Spike: 07
Title: Graphs

Author: Ford Killeen, 9731822

Goals / deliverables:
Goals this spike aims to achieve:

- Design a text file format that details an adventure and can be read into the Zorkish game
- Load an adventure in via text file and store the world locations in a graph
- Be able to move between world locations within the graph in any location

Deliverables required:

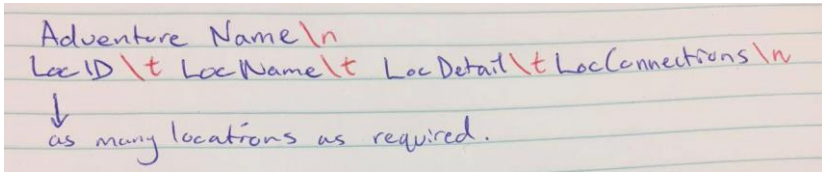
- Hand written text file format design
- Code showcasing the game world graph and “go” commands
- Spike report

Technologies, Tools, and Resources used:
The following is required to complete this spike:

- Visual Studio 2015
- Zorkish game specification
- Online C++ references
- A text file adventure of your own making

Tasks undertaken:
The list below details the steps taken to complete this spike.

- The first task completed for this spike was coming up with a text file format that could be used to create adventures. The format would need to be able to work with adventures of different sizes and also allow easy reading of all adventure data. I used a simple tab delimited text file, with the first line of the file being the adventure name and the remaining lines being the locations. See paper design and example below.



Test Adventure.txt

```
Test Adventure
1   Blue Area   You are in a blue area. In front of you there is a red glow, behind you a yellow light. forward>2+back>4
2   Red Area    You are in a red area. Behind you the glow of a blue light, in front of you a green light.  back>1+forward>3
3   Green Area  You are surrounded by green, nothing around you except for a red glow in the distance behind you.   back>2
4   Yellow Area You are in a bright yellow area, with nothing around except a blue glow ahead of you in the distance.   forward>1
```

- Next I worked on being able to read the text file before I worried about setting up any locations or graphs. Once I was able to successfully read something from file, I continued on to the next step.
- The `Adventure` and `Location` class definitions were set up. The basis for our graph lies within the `Location` class, as it contains a `map` with all of its connections to other world locations.

```
class Location
{
public:
    Location();
    Location(int _id, string _name, string _detail);
    ~Location();
    void setId(int _id);
    int getId();
    void setName(string _name);
    string getName();
    void setDetail(string _detail);
    string getDetail();
    void addConnection(string command, int connection);
    map<string, int>& getConnections();
private:
    int id;
    string name;
    string detail;
    map<string, int> *connections;
};
```

```
class Adventure
{
public:
    Adventure();
    ~Adventure();
    void setName(string _name);
    string getName();
    void addLocation(int id, Location* loc);
    map<int, Location*> getAllLocations();
    Location &getCurrentLocation();
    void setCurrentLocation(int _newloc);
private:
    string name;
    int currentLocation;
    map<int, Location*> locations;
};
```

- Once these classes were set up, I adjusted the file input to create the adventure and location objects on read.
- I then ran into the issue of being able to use the adventure I had loaded in the `SelectAdventureState` within my `GameplayState`. Because I didn't want to re-read the input files again I created another class `Adventures` which is a singleton and stores all of the `Adventure` objects created from adventure text files. This allows me to read in the adventure files once and access them over and over again from any state within the game.

```
cout << "Starting : " << Adventures::instance().currentName() << endl;
```

What we found out:

By completing this spike we found out the advantages of using graphs to map out the world within a game. By using connections between locations, we were easily able to move between locations without needing a grid or specific layout to our world. It can be unconventional and have locations in any direction possible, and still function perfectly.

Issues:

One of the main issues for me when completing this spike was my lack of experience using pointers within C++, and how to read in all files from a directory and split the strings using a set delimiter. It took me a lot longer to get these things done than it should have, purely because I just didn't know how to achieve them in C++ (not a language I use often).