Alex Ruef
ruefa@oregonstate.edu
Final Project Report

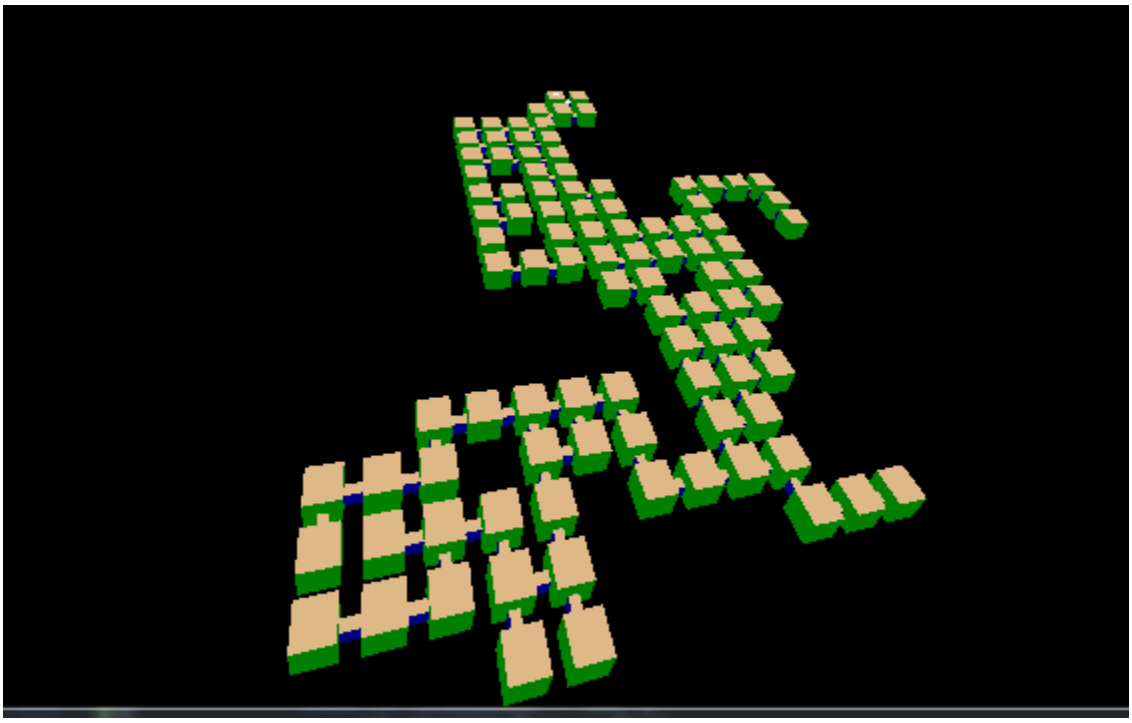
Project Proposal

I have put more hours than I would like to admit into dungeon crawler games like Diablo and Path of Exile. For my project I would like to implement randomly generated dungeons that are a cornerstone of these games. Generating the dungeon layouts would be the main focus of this project but more can be added to meet the final project requirements.

