Virtual Reality Visualization with the Oculus Rift

Nicholas Wereszczak, Dr. Christopher Martinez and Dr. Alice Fischer

Tagliatella College Of Engineering

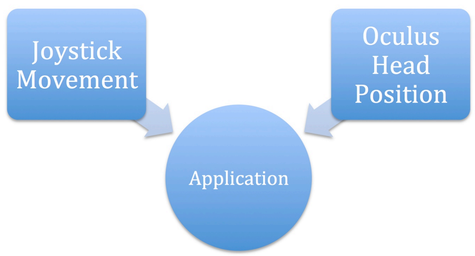
Introduction

- Expand the use of virtual reality into application beyond entertainment
- Explore how to view data in a 3D world
- Develop a technique to express two data sets into a series of coordinates in 3D space
- Show how to navigate through the data

Setup and Tools

- Hardware used: Oculus Rift headset, Logictech game controller, and MacBook
- Software: Xcode, C++, Oculus Software Development Kit (SDK) and Open GL
- Oculus SDK used to project two 2D images into each eye to create a 3D effect
- Oculus SDK also used head movement to change viewing angle
- OpenGL used to create 3D shapes

The Block Diagram to the right shows the inputs into the application.



Layout of Text File

data.txt

```
25
37
47
23
16
34
41
34
50
30
8
20
47
36
11
11
40
9
48
2
4
41
35
38
7
49
```

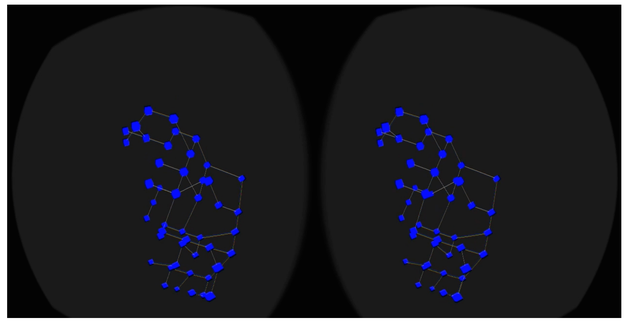
To the left you can see the text file for the complex Tree.

I used basic streams to read/write to and from the text files.

To the right you can see the user entered text file.

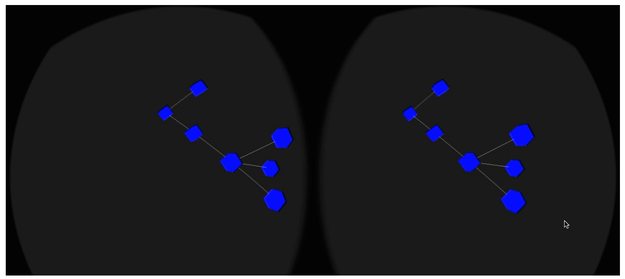
```
Node A
Node B
Node C
Node D
Node E
Node F
Node G

Node A Node B
Node A Node C
Node B Node E
Node E Node F
Node E Node G
Node E Node D
```

Visual Output



Above is an example of a random complex Tree graph.

Below is an example of a simple graph inputted by the user.



Methods

- Created own method to draw a Binary Search Tree (BST) in a three coordinate plane (x,y,z)
- First node is created at reference point (0,0,0)
- If Boolean *_left* is true then it will draw the next node on the left else on the right (adjusted in coordinates, either (-1,-1) or (1,-1)
- Every 6th node it adds a –z coordinate – i.e. (-1,-1,-2)
- Every 4th node it adds a +z coordinate – i.e. (1,-1, 3)

```
if (p!=root) {
    spot = prev->index;
    p->index = num;
    if (_left) {
        if (spot==0) {
            locs->append(((*locs)[spot])+glm::vec3(-0.2f, -0.2f, 0));
            p->el = (*locs)[spot+1];    //changes data in tree
            temp.p1 = (*locs)[spot];
            temp.p2 = (*locs)[num] - (*locs)[spot];
        }
        else {
            if (num%6 == 0)
                locs->append(((*locs)[spot])+glm::vec3(-0.1f, -0.1f, -0.2f));
            else if (num%4 == 0)
                locs->append(((*locs)[spot])+glm::vec3(-0.1f, -0.1f, 0.3f));
            else locs->append(((*locs)[spot])+glm::vec3(-0.1f, -0.1f, 0));
            p->el = (*locs)[spot+1];    //changes data in tree
            temp.p1 = (*locs)[spot];
            temp.p2 = (*locs)[num] - (*locs)[spot];
        }
        lines->append(temp);
    }
    else if (!_left) {
        if (spot==0) {
            locs->append(((*locs)[spot])+glm::vec3(0.2f, -0.2f, 0));
            p->el = (*locs)[spot+1];    //changes data in tree
            temp.p1 = (*locs)[spot];
            temp.p2 = (*locs)[num] - (*locs)[spot];
        }
        else {
            if (num%6 == 0)
                locs->append(((*locs)[spot])+glm::vec3(0.1f, -0.1f, -0.2f));
            else if (num%4 == 0)
                locs->append(((*locs)[spot])+glm::vec3(0.1f, -0.1f, 0.3f));
            else locs->append(((*locs)[spot])+glm::vec3(0.1f, -0.1f, 0));
            p->el = (*locs)[spot+1];    //changes data in tree
            temp.p1 = (*locs)[spot];
            temp.p2 = (*locs)[num] - (*locs)[spot];
        }
        lines->append(temp);
    }
```

Conclusions

- Programmed in C++ and used OpenGL and Oculus SDK
- Also used multiple 3rd Party Components such as glew, glm & zlib
- User can enter their own data to create a custom graph
- Can change color and shape of nodes

Future Improvements

- Work is needed to adjust joystick controls
- Computers with higher processing power can create larger graphs
- Work can be done to have multiple graphs at once
- The research done for this project can be used in future research, other projects, or can pick up where this project left off