CS 450 Computer Graphics: Final Project

Daniel Schroeder
CS 450 Computer Graphics
Fall 2017

Abstract
This document is a report outlining my final implementation of my CS 450 Computer Graphics final project. This report details how I coded my program, how my final implementation is different from my proposal, and notable displays of OpenGL expertise!

Contents

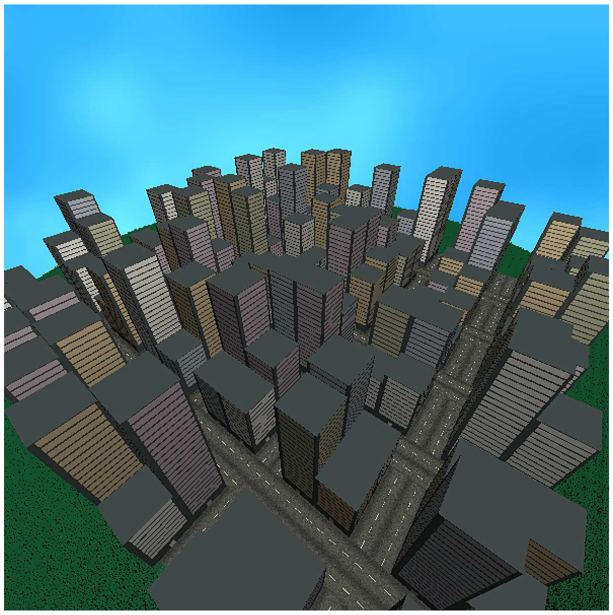
1 What you actually did for your project, with images

1.1 Town Layout

```
int town[25][25] = {
    {2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2},//1
    {1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1},//2
    {1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1},//3
    {2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2},//4
    {1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1},//5
    {1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1},//6
    {2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2},//7
    {1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1},//8
    {1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1},//9
    {2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2},//10
    {1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1},//11
    {1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1},//12
    {2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2},//13
    {1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1},//14
    {1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1},//15
    {2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2},//16
    {1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1},//17
    {1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1},//18
    {2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2},//19
    {1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1},//20
    {1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1},//21
    {2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2},//22
    {1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1},//23
    {1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1, 3, 3, 1},//24
    {2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2}//25
};
```

The town is generated from a 25x25 matrix with different integer values for the different objects at that position. A 3 indicates a building with a random height and 1's and 2's indicate road squares. This layout generates a 2-block grid with cross-hatching roads and 2x2 squares with four buildings each. Additionally, I used this grid to calculate the origin of the road textures (i.e. north-south or east-west) by changing the orientation every two columns.

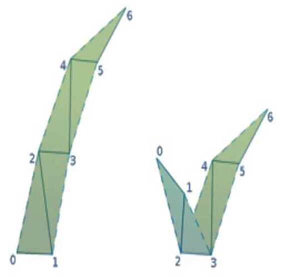
1.2 Building Generation



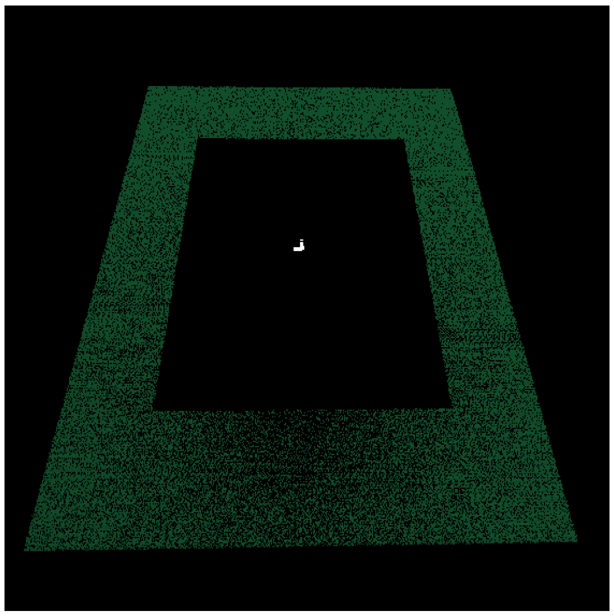
My buildings are generated with random heights and a random neutral window color. The building fragment shader spaces out the windows on each building to be proportionally aligned across the full height of the building and creates a door at the bottom of the building. Each building is comprised of four identical GL_QUADS rotated 90°*i about the center of the building. Then there are solid base-colored GL_QUADS for the floor and roof. This OpenGL draw commands are placed inside a "DrawBuilding" function that takes a width, height, and color as parameters in order to generate randomly sized and colored buildings for each "3" in the town layout grid.