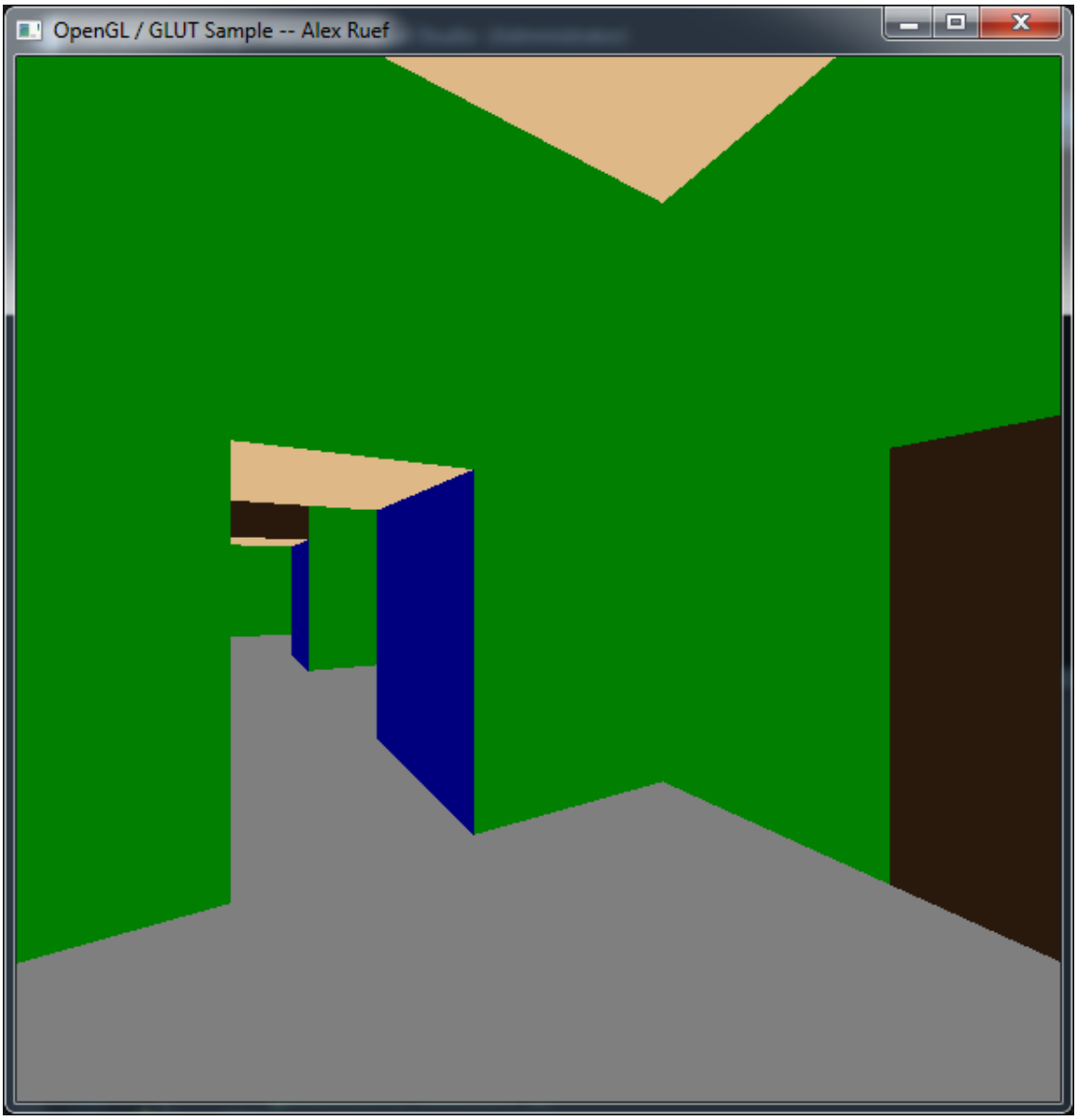
To implement the dungeons, I would generate “boxes” that would represent a room and smaller rectangles would represent corridors connecting the rooms. I would also like to have differing levels by varying the Y of each room and connecting them with a staircase. Things that would be randomized would be room positions (XYZ), room connections (north, south, east, west), number of connections on a given room, and room sizes. To give each room some uniqueness I will look for some wall textures online or come up with something on my own. These wall textures can be randomly chosen as well. I will also use lighting and have simple light fixtures in each room.

If what I have said already is not enough the next thing I would add is interactivity. I already plan to have the user walk through the dungeon by manipulating the camera. A simple application of this would take the form of doors that can be opened / closed. More advanced applications can be small puzzles to get through each room. I could create a list of puzzles and randomly add them to each room. These would all be accompanied with animations such as the door opening and closing.

What I did

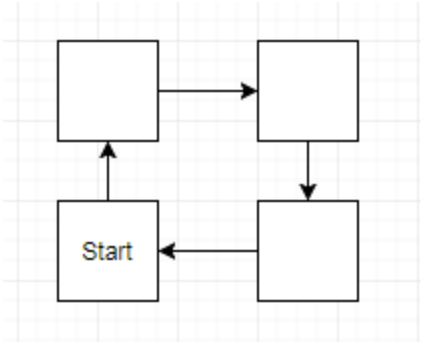


I made a randomly generated dungeon layout! It ended up being more maze like but for a first prototype I'm happy with it. The rooms can be traversed by using the WASD keys which move the camera position in the rooms. When standing near a door the e key can be pressed to open and close the doors.



