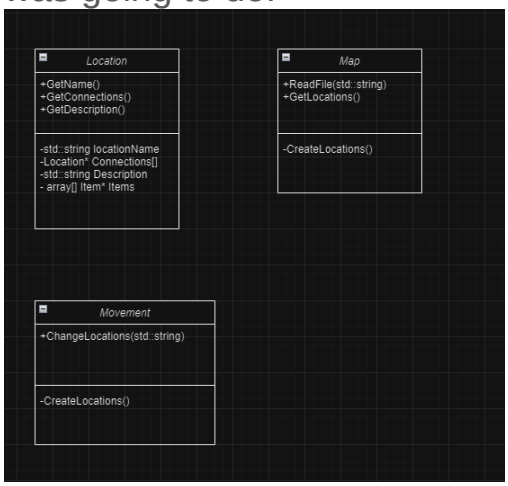
Spike: Task 11
Title: Game Graphs from Data

Author: Liam Whitehouse, 103862481

Goals / deliverables:
The goal of this spike is to create a Graph Data Map. This Map will have Locations that have their own connections to other location on the map. This will all be loaded in via a text file and displayed and allow the player to traverse through the world.

Technologies, Tools, and Resources used:
To create this task, I used Visual Studios and Draw.IO. I designed a rough layout of how the classes would be interconnected and then started creating it.

Tasks undertaken:
Starting with Draw.IO, I created a very simple diagram to help visualize what I was going to do.



It is not much but it helped get me started on the functions and variables required for making this work.

Next, I started to implement this system. I first started by creating a Locations Class which holds all the information about any given location such as its name, description, connections and eventually Items!

```cpp
#include "Location.h"
#include "Item.h"
#include <sstream>
#include <iostream>


Location::Location()
{
    LocationName = "Default Location yet to be made";
    LocationDescription = "A Location yet to be created";
}

Location::Location(std::string a_locationName, std::string a_locationDescription, std::string a_locationConnections)
{
    LocationName = a_locationName;
    LocationDescription = a_locationDescription;
    CreateConnections(a_locationConnections);
}

std::string Location::GetLocationName() const
{
    return LocationName;
}

std::string Location::GetLocationDescription() const
{
    return LocationDescription;
}

std::vector<std::string> Location::GetLocationConnections() const
{
    return LocationConnections;
}

void Location::CreateConnections(std::string a_locationConnections)
{
    std::istringstream ConnectionStream(a_locationConnections);
    std::string ConnectionName;

    while (std::getline(ConnectionStream, ConnectionName, ' '))
    {
        LocationConnections.push_back(ConnectionName);
    }
}
```

Once the locations were in and working, I next worked on the Map. This was a sizable undertaking as the Map needs to load all locations that will exist, as well as their data from a text file. I struggled a little on this but in the end, I was able to have the map Generate each location from the Text File with their description and Connections.

```cpp
#include "Map.h"
#include <fstream>
#include <vector>

#include "Location.h"
#include "Item.h"

Map::Map(std::string a_file)
{
    std::string fileLine;
    std::ifstream mapFile(a_file);

    if (mapFile.is_open())
    {
        while (std::getline(mapFile, fileLine, '\n'))
        {
            if (fileLine.empty())
            {
                continue;
            }
            locationInformation.push_back(fileLine);
        }
    }

    CreateLocations();
}

std::vector<Location*> Map::GetMapLocations()
{
    return LocationsArray;
}

void Map::CreateLocations()
{
    int informationSize = 3;
    std::vector<std::string> information;
    for (int i = 0; i < locationInformation.size(); i += informationSize)
    {
        Location* newLocation = new Location(locationInformation[i], locationInformation[i + 1], locationInformation[i + 2]);
        LocationsArray.push_back(newLocation);
    }
}
```

Finally, was Movement, this is still a work in progress, but the basic commands are in for walking and looking, the player's location changes if they walk to a connecting location.

Zorkish Movement

```
Movement::Movement(Map* a_Map, Player* a_Player)
{
    MapLocations = a_Map;
    PlayerCharacter = a_Player;
    bIsMoving = false;
    bIsLooking = false;
}

void Movement::ChangeLocations(std::string a_MovementText)
{
    /* Moving (Need to implement this better) */
    std::string CommandWord = a_MovementText.substr(0, a_MovementText.find(" "));

    for (const auto& MoveWord : Commands.MoveWords)
    {
        if (CommandWord == MoveWord)
        {
            bIsMoving = true;
        }
    }

    if (bIsMoving)
    {
        std::vector<std::string> LocationConnections = PlayerCharacter->GetCurrentLocation()->GetLocationConnections();

        for (const auto& location : LocationConnections)
        {
            if (a_MovementText.find(location) != std::string::npos)
            {
                for (const auto& ChosenLocation: MapLocations->GetMapLocations())
                {
                    if (ChosenLocation->GetLocationName() == location)
                    {
                        std::cout << "Moving to Location: " << location << "\n";
                        PlayerCharacter->ChangeLocations(ChosenLocation);
                        bIsMoving = false;
                    }
                }
            }
        }
    }

    /* Looking (Need to implement this better) */
    for (const auto& LookWord : Commands.LookWords)
    {
        if (CommandWord == LookWord)
        {
            bIsLooking = true;
        }
    }

    if (bIsLooking)
    {
        std::cout << PlayerCharacter->GetCurrentLocation()->GetLocationDescription() << "\n";
        std::cout << "Off in the distance you see: ";
        for (auto const& location: PlayerCharacter->GetCurrentLocation()->GetLocationConnections())
        {
            std::cout << location << " ";
            bIsLooking = false;
        }
        std::cout << "\n";
    }
}
```

What we found out:

This was quite challenging to setup and I still need to refine the Movement Class to make it be more flexible, test out more scenarios but in terms of functionality it performs the required tasks.