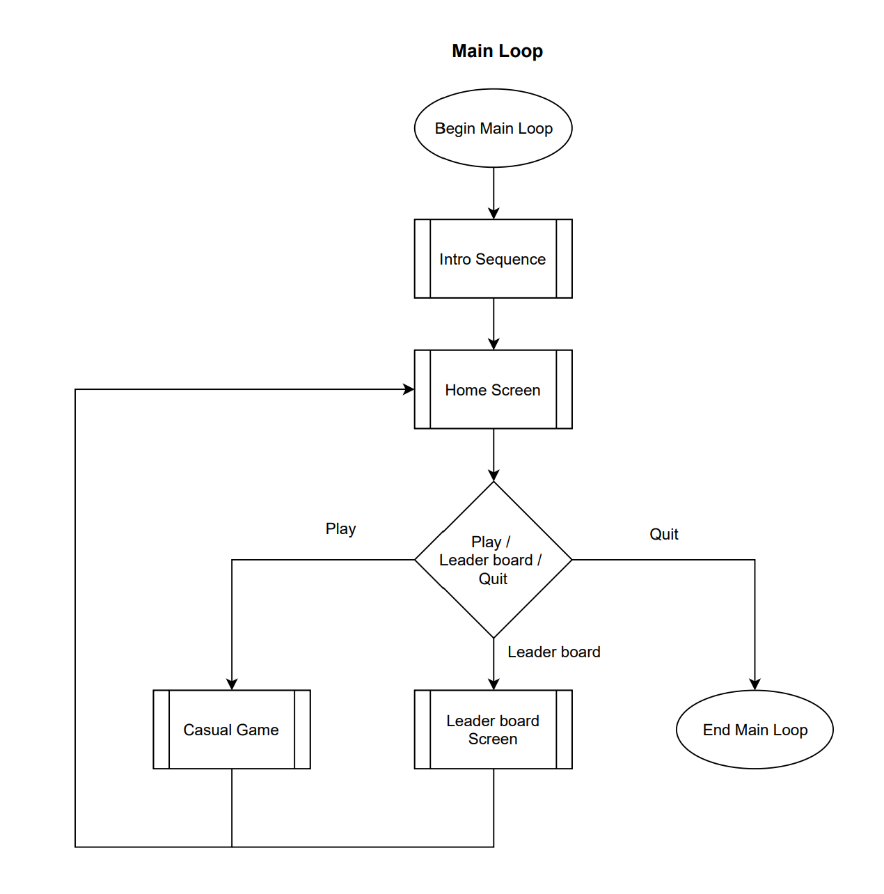
Student Number: 3168 3025

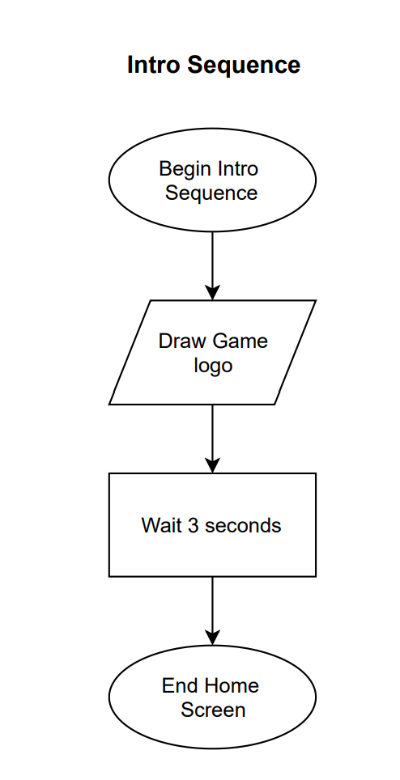
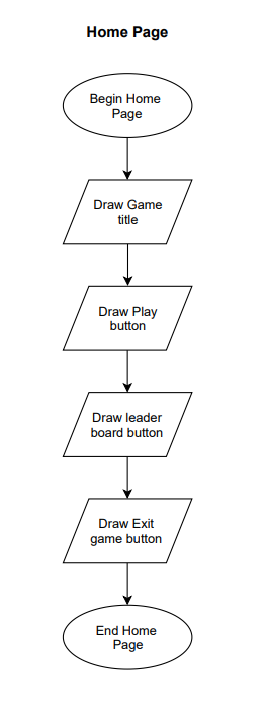


