CS 450/550 -- Fall Quarter 2020
Project #4

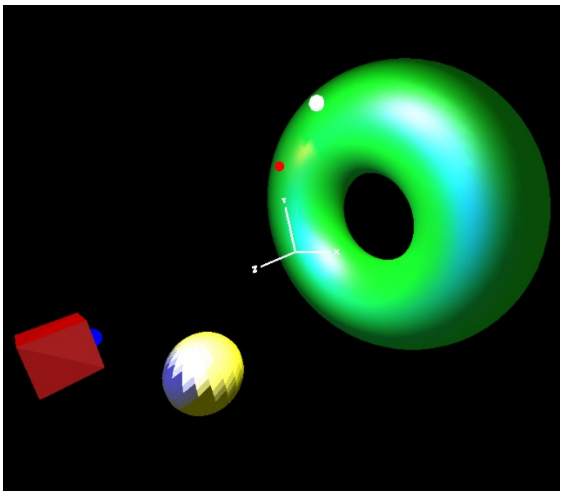
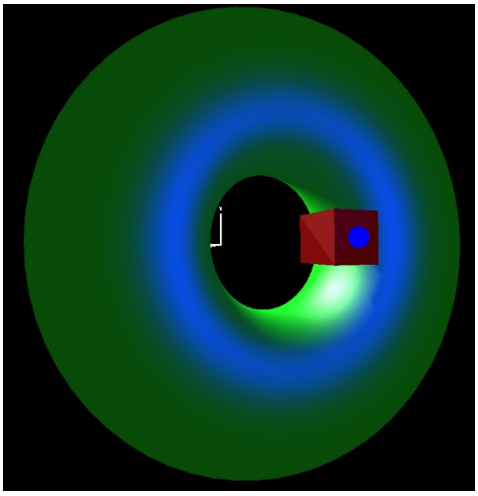
100 Points

Due: November 2

Lighting

This page was last updated: July 27, 2020

Introduction

The goal of this project is to create a 3D animated scene that demonstrates OpenGL lighting.

Requirements:

1. Create a 3D scene consisting of at least 3 objects and at least 3 light sources:

Category	Minimum number	Minimum number
Light motion	2 stationary	1 moving
Light types	1 point	1 spotlight
Light colors	1 white	1 color
Object material	1 shiny	1 dull
Object glShadeModel	1 GL_FLAT	1 GL_SMOOTH
Object motion	1 stationary	1 moving

2. Notice that there only need to be 3 light sources total. Clearly you can be double-dipping with the motion, types, and colors.
3. Be sure to spread things out enough that we can see the effect of the 3 lights.
4. Use RGB color for each object's GL_FRONT.
   Use a shade of gray for each object's GL_BACK.
5. Put small spheres at the light source locations so that you and I can see where they are. Make each sphere the same color as the light source it is representing. Don't use lighting on the light-source spheres. Just make them **glColor3f** blobs.
6. Feel free to use the GLUT solids for your objects. All have normals included.
7. Or, you can design your own 3D objects. Just remember that they will need, at the least, per-face surface normals. If they are to be smooth-shaded, they will need per-vertex surface normals.
8. Remember that light positions automatically get transformed by the GL_MODELVIEW transformation as it exists at the moment you specify the light position. You can't prevent this from happening. You can only control the GL_MODELVIEW matrix to be what you need it to be at the time.
9. Enable the following keys:

'0'	Toggle light #0 off/on
'1'	Toggle light #1 off/on
'2'	Toggle light #2 off/on
'f'	Freese/un-freeze the animation

10. Even if you weren't required to turn lights off and on, you would want to anyway. This *really* helps debug lighting programs.
11. You can do anything with the attenuation that you'd like.
12. Feel free to use the "Shortcut" functions from the Lighting notes.
13. Even though you have lots of flexibility about how to create the scene, you must make it obvious to us that it is handling the lighting correctly!