1.3 Grass Shaders

My grass shader was probably the biggest component of the finalized project. I generates a vertex buffer array comprised of 4,840 vertices that used a GL_TRIANGLE_STRIP to create 100 blades of grass in a one-by-one square. A blade of grass consisted of 11 triangles that followed one of the two following constructs:

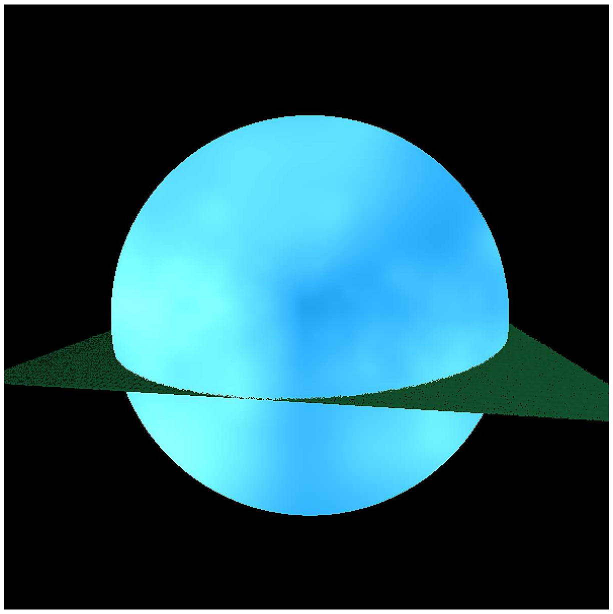


Each blade was randomly rotated some amount between 0°-360°and sent to the grass shader. To draw the entire grass terrain, I loaded my vertex buffer object and used "glDrawArrays(GL_TRIANGLE_STRIP, 0, 4840);" in a double for-loop to effectively render a 40x40 grid (with a 25x25 gap in the center for the town) with each square containing 100 blades of grass that created the outer terrain of the city-scape:



Doing the math, this produced 1,089,000 vertices for all the blades of grass of the terrain. This did not make my GPU happy when I tried to do it all with OpenGL drawing functions, so it forced me to create a vertex buffer object to store on the GPU and drastically reduced the individual drawing calls for each blade of grass from 22,500 to only 225 by storing the each one-by-one patch as a vertex buffer array. Once I could get the grass rendered without any latency issues, I toggles with the ambient, diffuse, and specular in the fragment shader to get a soft, flat glow on the individual blades of grass as they were constantly "blowing in the wind" as a function of a uniform time variable passed into the vertex shader. The movement of the grass changed the position of the z-coordinates proportional to the y-coordinate so the bottom of the grass plate appears "grounded" while the rest of blade sways more vigorously the higher it is.

1.4 Skydome Shader



My skydome object was a basic glutSolidSphere object passed to my cloud glsl shader which generated random noise to simulate clouds. I found a basic example of noise generation online and manipulated the shader to have a flat blue background to simulate a sky. I had an issue getting the skydome to work completely correct and stay fixed as the user changes the gluLookAt position to effectively "look around." The current implementation generates a sky that moves with the user eye position so you are stuck looking at the same portion of the dome the entire time.

1.5 Road Texture Mapping

For my city roads, I used a seamless road texture mapped to the GL_QUAD for every square of street. This created a better looking, continuous road that simply using colors and shaders. The roads also depict direction with long "boulevard" style streets stretching from "North" to "South" and smaller incremental streets connecting the "boulevards" from "east" to "west."