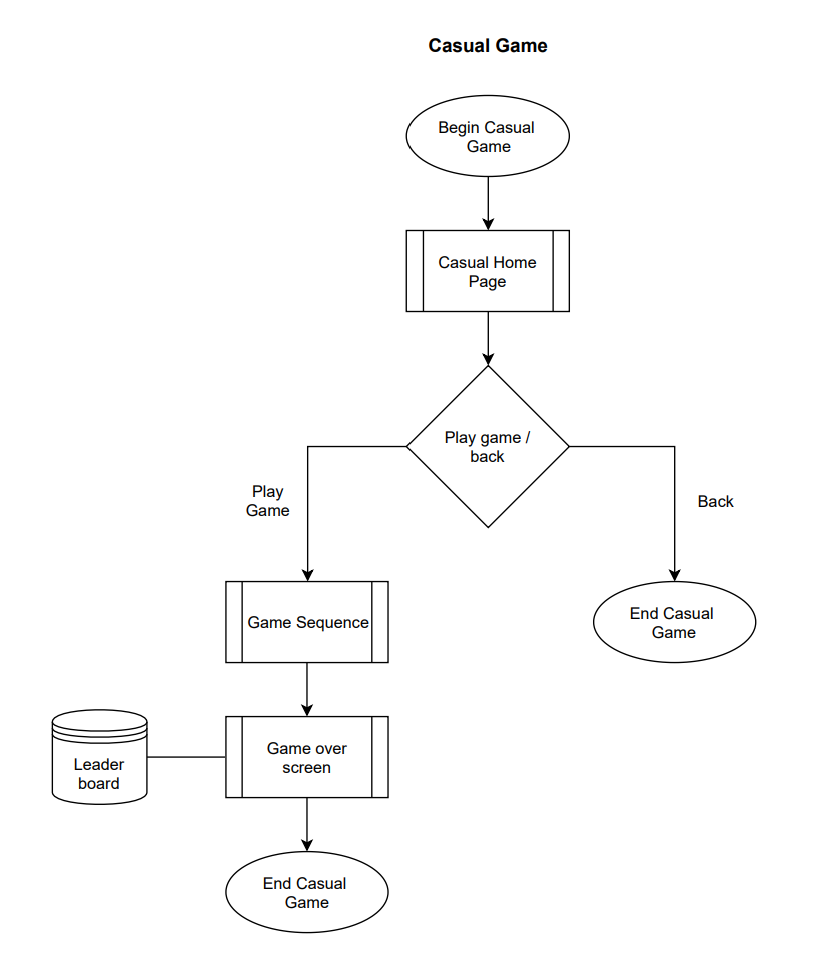
GUI development PLANING AND Project Logbook

