

**Spike:** Task 12**Title:** Game Graphs from Data**Author:** Sam Huffer, 101633177

## Goals / deliverables:

- The specification for a text-file format for representing (at this stage) the details of a world, its locations, and the connections between them.
- Code that can:
  - Load the world from a text file from the Select Adventure stage
  - Store the world data in your program.
  - Allow players to move between locations in the world, with more directions than just “North”, “East”, etc.

## Technologies, Tools, and Resources used:

- Visual Studio 2019
- Microsoft Word

## Tasks undertaken

- I copied the spike report template into the task folder, stripping out the original content and replacing it with goals and resources pertaining to the task at hand.
- I created a blank Visual Studio project “Zorkish Adventure” and copied in all the source files used in Task 10: Game Data Structures, so that for future tasks I wouldn’t have any folders or files named for previous tasks to worry about renaming.
- I wrote out the specification for recording the details of a world, its locations and its items in a text file that could be loaded into the game, as well as how to format a list of available worlds and their text files.
- I designed a basic world with a room for each cardinal direction, one in the middle of all of them, a cellar under the north room, an attic above the east room, and a void that one enters from the west room and exits from into the south room. I then typed up the details of the world in a text file, adding the items used in the previous void test world to the specification. Next, I added a text file for listing all available worlds, listing the test world in it.
- I reconfigured `SelectAdventure.Setup()` to read in the text file “Worlds.txt” to ascertain the worlds available to the player and print them out instead of a hard-coded list of worlds.
- I configured `World’s` constructor to read in the lines in a file (ignoring blanks and comments) and split the text by colons. I added a set of if statements that check that lines that are read in have valid prefixes and the right number of details (that are filled in) for each prefix, and that those details refer to objects that have already been instantiated earlier in the file, producing error messages pinpointing where an error has been made by filename, line, and what the error is.
- I added code for creating objects according to properly formatted lines read from the text file. For this, I added a `Path` class that holds a description and a pointer to its destination, all to be stored in `Location.paths`. However, Visual Studio displayed a weird error that I couldn’t convert a type’s `Ivalue` to that parameter type, so I had to change `Path.destination` to be the id of the location the path leads to rather than a pointer to the path itself.
- Once I finished putting together the code to create objects of the specified type, I ran the game with the test world that I had created, and identified errors in formatting in the text file, as well as bugs in the `World` class’s constructor. I fixed those, and reran the code, and it worked as planned. When I quit the world to return to the main menu, and went back into the world, however, the options of available worlds had duplicated, so I modified the code to replace the list of world names and world file names every time `SelectAdventure.Setup()` is called, which fixed the issue. Next, I tweaked some of the output formatting.

- I added the ability for players to see paths when they look at their current location, and moved all code for the results of looking at one's location or inventory into their own private methods to avoid having to recode that multiple times.
- I added the ability for players to move around according to the directions that a location's paths lead in. Once I had that working, I added to the text file specification a formatting option for aliases of directions, I added to "Test World.txt" direction aliases in line with that formatting, I added to World's constructor checks to process and list aliases, I added to World the methods HasDirectionWithAlias() and GetDirectionWithAlias(), I added to Gameplay.Move() the capacity to swap out a direction alias for the direction it represents, and I added to Gameplay.Update() to handle moving when only specifying the direction or a direction alias.

## Text-File Format

### List of Worlds (Worlds.txt)

The list of worlds formatting is very simple. Each line is a world, listing its name and filename, separating the two with a ":". For example:

"World Name:world\_filename.txt"

would have "World Name" displayed in the SelectAdventure stage, while "world\_filename.txt" would be passed to the constructor of the World class to draw its specification from there.

### Worlds (e.g. "Test World.txt")

The formatting for a world's details are a bit trickier. Again, different details are separated by ":"s, but here different pieces of information are given prefixes to identify what they entail:

- #Comment ==> Ignore
- *(blank line; ignore)*
- W:World Name
- L:location\_id:location\_name:location description
- C:container\_item\_id:container\_item\_name:container item description:container\_id  
*(Note: container\_id is what it resides within, whether a location or another container item.)*
- I:item\_id:item\_name:item description:container\_id
- P:location\_from\_id:direction:location\_id\_to:path description  
*(Note: this lists the connections or pathways that a location (location\_from\_id) has to other locations (location\_to\_id), which are stored in the <string, string> map Location.neighbours, hence the prefix "N".)*
- S:starting\_location\_id
- A:direction name:direction alias

## What we found out

- Visual Studio didn't like "map[index] = MethodName(vector[int], new Class(map[vector[int]], vector[int]))", specifically the object pointer "map[vector[int]]" being passed to Class()'s constructor. Assigning the pointer to a variable first didn't help. Just ended up passing the id string instead, and then looking up the object pointer later.
- When re-loading a stage, you need to wipe values that you read into a vector when you first loaded it, so that you don't add duplicate values when you reload it.
- C++ doesn't support nested functions.