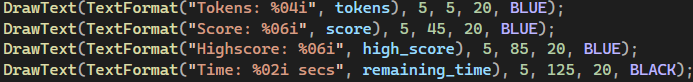
# Introduction

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



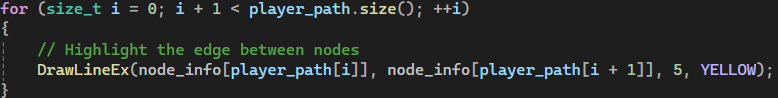
**Step 3** was to add each node to the player path by clicking it with the mouse button. This was done by adding the node closest to the mouse to the player\_path with the push\_back() method



**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. We did this by just taking away the cost of each path the player clicked on from the remaining tokens of the player.



**Step 9** was to allow the user to remove nodes from the player path by clicking on the previously selected node. We did this by seeing if the node that the player was clicking was the last node they clicked on, if so it would call the pop\_last\_node function. In this function it finds the latest node removes it from the list and assigns previous node as the current.

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.

**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. We did this by checking if the player had reached the end node by comparing current node to the end node. If they are the same, then the program calculates the score you earned so far by using the calc\_score function. In this function it calculates the cost of the ideal path and then takes that cost away from the players cost to produce your score. The program then calls the rest function which wipes the drawing and sets new start and end nodes.

A computer screen shot of a math equation

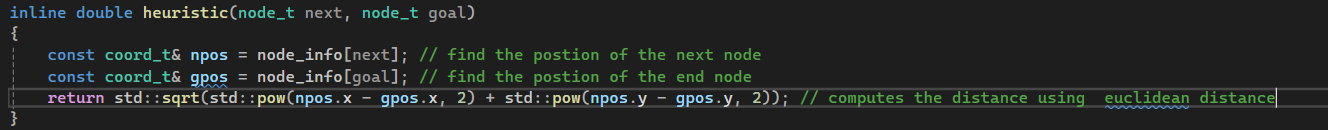
AI-generated content may be incorrect.

A computer screen with white and blue text

AI-generated content may be incorrect.

A computer screen with green 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. We did this by just switching the formal of the equation that computed the distance Manhattan to Euclidean which is d = √[ (x22 – x11)2 + (y22 – y11)2] 

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. We did this by first comparing the current node and the end node if they are the same and tokens are under 0 then. It calls the calc\_score function and the initialise function. Inside the initialise function it sets the FPS, clears everything,generates the graph and then rests everything. The program also checks if remaining time has not gone below 0 if it has it calls the same two functions.

A black screen with white text

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

A screen shot of a computer program

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. We did this by setting both start and end nodes by using the get\_random\_nodes function we made, we also have a while loop to make sure that it doesn’t pick the same start and end as if they equal the same it will pick a new end node. The get\_random\_nodes function works by by setting a min value to 0 and the max to amount of available nodes itthen picks a random number in the index and returns it.

A screen shot of a computer program

AI-generated content may be incorrect.

# 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.