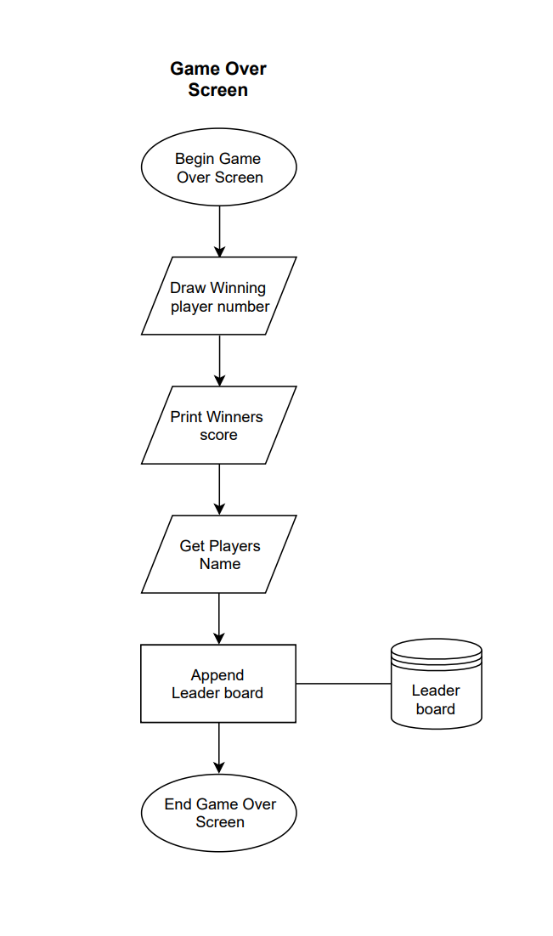
Matt Starr

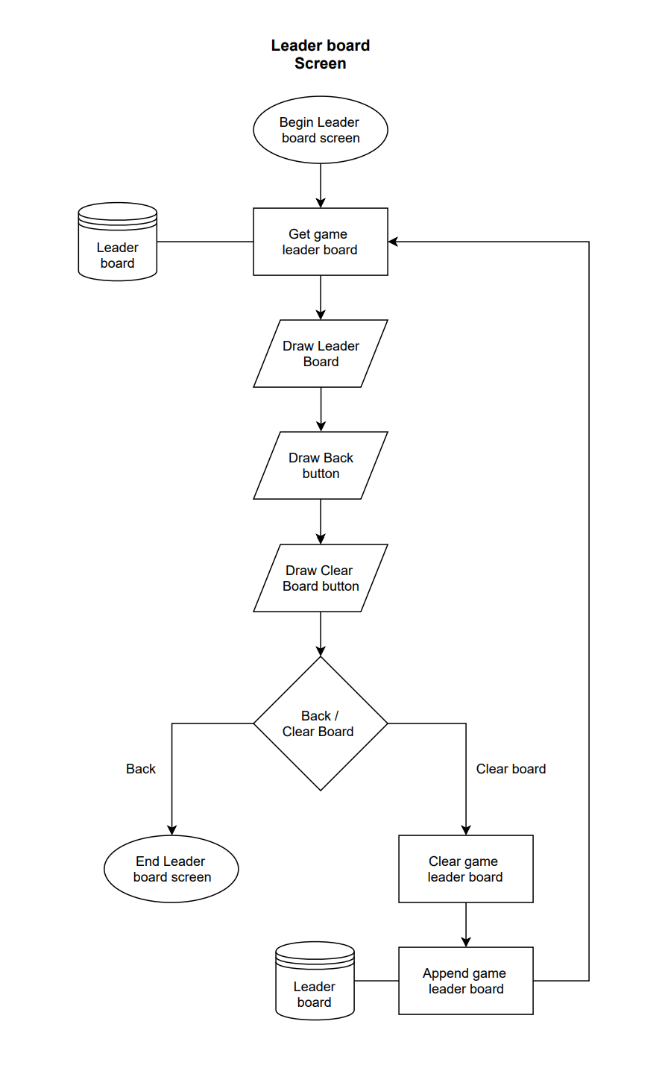
**Planning**

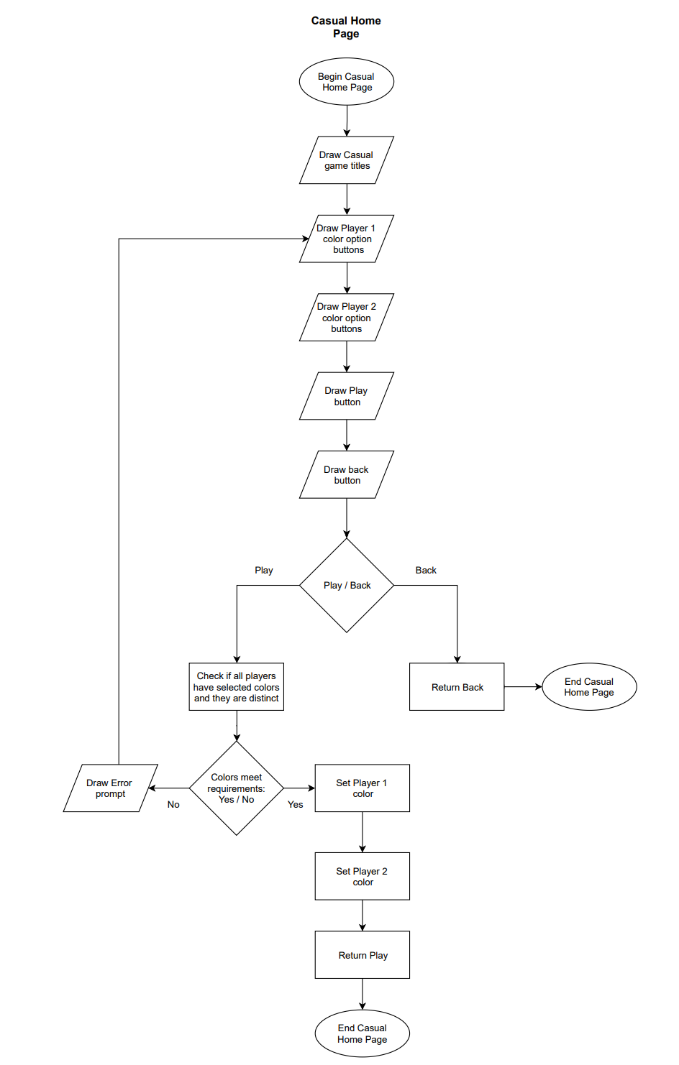


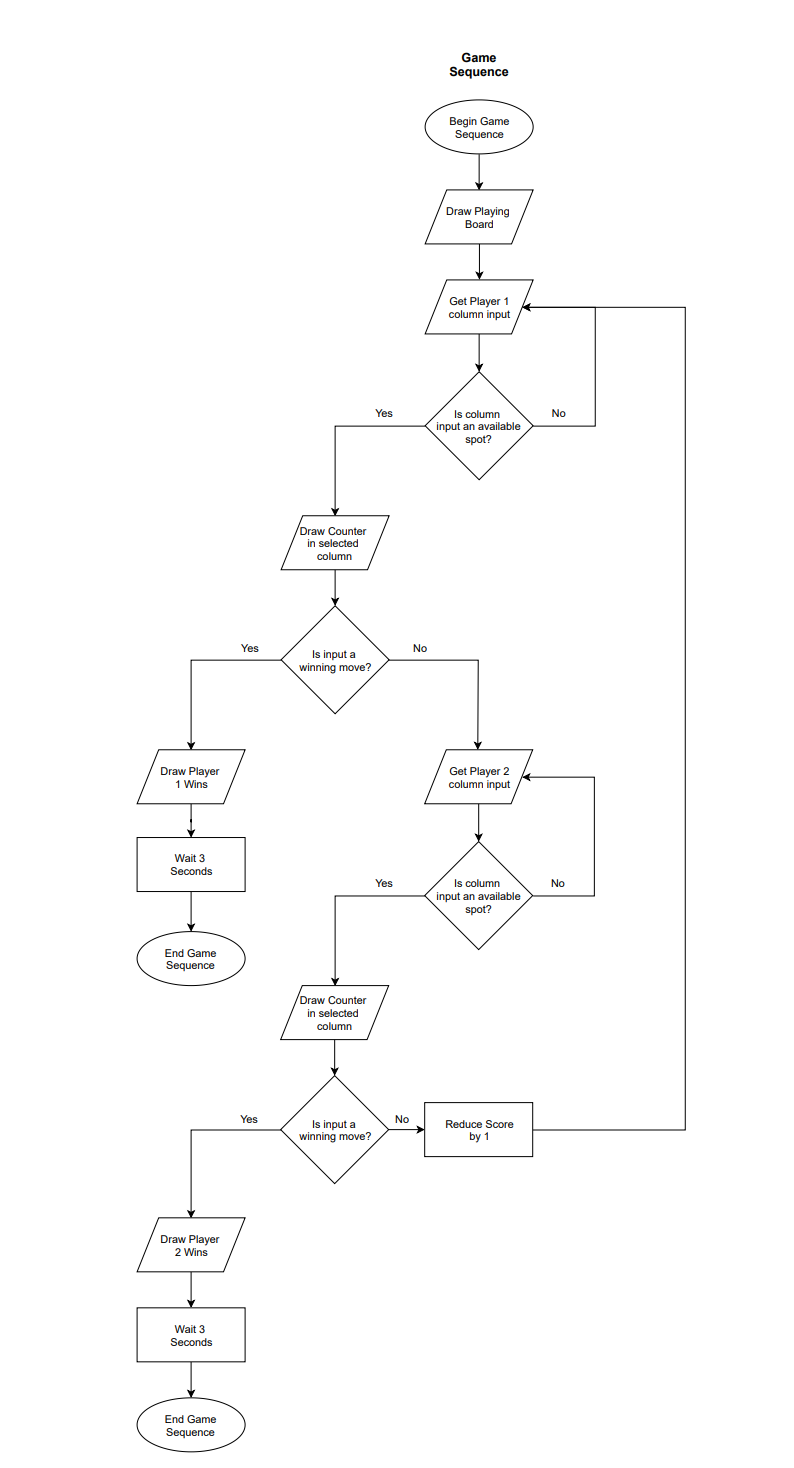


****



****

****



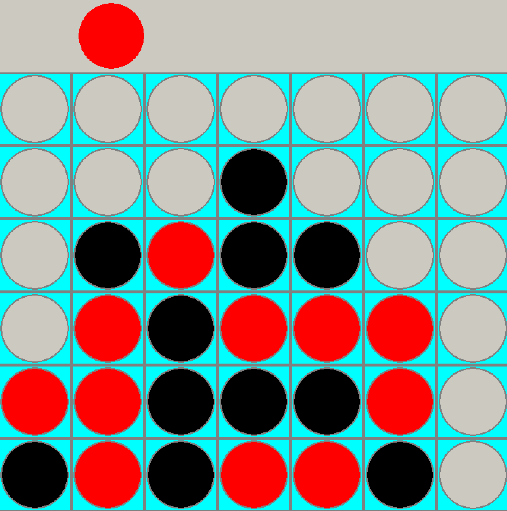
**Project Logbook**

Machine generated alternative text:
Connect 4 GUI.py 
Connect 4 Terminal interface.py X 
Assignment 2 Matt-Starr-SDD-Assignment-2 •P Connect 4 Terminal in 
1 
2 
3 
4 
5 
6 
7 
8 
9 
10 
11 
12 
13 
14 
15 
16 
17 
18 
19 
20 
21 
22 
23 
24 
25 
26 
import numpy as np 
ROW COUNT = 6 
COLUMN COUNT 
7 
def 
def 
def 
def 
def 
def 
create _ board( ) : 
np . zeros ( ( ROW_COUNT , COLUMN_COUNT) ) 
board 
return board 
drop_piece(board, row, col, piece) : 
board[row] [col] - 
piece 
is_valid_location( board , 
return board [ROW_COUNT - 
get_next_open row( board, 
col): 
1] [col] 
col): 
for r in range(ROW COUNT) : 
if board[r][col] - 
e: 
return r 
print _ board (board) : 
print(np.flip(board, 0)) 
winning_move(board, piece) : 
#check horizontal 
for c in range(COLUMN COUNT-3)•. Logbook Entry 1 - Tuesday, 23 July

Today I developed a plan for the features of my connect 4 game and began construction on a test program that uses terminal as a user interface. I have decided to construct a terminal version of the connect 4 game as I am more familiar with the coding environment used to construct this version of the game. I can utilise this familiarity to construct the algorithms required to make the game, which can later be repurposed for the new environment.

I have also decided to create the GUI version of the game using pygame, which is a python extension that allows the control of a graphical interface. Once I have completed the terminal-based version of the game I will adapt it into something that works within the pygame extension.

