CSCI 441 - Lab 04

Friday, September 20, 2019

LAB IS DUE BY **FRIDAY SEPTEMBER 27 11:59 PM**!!

Today, we'll be enhancing our flight simulator visually by adding lighting around our scene. Please note: We will not be using the specular or ambient components in this lab. We will only be using the diffuse component. For your future projects/assignments, you must make use of all three components to make more realistic looking materials and objects.

Please answer the questions as you go inside your README.txt file.

Step 0 – New Library File

You'll notice an include folder with this lab. Copy the OpenGLUtils.hpp file into your include/CSCI441 folder as this gives us some additional helper functions.

Step 1 – Using Materials

Start off by building the existing code and run it. Be sure to read the README for the flight controls.

How's everything look? Pretty huh?

Well it's all about to change mwahahahaha. Let's start by saying goodbye to glColor calls. Locate `TODO #01`. These two lines were telling OpenGL to convert whatever color we passed in to the corresponding material properties. Delete these lines, we don't want them anymore. Compile and run again.

How's everything look?

That's somewhat sad. Let's bring back the color by coloring our buildings to bring our scene back to life. Locate `TODO #02`. Previously we were setting each building to a random color. Now, we want to use material properties instead of color.

Let's first set a random diffuse material to the front and back face of our objects (needed code is on slides). Be sure to set the alpha channel equal to 1.0.

Compile and run.

How's everything look? What about our teapot & plane? (The teapot is located at the origin of our world…fly in to find it)

Well the buildings look good, but now the plane is the same color as the teapot! Let's make the teapot look a little more fancy. Choose a material of your choice (from here http://devernay.free.fr/cours/opengl/materials.html). Set the appropriate properties to the teapot at `TODO #03`.

Compile and run.

And now our plane has the same properties as the teapot. It's left as an exercise to the reader to set properties for each plane component like it was previously colored, but this is not required (nor worth extra credit).

Step 2 – Creating a Floor

We'd like something more interesting than just the grid. At `TODO #04`, change the `GL_LINES` call to `GL_TRIANGLE_STRIP` and make a quad out of the four points – not a triangle. Be careful of winding order as well! Compile and build as needed.

Do you have an all white floor? What order did you specify the vertices in?

That will work for now. Let's move on.

Step 3 – Toying With Our First Light

We already have a light in our scene, but we need to make it better.

Run again and take note of *how* the buildings are lit.

We're going to do two things. First we need to make sure we are placing it following good practices. As stated on the slides, we want to position our light immediately after we call `glm::lookAt()` and multiply by the view matrix. Move the calls for placing GL_LIGHT0 to the appropriate place (`TODO #05`).

Compile and run.

Voila! What changed?

Now our light point is being passed through our vertex transformation pipeline and properly placed into the scene. Cool. Let's play with the colors a bit for Light0. Let's turn the diffuse down, it's way too bright (`TODO #06`). Choose a color you like, but make it dim (e.g. close to 0, no larger than 0.2 for each component. 0.0 would be black/dark and 1.0 would be white/bright). Compile and run again.

Are the sides of the buildings facing the teapot brighter than the sides of the buildings facing away from the teapot?

Step 4 – Making Our Floor Better

When we made our grid before, we turned lighting off to make everything white. We need to set our floor to use materials. Remove the lines of code near our grid setup that turns lighting off/on (`TODO #07`). We now need to replace the glColor() call with a material instead. Choose the emerald material for the diffuse properties. Compile and run.

Can you notice any lighting effects on the ground?

Since we are working with OpenGL primitives, we need to specify a normal for every vertex in our primitive. Recall we cannot use the surface normal as determined by the winding order because we may want to specify non-perpendicular normals. Lighting is computed per-vertex, so specify a vertex normal for every vertex in the quad comprising our floor. The normals should all align with the positive Y-axis. Compile and run.

Now does the floor appear brighter near the center and darker near the edges?

Great. Let's add a second light.

Step 5 – Light #2 (But we call it LIGHT1 since we count from zero)

Find `TODO #08`, this is where we will set up our light since the colors and type will never change. We want this to be a white light so set the diffuse color to white (and nothing for specular or ambient).

This light will be a spotlight so we need to set some different properties. Our spotlight will have a small projection cone, so set the cutoff angle to 12 degrees. Don't forget to turn the light on!

Now let's place our second light. `TODO #09` puts it in place for you. This position corresponds to the front of our plane. If we were to look at our matrix stack right now, there has been a series of rotations, translations, more rotations, etc. We want our light to be right here. Since our object is structured hierarchically, we will place this light at the local origin right now. Additionally, we want our spotlight to be pointing along the positive Y-axis. (After all the rotations, this will be pointing forwards.) Compile and run.

Do you see a spotlight projecting from the plane? As you fly around, does the light move with you?

Let's make it look more realistic and have the edges of our spotlight darker than the center. Adding one more line of code in our light setup, set the spotlight's exponent to 100 (we want a drastic effect!). Compile and run.

