ADS103 Post-Mortem

By: Angelo Estoque

The project I created is a simple game of tic-tac-toe with a simple UI of a menu and a sub-menu for the different game modes available for the player. There are 3 different types of game modes, the standard 2 player game modes which is a normal player versus player game of tic tac toe. The single player tic tac toe in which the player will play against an AI which plays extremely well and makes calculated decisions. The third and final one is just a spectating mode where the player watches as two calculative AIs will play against each other.

During the creation of the project creating the menu went smoothly, due to past experience in a separate class ISE102 which I created a menu for two assessments. It was fairly simple to add the necessary loops and statements to create a simple, yet effective UI for the user to navigate through the project during their usage. Creating the field proved to be the easiest task in the project as its just a fixed data structure using an array and populating it just needed a utility function which made it relatively simple to reset the board again after/or during gameplay. Toggling the player proved to be quite a challenge, however it was easily solved by creating a global variable that sets each player with a predetermined symbol(player1 always being ‘X’ and player2 being ‘O’). Doing this greatly simplified the toggling process of whose turn it is, who won and which name is being displayed.

Creating inputs for both the player and the AI created a myriad of bugs. The input for the player at first did not function properly and kept destroying the data in the array, it was caused by a simple syntax error which was easily resolved. Detecting errors for the player input also presented a similar syntax error.

The most problems arose from the development of the AI, in the initial stages of its development, the AI would blatantly cheat and input more than once by filling the empty spots with its symbol. I quickly realised that the AI itself has enough functions and needs a lot of lines that it can be turned into a class, however it created more bugs, it would reset the AI’s input which resulted in the AI inputting in the same spot and managing the player toggling rose in difficulty again. So the idea of creating an AI class was scrapped and just brute forced the lines into one main.cpp file. After the input was finally resolved, solving where the AI should input presented its own set of difficulties. The AI would make unnecessary inputs, initially instead of stopping the player from winning, the AI would simply just create pseudo random inputs, after resolving that problem by brute forcing the lines which is not best practice and definitely not efficient.

After successfully creating the inputs for the AI which will do its best to counter the player, developing the winning inputs for the AI proved to be relatively simple. Two functions were created, a utility function which is of type bool which signals to the AI that it has a chance to win and prioritises winning over trying to counter the player, unless the player is about to win and the winning input function which is of type void as it doesn’t need to return anything. The two functions have identical code blocks, the only difference is the actions it does after certain conditional statements prove true. However, these utility functions are co-dependent of each other and will not work without the other.

After finishing the AI, I felt that there is definitely a more efficient way of creating the AI which involves mathematical processing of some sort. Similar to pathfinding, it will assign an integer value to each spot and depending on the current value of the other player, the algorithm will find the next spot with the highest value and replace it. However, the AI still fulfils the task albeit it took way too many lines of code to achieve.

After finishing the project I felt a great sense of accomplishment which I feel is unique to coding. Being able to solve the many problems presented, sometimes not with the best solutions, but getting it done and then later optimizing and trying to create better and faster solutions to the previously daunting problems I could barely solve is a great feeling. The AI for this project was definitely daunting at first, but as I kept solving problems upon problems, I gained better insight of how it works and how it should work and ideas kept popping in my head of ways to solve the problem. I did not end up trying to create an algorithm which does any mathematical processes due to the difficulty it poses when at first creating a basic AI proved difficult. However, I am confident I developed a decent AI that will never lose, always forces a draw or if the player makes a mistake, will punish accordingly and possibly win.