Terminal controlled version of Connect 4 game

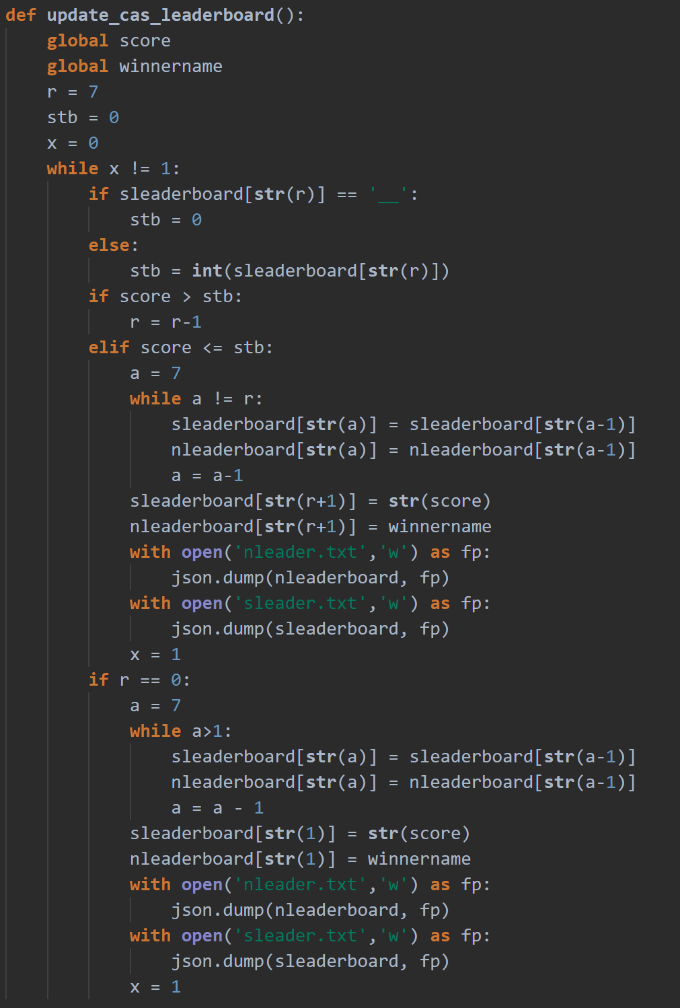
 Logbook Entry 2 - Wednesday, 31 July

Today I constructed a basic version of a GUI based connect 4 game that consisted only of the connect 4 game, with no navigation or leader board features. I encountered an error in which the counters would not add up properly, and it would register that player one had won even if they had only two counters in a row. I was able to fix this problem when I identified that there was a problem in the counting of player two’s counters, in which it would count them as player one’s counters, and also an issue in which the algorithm only checked rows that were one space in, which I fixed by changing the domain of a loop made to check for winning moves.



Logbook Entry 3 - Thursday, 8 August

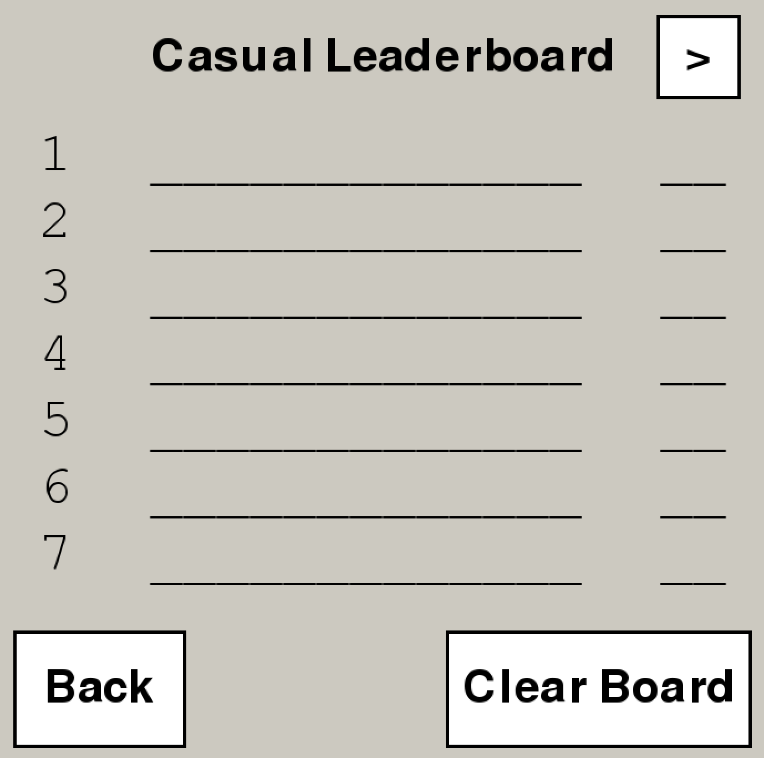
Today I began to work on making a functioning home screen for my game, consisting of a title and working navigation buttons, to play the game and check the leader board. I was able to develop a button class, which can be adapted through changing size, position and text. This button can be used for any functionality as it will work as an event driven object. This was achieved by using for loops which check for events in the pygame window. This can check for mouse inputs and position so that certain functions are only activated based on user driven events.