Differences from proposal

I initially bit of more than I could chew with my proposal. Due to design flaws early on I had to cut lighting and textures. To generate the rooms I use a recursive tree like structure that I will discuss a little more later. The problem came when I did not realize that the rooms can be generated in a loop which creates a cycle and forms infinite loops when iterating through the tree.



To solve this issue I created a linked list with the XYZ coordinates of each room. Then I prevented any room from creating or linking to a room that existed in the linked list. The second issue I ran into was interactivity. Getting WASD controls working to move through the rooms was more difficult than I thought but doable. Opening and closing doors however was not easy due to my initial design of the doors.

While working on this project I did not consider how I would determine how to open and close doors or how to even determine what door the user was trying to open. All doors are drawn from a single display list and then translated based on the rooms XYZ position and position of the wall the door should go on. Because of this I had no information to identify each individual door for interaction. My solution was to first iterate through the room tree structure to find what room the user is in based on camera position. Once I have the room I used geometry to identify 4 areas in the room, 1 area around each door. Then I check if the users camera position is within one of those areas and if it is, I add the door to a linked list struct I had to create for the purpose of door interactivity. The animate function iterates through the linked list which contains currently active doors. Once the door is finished moving it is removed from the list. I should have made each door its own entity from the beginning to avoid all this extra work and headache.

Impressive cleverness

My favorite part of working on this was writing the room generation logic.

```
struct room {
    float x, y, z;
    GLuint  wallList;
    struct room *door0 = NULL;
    struct room *door1 = NULL;
    struct room *door2 = NULL;
    struct room *door3 = NULL;

    struct doorList *door_trans0 = NULL;
    struct doorList *door_trans1 = NULL;
    struct doorList *door_trans2 = NULL;
    struct doorList *door_trans3 = NULL;
}headRoom;
```

Each room is a struct containing information on itself and its position. It also doubles as a tree data structure to hold all the rooms together. The door_trans variables are an unfortunate side effect of the issues I discussed in the previous section.

To generate each room I started at the headRoom and recursively iterate through each room pointer until a maximum number of rooms is reached or randomly a room was decided it would get no additional connections. Once each room is created I iterate through the tree structure to create their display lists. I then iterate through the tree again in display() to call the display lists.