2 How your project differs from what you proposed, and why

Although the aspect of terrain was not explicitly specified in my proposal, I became fascinated with the idea of hyper-realistic terrain rendering and began doing research on concepts like atmospheric scattering, high-density terrain rendering, and OpenGL grass rendering. Generating the buildings with a simple glsl shader proved to be simple enough from my experience from previous projects, so I began to challenge myself by trying to create a hyper-realistic terrain outside of the city itself. The image attached to my proposal was what I expected my final project to look like, but I ended up being able to generate a much more complicated scene with higher detail and more intricate features. For example, I used seamless textures for my roads rather than creating a crude coloring scheme of black with yellow divider lines. I also wanted to add perspective outside the city to simulate a distant horizon rather than keeping the default black distance buffer. To do this I researched and generated a (semi-functional) skydome objects that uses a template algorithm for noise generation by using "Fractional Brownian Motion" and "bilinearly interpolated lattices" from an example (link: *PROCEDURALLY GENERATING NOISE IN A SHADER*) by the team at xdPixel. I manipulated the code to produce a flat blue background with white noise on top to simulate a sky and clouds, rather than the white and black smoke-like example from the website. Additionally, I changed the way my camera moves with user input by creating a key handler with a key buffer rather than handling movement with switch cases in the Keyboard() function from the sample program. I followed the style for fluid movement from another classmate's implementation for his final project found at (Link: *Ben's Github Repository*). This method created fluid user movement rather than staggered, incremental movements generated by individual key presses.

3 Any impressive cleverness you want us to know about

I successfully surprised myself with multiple aspects of this project through the course of its development. Notably, my complex generation of grass objects with complex GL_TRIANGLE_STRIP configurations and trigonometry to calculate random vertex rotation per blade of grass. I would like to include my code because I am just too proud of my methodology. To generate my vertex buffer array, I first created a vertex template for each of the two grass-blade variations shown in Section: Grass Shader. These vertices map out to GL_TRIANGLE_STRIPS that end as a point at y==0 so it can begin drawing the next blade from the ground up rather than at the top of the previous blade (resulting in a connected triangle between the two).

```
GLfloat one[][3]={
    {0.000000, 0.000000, 0.000000},
    {0.000000, 0.000000, 0.000000},
    {0.030000, 0.000000, 0.000000},
    {0.015000, 0.060000, 0.000000},
    {0.042500, 0.060000, 0.000000},
    {0.039000, 0.120000, 0.000000},
    {0.062500, 0.120000, 0.000000},
    {0.090000, 0.162500, 0.000000},
    {0.062500, 0.120000, 0.000000},
    {0.042500, 0.060000, 0.000000},
    {0.030000, 0.000000, 0.000000},
    {0.030000, 0.000000, 0.000000}
};
GLfloat two[][3]={
    {0.015000, 0.000000, 0.000000},
    {0.015000, 0.000000, 0.000000},
    {0.000000, 0.055000, 0.000000},
    {0.023000, 0.350000, 0.000000},
    {0.015000, 0.000000, 0.000000},
    {0.050000, 0.000000, 0.000000},
    {0.050000, 0.055000, 0.000000},
    {0.075000, 0.055000, 0.000000},
    {0.110000, 0.100000, 0.000000},
    {0.075000, 0.055000, 0.000000},
    {0.050000, 0.000000, 0.000000},
    {0.050000, 0.000000, 0.000000}
};
```

