

Spike: Spike No.11

Title: Task 11 – Graphs from Data

Author: Khang Trinh - 102118468

Goals / Deliverables:

The goal of this spike is to teach the developer how to implement graphs from data. The application being used for this spike will be mapping out locations for a text-based adventure game.

Technologies, Tools, and Resources used:

- Visual Studio 2017

Tasks undertaken:

Step 1: Create the graph

A location can have multiple exits, but each exit is unique for that location. To represent this, we're going to use a map. A map has a key and a value, which can be the unique exits and the location that has those exits respectively.

```
class Location {  
public:  
    string name;  
    std::map<string, Location*> exits;  
};
```

Fig 1. Using a map for the graph

Step 2: Assign each node and edges to the map

To help with the example, here's a graph to visualize how the locations will be connected to each other:

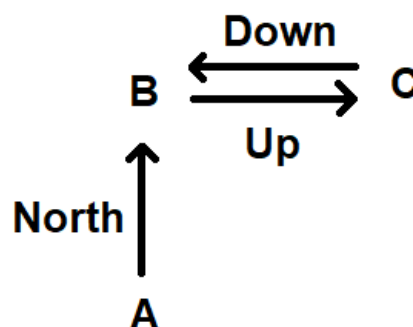


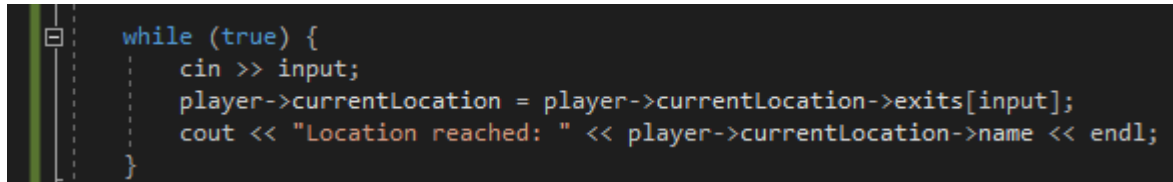
Fig 2. A visual for the locations and exits

```
A->exits.insert(make_pair("N", B));  
B->exits.insert(make_pair("U", C));  
C->exits.insert(make_pair("D", B));
```

Fig 3. Adding locations and exits to the map

Step 3: Usage

To move from one node to another, we tell the map which way we'd like to go from the current node (pass the edge's info). In this particular case, it'll be whichever direction we assigned for the locations in the steps above.



```
while (true) {  
    cin >> input;  
    player->currentLocation = player->currentLocation->exits[input];  
    cout << "Location reached: " << player->currentLocation->name << endl;  
}
```

Fig 4. Switching locations

What we found:

- You can design your own data structure to support your graph (array of list, etc.). Keep in mind the extensibility of your graph (adding/removing nodes)
- You may need to implement some sort of fail-safe code (a command processor in this case) that can handle input outside the range of your container
- There are other applications that can benefit from this implementation. Some examples are, dependency graph (skill tree), navigation graph/pathfinding, etc.