Polygonal Meshes

Discussion

The goal of this assignment is to display the Stanford bunny from an OBJ file in OpenGL. I used the code for lab 2 as the basis for my solution since it already has code for displaying an OBJ file. I zoom in to the bunny by first changing the `fovy` in `glm::perspective()` to 0.03, then modifying both the `eyey` position and sphere radius `r` to 10.0. To rotate the bunny up along the y axis, I create a 4x4 identity matrix `model` and apply a rotation to it by calling `glm::rotate()`.

The lighting involves using fragment shaders and computing the normal. I used the original fragment shaders from lab 2 and calculated the normal manually. An array of containing `glm::vec3()` vectors, `normals`, is used to store the average normal of each vertex. This array would then be converted to a pointer of type `GLfloat` and passed to `glBufferSubData()` as our data. I followed the pseudocode from the lecture slides to calculate the normals.

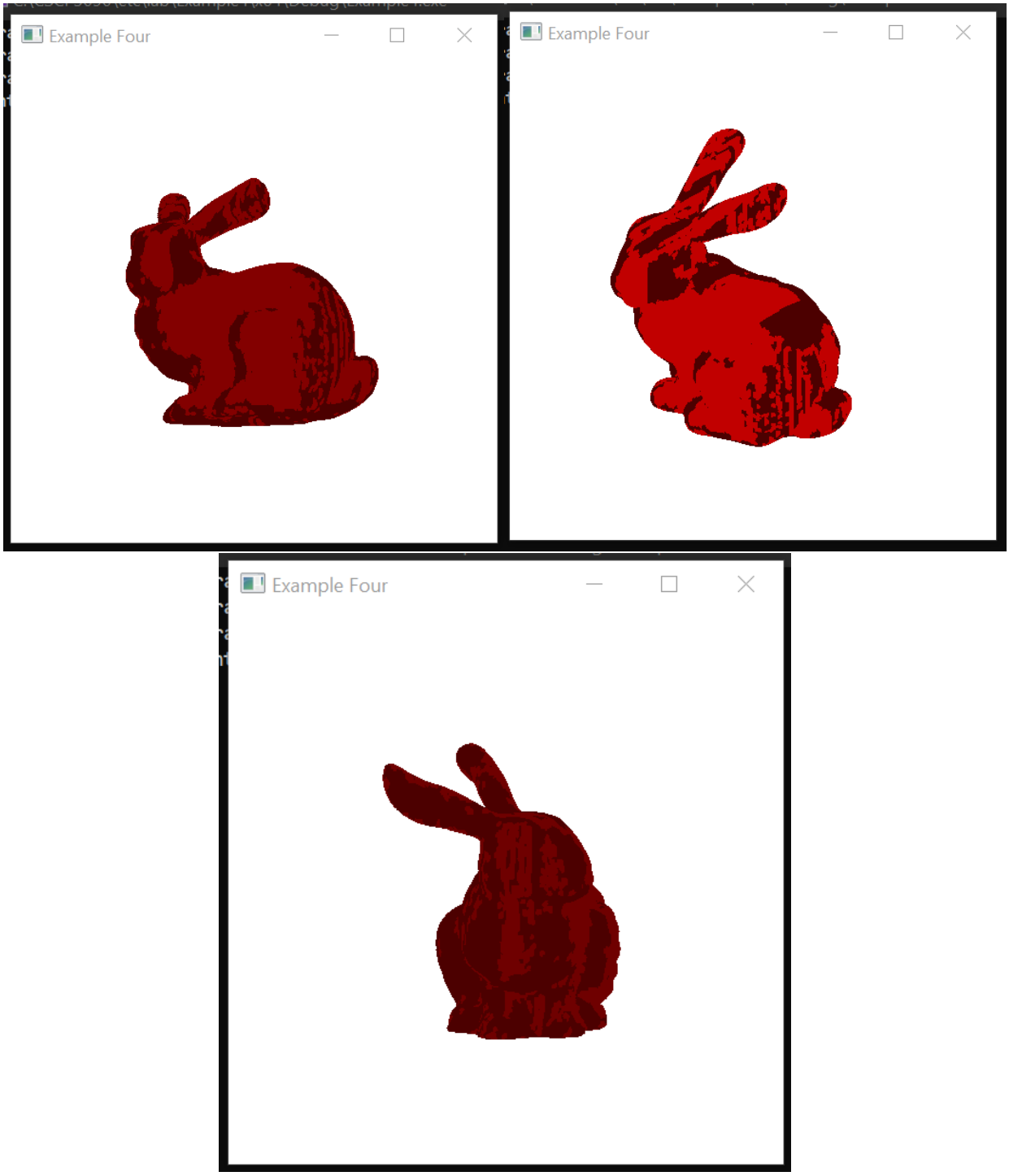
According to the pseudocode, we must first calculate the normals of each polygon. To do this, construct a normal table and initialize it to 0. Our table is an array of size `nv/3` containing `glm::vec4()` vectors, which has columns `x`, `y`, `z`, and `w`. Also, initialize the polygon count to be 1. Next, iterate through each triangle and obtain three vertices. Use the `indices[]` array to find the vertex number and use this vertex number to find coordinates for a vertex in the `vertices[]` array.

Using these vertices, construct two vectors, `u` and `v`, and find its cross product using `glm::cross()`. This gives us a normal vector for each polygon, which we then normalize using `glm::normalize()`. The normalized vector would be added to the vertex entry in the table (columns `x`, `y`, and `z`) and the number of polygons traversed would be added to column `w`. Increment the polygon count by one and repeat for all vertices in the polygon.

With the table filled, we now find the vertex normals. Iterate over each row in the normal table. Retrieve columns `x`, `y`, and `z` as a `glm::vec3()` vector and divide this vector by the polygon count in column `w`. This gives us the average normal vector of a vertex. Normalize this vector using `glm::normalize()` and add the result to `normals`. Additionally, we calculate the number of vertex normals for the size of the data store region being replaced in `glBufferSubData()`.

A known problem is that one of the normals are likely being calculated incorrectly. As a result, the bunny does not have even lighting on its surface. The data structures used in the lab 2 code were a bit confusing. I may have retrieved my vertices incorrectly because of the way I traversed through the `vertices[]` and `indices[]` arrays. Either I have done one of these methods correctly or I have done both of them incorrectly, all from trying to interpret the pseudocode.

Screenshots



Build Instructions

These instructions only apply to the Windows operating system.

1. Download and extract the .zip file from Canvas
2. Add `bunny.obj` to the assignment folder
3. Open `Example4.sln` in Visual Studio 2019
4. Open Solution Explorer. Check to see if `lab2.fs` and `lab2.vs` are there
    a. If not, add these files by right-clicking Example4 → Add → Existing Item, then add both files.
5. Press `F5` to build and run the solution.
    a. Make sure you have selected `x64` as your Solution Platform before build.
    b. It takes around five minutes for the bunny to appear on the output window.