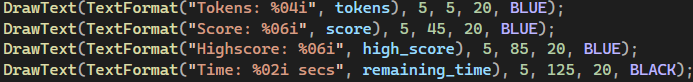
# Introduction

For our assignment we were given a C++ pathfinding project, we were asked to turn this into a game with interaction, time limits, scores and high scores. To do this we would need to edit and update the source code given to us.

# Steps Taken

Step 1 was to add token, score, high score and the timer to the display. We managed to do this by using the predefined variables tokens, score, highscore and time. We then used the DrawText() function to display these on the screen.



Step 2 asked us to Highlight the start and end node, this was managed by using the predefined variable start and end, Then using the DrawCircleV() function



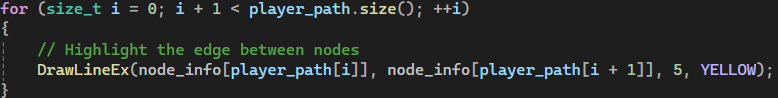
Step 3 was to add each node to the player path by clicking it with the mouse button. This was done by adding the node closest to the mouse to the player\_path with the push\_back() method



Step 4 asked us to add a sound effect each time a node is added to the player\_path. We done this with the raylib built in preview sounds.



Step 5 was to highlight the player\_path and display the players current path on screen. We managed this by creating a loop that adds highlighted edges based on the player\_path



Step 6 asked us to ensure the first node selected was a neighbour of the start node. We managed to do this by implementing a function to check the nodes were connected by an edge. This also completed Step 7 which was to ensure only connected nodes can be added to player\_path.

A screen shot of a computer

AI-generated content may be incorrect.

This didn’t completely solve the problem, so we initialized the player path with the start node. This way when the is\_connected() function runs the first time it will always be checking for a connection to the start node.



Step 8 asked us to add a cost to the nodes in player\_path and remove them from the tokens. We managed this by using the path\_cost() function.



Step 9 was to allow the user to remove nodes from the player path by clicking on the previously selected node.

A black screen with white text

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

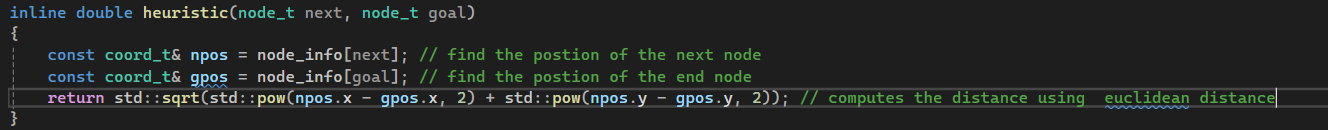
Step 10 was to make it so that upon reaching the end node, The score would be updated, tokens would be awarded back to the player and the level restarted.

A black screen with white text

AI-generated content may be incorrect.

A computer screen with white and blue text

AI-generated content may be incorrect.

Step 11 asked us to update he heuristic in graph.hpp to use Euclidean distance instead of Manhattan distance. 

Step 12 asked us to make the timer count from 60 to 0. We used the getTime() function and record the elapsed time, we then removed this from the start time to get remaining time.



Step 13 was to create a game over if the timer reaches 0 or tokens reach 0. We are also asked to update the high score, clear the score and reset the game.

A black screen with white text

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

Step 14 asks us to update the game so that instead of having A & G as the start and end nodes to randomly assign them.

UPDATE CODE AND ADD SCREENSHOT

# Conclusion

In conclusion our group manged to complete all the tasks that were assigned, we all worked well together to make sure that they got done on time and in an efficient manner. While completing the task we gained a heightened understanding of C++ and the raylib library and will use these skill further in our degrees and careers.