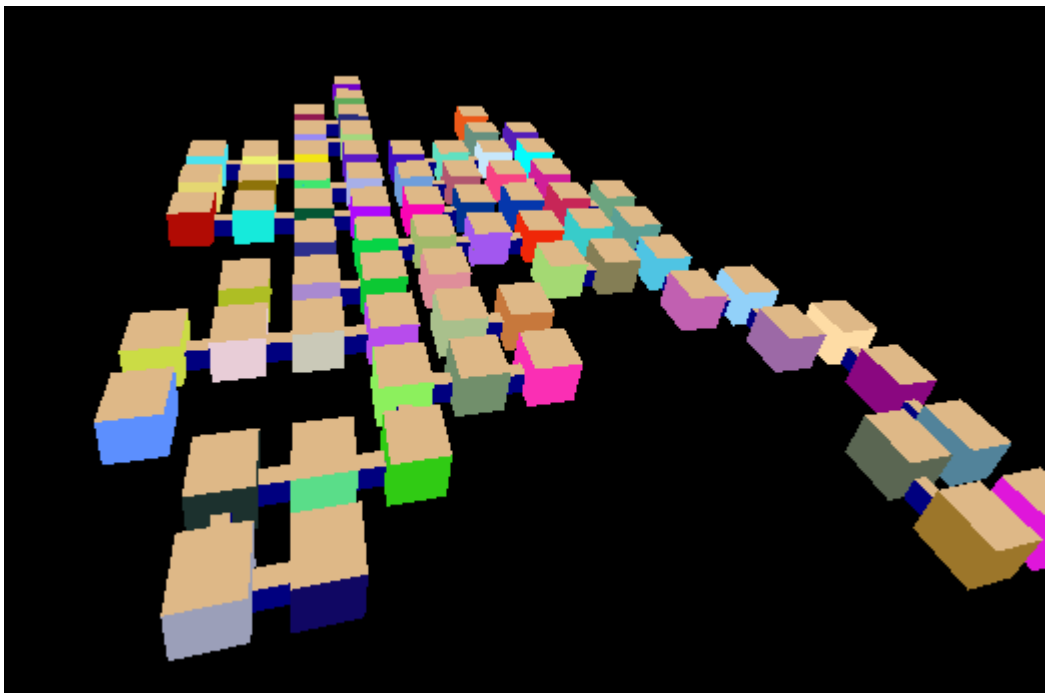
What I learned

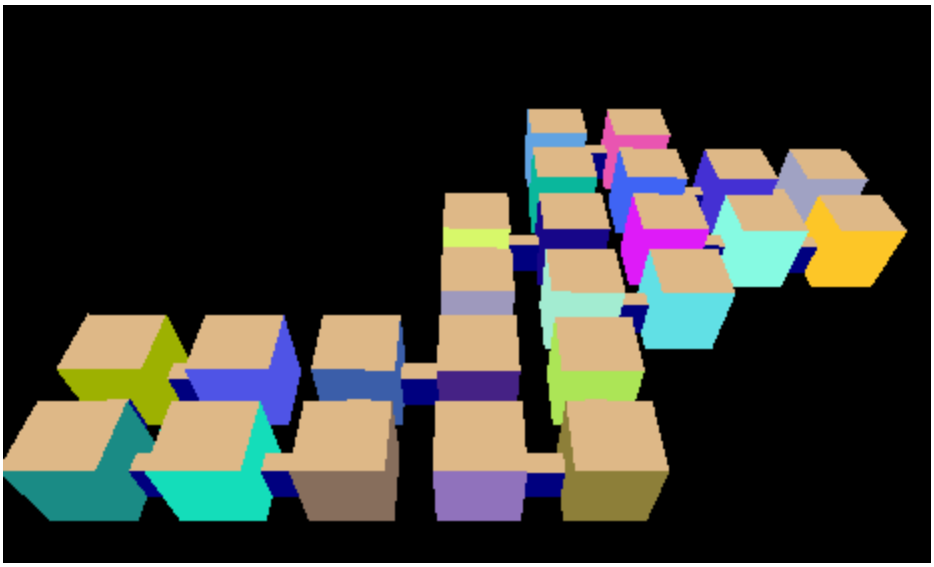
This project taught me a lot about planning and designing. As a student of computer science I have not done a lot of larger projects and thus there are many things I never think to consider when planning. With every project I learned something new that I need to include in my designs to cause less headaches during implementation.

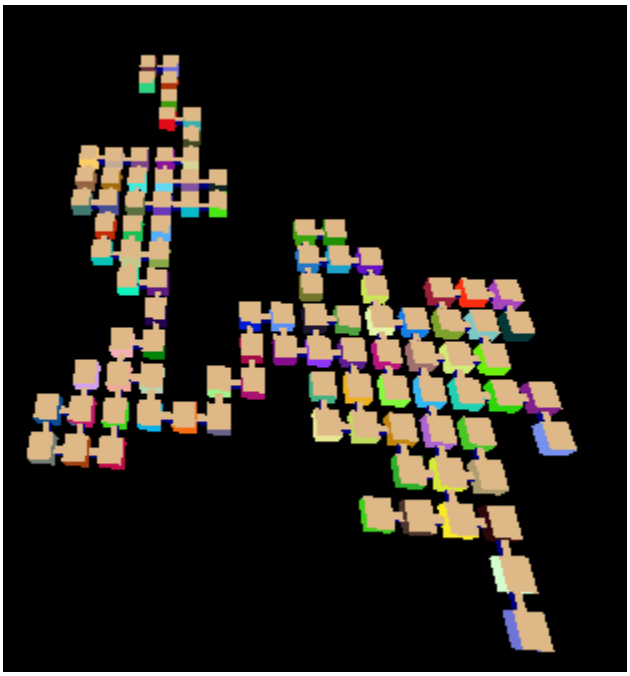
I learned about C and the power of pointers and recursion. I also learned that pointers and recursion can be dangerous and difficult to debug. I learned a lot of quirks about C pointers such as if you pass a pointer into a function and then malloc data for it, the original pointer still points to the old location.

Project Images

When you are inside a room it hard to see differences except in room color. Move the camera outside the rooms and you can see a wildly different structure each time the program is ran.







Video Link

https://media.oregonstate.edu/media/t/0_4rjhqxdi