Logbook Entry 4 - Friday, 16 August

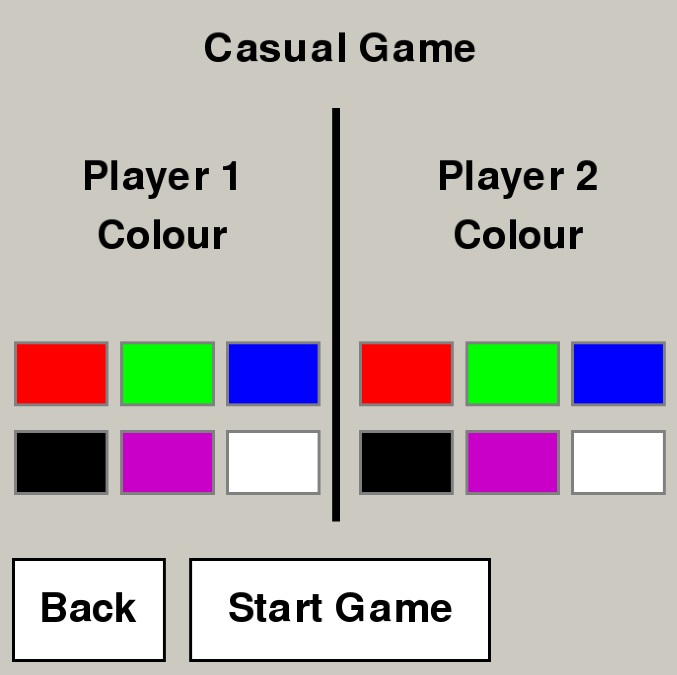
Today I began development on a leader board system, in which at the end of each match, the winning player is assigned a score, and asked to input their name, which can be appended to a leader board list in descending order of highest scores. I have managed to create a leader board list that can be changed and updated when a user inputs their name, sorting that name into the leader board in the right order. I still have to make the leader board able to save to the device so that the leader board can be viewed after closing and opening the application again.

Most of the issues I encountered when implementing the leader board feature were in the implementation of a sorting system for the scores. For each new score added I developed an algorithm that checks the current users score against the scores that already exist on the leader board, until it reaches a score higher than that of the current user. Once the algorithm finds this higher score, it adds the users name and score one place below the higher score and moves all other scores on the leader board down one position.

Logbook Entry 5 - Saturday, 24 August

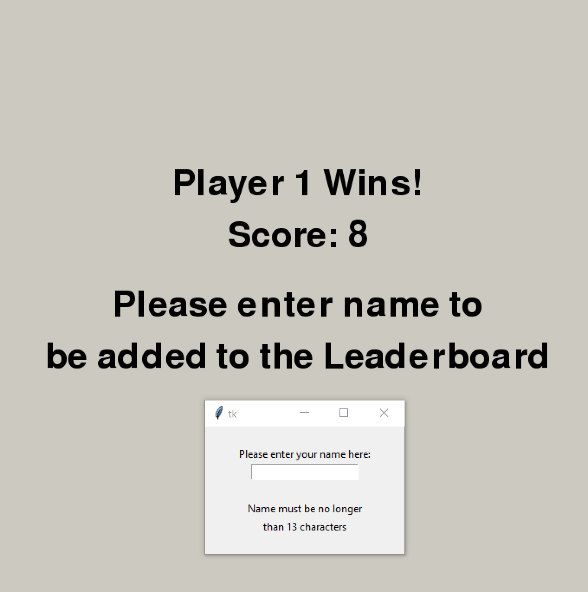
Today I finished developing the leader board functionality for the application by adding a saving functionality to the leader board, allowing users to see their scores still on the leader board even after opening and closing the application. I added this functionality using a json module in python, which stores the leader board data as a text file, which can be viewed later by the module, and incorporated back into the leader board functionality.

A problem I encountered as I drew the leader board was that when drawing the leader board, I was only able to display 7 users without it being too small. To resolve this problem, I added a feature in which users can clear the leader board using a button in the leader board screen. This function empties the json text file, while keeping the same functionality of being able to store the names of the future high scorers.

Logbook Entry 6 - Sunday, 1 September

Today I added the ability for players to choose a unique color for the game before the game begins. I achieved this through adapting my button class to work in a way where a user can identify the button, they have selected by displaying a dark red outline around the color that they have chosen.

Another problem I encountered in the developing of this color choosing system was in the realising that users could potentially try to advance either without selecting a color, or by both users selecting the same color. I adapted my application to work around this problem by not allowing users to advance if they have not met the necessary requirements in selecting their colours.

Logbook Entry 7 - Wednesday, 4 September

Today I added a pop-up text entry box as well as a winner congratulations screen to be displayed at the end of each game. A problem that I encountered when designing the text entry for the winner’s name was that pygame did not have any built-in ways of making a text entry field. Therefore, instead I added a tkinter module into my application, so that when the winners name is required, a tkinter window will pop up with a text entry field and instructions for text entry.

A problem I encountered with the text entry was that words over 13 characters were too long to be displayed in the leader board screen and would not fit. To solve this problem, I added an instruction on the text entry field that asked users to make sure that the name they entered was less than 13 characters. I also constructed an algorithm that checks the length of each inputted name before the submission, and re-prompts the user if the name is too long.