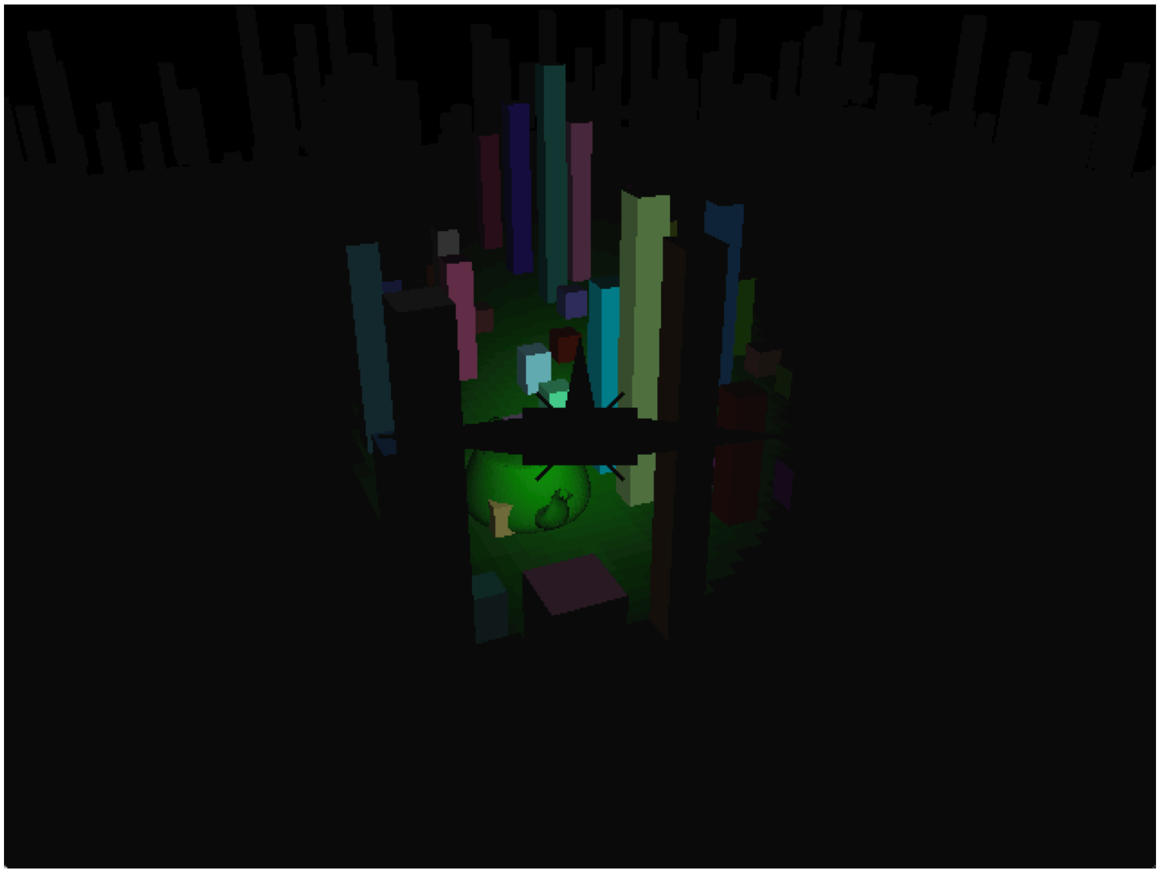
Do the edges of the spotlight seem softer?

We're going to do one last thing…

Step 6 – Flip the Switch

Add flags to save the state of each light, whether it is on or off. When the '1' key is pressed, then LIGHT0 should turn on or off. Likewise, when '2' is pressed, then LIGHT1 should turn on or off. You should know where these calls need to go. Compile and run.

When you turn LIGHT0 off and LIGHT1 on, you should see a scene similar to the image below. Be carefully flying around with both lights off though!!



When you turn LIGHT0 off, can you still see a faint outline of the city against the background? Well, OpenGL adds a small amount of ambient light to everything by default (0.2, 0.2, 0.2). Luckily, we can change this value. In our light setup code, let's add a new line to set the light model.

The function call looks like follows:

```
glLightModelfv( GLenum name, GLfloat *values );
```

We will pass in `GL_LIGHT_MODEL_AMBIENT` for the name and pass in a color array set to (0, 0, 0, 1) for the values. Compile and run one last time. When you turn both lights off, is everything completely dark now? Great! You're done.

Q1: Was this lab fun? 1-10 (1 least fun, 10 most fun)
Q2: How was the write-up for the lab? Too much hand holding? Too thorough? Too vague? Just right?
Q3: How long did this lab take you?
Q4: Any other comments?

To submit this lab, zip together your source code, Makefile, screenshot, and README.txt with questions. Name the zip file <HeroName>_L04.zip. Upload this on to Canvas under the L04 section.

LAB IS DUE BY **<u>FRIDAY SEPTEMBER 27 11:59 PM</u>**!!