**Spike:** 11

**Title:** Game Graphs From Data

**Author:** Ben Holmes, 103024841

**Goals / deliverables:**

* A design for the game graph
* Main function takes the txt file name via command line argument
* The graph created from the text file in a data structure
* GO and QUIT commands working

**Technologies, Tools, and Resources used:**

* Visual studio
* <https://github.com/nlohmann/json>
* Word
* <https://json.nlohmann.me/features/arbitrary_types/>

**Tasks undertaken:**

* Created design
* Created file structure
* Created the struct for location, with Json
* Created the Adventure world with graph and current locations
* Created the game loop
* Created the main function
* Created the command line accept

**What we found out:**

First was the creation of the design for the game graph, the first of the deliverables

A black screen with white text

Description automatically generated

This was simple albeit I had to make edits as I forgot to edit the descriptions a couple times.

A screenshot of a computer screen

Description automatically generated

Next I created the basic file structure, meaning adding in the blank files I thought were necessary and created the initial structs and classes I would use.

A computer screen shot of a code

Description automatically generated

A computer screen shot of a program code

Description automatically generated

I also created the json file here

A computer screen shot of text

Description automatically generated(one of the locations in the locations array)

A computer screen with white text

Description automatically generated

Next was the Json parsing from the file which took a long time to figure out because of my lack of understanding of how to iterate through the files, and a misunderstanding of how the Json file should be laid out to allow easy creation of the multiple locations in a Vector.

I ended up putting all the locations in a single array so that I could parse them all with a single command and not have to worry about loops.

A screen shot of a computer code

Description automatically generatedA screen shot of a computer code

Description automatically generated

This is the third deliverable

A screen shot of a computer

Description automatically generated

Finally was the game loop, after testing all of that worked, I created a render, update and input functions in the Adventure class (so that I would not have to pass the adventure into each of the functions)

A screen shot of a computer code

Description automatically generated

A computer screen shot of code

Description automatically generated

A computer screen with colorful text

Description automatically generated

These would be updated to match the spec when future implementation has been done, the extensive output of the render and input functions are for testing purposes.

A screen shot of a computer

Description automatically generated

This is the fourth deliverable

A computer screen shot of a program

Description automatically generated

Command line file name input, the second deliverable, was done last as it was simple and I didn’t think about it to the end.

Currently there is not specific player, merely a current location as it was not required for there to be a player in this spikes specs, if I needed to add one, it would be stored in the Adventure class.

Just after I submitted the first time, I realised how to make the graph use map instead of vector. Current was switched to string instead of a location as well (the string is the key mapping to the current location in the graph)

From:

A screenshot of a computer

Description automatically generated

To:

A screen shot of a computer code

Description automatically generated

This will significantly help me out in regards to future spikes both in figuring out how to do similar, and accessing the graph in the future.

A screenshot of a computer

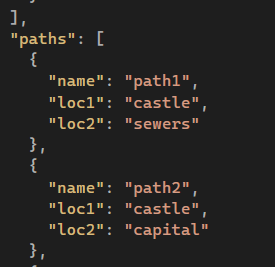
Description automatically generated

# Big change:

During one of the classes, I realised I had not quite complete the graph as the paths/edges weren’t sperate objects that could be created (this may not have been required but I asked some people and they said they had done it this way so thought I should update this in case)

These changes were not transferred onwards to other zorkish spikes but they still work (no commands are changed just how it works backend)

Added a paths array to the json



Added a path class and json deserialiser

A screen shot of a computer code

Description automatically generated

A computer screen shot of code

Description automatically generated

Added paths map to Adventure class



Added paths creation to adventure Start function:

A screen shot of a computer code

Description automatically generated

Updated the gameInput section that changed with paths being added

A screen shot of a computer code

Description automatically generated

Example use:

A computer screen shot of a black background

Description automatically generated