After generating the template vertices, I perform a double for-loop in the InitGraphics() function that generates ten rows of ten blades of a random choice between blade style one or blade style two. This for-loop also calculates the position of the vertices after a random rotation from 0°-360°. These vertices are printed to a text file as a list of size GLFloat[4840][3].

```
FILE * pFile;
pFile = fopen ("myfile.txt","w");
float i;
float j;
int rot;
float rad;
int r;
```

```
for (i = 0; i < 1; i+=0.05){
for (j = 0; j < 1; j+=0.05){
    r = rand()%2;
    rot = rand()%360;
    rad =rot*(M_PI/180);
    if (r == 1) {
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (one[0][2]*sin(rad) + one[0][0]*cos(rad)) + j, 0.000000, (one[0][2]*cos(rad) - one[0][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (one[1][2]*sin(rad) + one[1][0]*cos(rad)) + j, 0.000000, (one[1][2]*cos(rad) - one[1][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (one[2][2]*sin(rad) + one[2][0]*cos(rad)) + j, 0.000000, (one[2][2]*cos(rad) - one[2][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (one[3][2]*sin(rad) + one[3][0]*cos(rad)) + j, 0.060000, (one[3][2]*cos(rad) - one[3][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (one[4][2]*sin(rad) + one[4][0]*cos(rad)) + j, 0.060000, (one[4][2]*cos(rad) - one[4][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (one[5][2]*sin(rad) + one[5][0]*cos(rad)) + j, 0.120000, (one[5][2]*cos(rad) - one[5][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (one[6][2]*sin(rad) + one[6][0]*cos(rad)) + j, 0.120000, (one[6][2]*cos(rad) - one[6][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (one[7][2]*sin(rad) + one[7][0]*cos(rad)) + j, 0.162500, (one[7][2]*cos(rad) - one[7][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (one[8][2]*sin(rad) + one[8][0]*cos(rad)) + j, 0.120000, (one[8][2]*cos(rad) - one[8][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (one[9][2]*sin(rad) + one[9][0]*cos(rad)) + j, 0.060000, (one[9][2]*cos(rad) - one[9][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (one[10][2]*sin(rad) + one[10][0]*cos(rad)) + j, 0.000000, (one[10][2]*cos(rad) - one[10][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (one[11][2]*sin(rad) + one[11][0]*cos(rad)) + j, 0.000000, (one[11][2]*cos(rad) - one[11][0]*sin(rad)) + i);
    }
    else{
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (two[0][2]*sin(rad) + two[0][0]*cos(rad)) + j, 0.000000, (two[0][2]*cos(rad) - two[0][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (two[1][2]*sin(rad) + two[1][0]*cos(rad)) + j, 0.000000, (two[1][2]*cos(rad) - two[1][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (two[2][2]*sin(rad) + two[2][0]*cos(rad)) + j, 0.055000, (two[2][2]*cos(rad) - two[2][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (two[3][2]*sin(rad) + two[3][0]*cos(rad)) + j, 0.350000, (two[3][2]*cos(rad) - two[3][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (two[4][2]*sin(rad) + two[4][0]*cos(rad)) + j, 0.000000, (two[4][2]*cos(rad) - two[4][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (two[5][2]*sin(rad) + two[5][0]*cos(rad)) + j, 0.000000, (two[5][2]*cos(rad) - two[5][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (two[6][2]*sin(rad) + two[6][0]*cos(rad)) + j, 0.055000, (two[6][2]*cos(rad) - two[6][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (two[7][2]*sin(rad) + two[7][0]*cos(rad)) + j, 0.055000, (two[7][2]*cos(rad) - two[7][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (two[8][2]*sin(rad) + two[8][0]*cos(rad)) + j, 0.100000, (two[8][2]*cos(rad) - two[8][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (two[9][2]*sin(rad) + two[9][0]*cos(rad)) + j, 0.055000, (two[9][2]*cos(rad) - two[9][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (two[10][2]*sin(rad) + two[10][0]*cos(rad)) + j, 0.000000, (two[10][2]*cos(rad) - two[10][0]*sin(rad)) + i);
        fprintf(pFile, "{%f,␣%f,␣%f},\n", (two[11][2]*sin(rad) + two[11][0]*cos(rad)) + j, 0.000000, (two[11][2]*cos(rad) - two[11][0]*sin(rad)) + i);
    }
}
fprintf (pFile, "{%f,␣%f,␣%f},\n",0.000000+j, 0.000000, 0.000000+i);
fprintf (pFile, "{%f,␣%f,␣%f},\n",0.000000+j, 0.000000, 0.000000+i);
}
fclose (pFile);
```

I then copied the contents of this text file into the global GLfloat array that I use for my vertex buffer array. I can effectively generate a new configuration of the one-hundred-blade 1x1 squares with every new run of the program.

4 What you learned from doing this project

This project exposed me to numerous new elements of OpenGL that I previously had no experience with. Most notable are noise generation from my skydome shader, vertex buffer objects used for my grass rendering, and highly technical trigonometric math to perform rotations and simulate fluid movement. Descriptions of these different components are explained, in detail, in the previous sections. Most of the other techniques were implementations of strategies acquired from previous assignments, but executed to a much higher level of detail and technicality like all the shaders, texture mapping, and basic OpenGL geometry.

5 A link to the video showing off your project

Link to Kaltura Video: https://media.oregonstate.edu/media/t/0_d73n622x

6 Original Project Proposal

Daniel Schroeder
schrodan@oregonstate.edu
Final Project Proposal

For my final project, I would like to create a city-scape where the user can "move" around and traverse the streets. Similar to the video we saw flying over Chicago, I would like to create a scene with buildings of different shapes and sizes but with the eye positing at ground level. This program will involve shaders for coloring the buildings, geometric modeling for generating the buildings, and more versatile use of gluLookAt and eye position techniques to allow the user to "move" and "look" around. I think this project should be worth 200 points because it combines a lot of different aspects from previous programs and applies the different techniques to multiple objects (buildings) in order to create a large scene. I've included a crude image of what the city-scape might look like.