Implementing the Keyboard Keys

The 'f' key is implemented by turning the Idle Function on and off, like in Project #3:

```
case 'f':
case 'F':
        Frozen = ! Frozen;
        if( Frozen )
                glutIdleFunc( NULL );
        else
                glutIdleFunc( Animate );
        break;
```

The number keys are implemented something like this:

```
case '0':
        Light0On = ! Light0On;
        break;

case '1':
        Light1On = ! Light1On;
        break;
```

```
. . .


if( Light0On )
	glEnable( GL_LIGHT0 );
else
	glDisable( GL_LIGHT0 );

if( Light1On )
	glEnable( GL_LIGHT1 );
else
	glDisable( GL_LIGHT1 );
```

Timing Your Scene Animation

Deliberately time the animation like we've seen before. Here is a good way to do that. Set a constant called something like MS_PER_CYCLE that specifies the number of milliseconds per animation cycle. Then, in your Idle Function, query the number of milliseconds since your program started and turn that into a floating point number between 0. and 1. that indicates how far through the animation cycle you are. So, in Animate, you might say:

```
int ms = glutGet( GLUT_ELAPSED_TIME );
ms %= MS_PER_CYCLE;
Time = (float)ms / (float)MS_PER_CYCLE;			// [0.,1.)
```

and then in Display, you might use that 0.-1. number something like this:

```
 glRotatef( 360.*Time, 0., 1., 0. );
```

Using Objects You Get From Elsewhere

If you want to bring in another 3D object to work with (and there are a lot of them on the web), look for something in a **.obj** format.

If you want to load a .obj file as part of one of your projects, incorporate the file loadobjfile.cpp into your own code. Use this by placing the .obj object into a display list:

```
GLuint DL;
. . .
DL = glGenLists( 1 );
glNewList( DL, GL_COMPILE );
LoadObjFile( "spaceship.obj" );
glEndList( );
. . .
glCallList( DL );
```

Warning! Not all obj files have normals and textures
Take a look at the obj file (it is ascii-editable).
If you see lines of text beginning with **vn**, it has normals.
If you see lines beginning with **vt**, it has texture coordinates.

Lighting is *Hard*!

One of the things that you are supposed to learn from this project is that lighting is difficult, especially *good* lighting. Next time you see a CG-animated movie, look to see how many people in the credits have the word "Lighting" somewhere in their title. There's a reason the list is long.

Turn-in:

Use the Teach system to turn in your:

1. .cpp file
2. A one-page PDF with a title, your name, your email address, a nice screen shot from your program, and the link to the Kaltura video demonstrating that your project does what the requirements ask for. Narrate your video so that you can tell us what it is doing.
   Be sure that your video is flagged as *unlisted*.

Bonus Days:

Each of you has been granted five total Bonus Days, which are no-questions-asked one-day project extensions, but no more than **2** Bonus Days may be applied to any one project.

+5 Points Extra Credit

Place a texture on at least one of the objects so that it looks both textured and lighted (GL_MODULATE). Warning: among the GLUT solids, only the sphere and teapot have texture coordinates built into them. Same comment as before about **vn** and **vt** in .obj files.

Warning!

Don't spin a smooth-shaded lighted sphere about an axis through its center! You can't tell that it is spinning. Same goes for any other 360-degree smooth objects like torii and cones.

Grading:

Note: you don't get credit for these things by just having done them. You get credit for convincing us that their lighting behavior is correct. That is, code that looks correct will only get credit if you have made a scene whose video shows the correct visual lighting behavior.

Item	Points
Lights: 2 stationary, 1 moving	15
Lights: 1 point, 1 spotlight	15
Lights: 1 white, 1 colored	15
Objects: 1 stationary, 1 moving	15
Objects: 1 shiny, 1 dull	15
Objects: 1 GL_FLAT, 1 GL_SMOOTH	15
Small spheres	10
Extra Credit	5
Potential Total	105