

## Simulating rigid object animation using lighting effects in OpenGL

- Animation is achieved by rotating the position of light on the object that creates an illusion of the object itself rotating with the light.
- Material color property of the object is also allowed to change when the light position is varied.
- This is also done in OpenGL which provides more flexibility to do this work.

## Coloring in OpenGL

- OpenGL supports two color models: RGB or RGBA mode and color-index mode.
- In RGB mode, each color is a triplet of red, green, and blue values and its range should be in [0, 1]. The eye blends these primary colors, forming the color that we see. We use a fourth color component, A (or alpha) which represents an opacity factor.
- There are two types of shading in OpenGL. That is, a line or a filled polygon primitive can be drawn with a single color (flat shading) or with many different colors (smooth shading, also called as Gouraud shading).
- We can specify the desired shading technique in OpenGL with the help of the command `glShadeModel()` by passing the parameters such as `GL_FLAT` or `GL_SMOOTH` for flat and smooth shading respectively.
- OpenGL does not have any routines to load values into the color-lookup table. Window systems typically already have such operations. GLUT has the routine `glutSetColor()` to call the window system specific commands.

## Material Properties

- An object's material properties determine how it reflects light and therefore of what material it seems to be made. Because the interaction between an object's material surface and incident light is complex, specifying material properties so that an object has a certain desired appearance in an art.
- We can specify a material's ambient, diffuse and specular colors and how shiny it is.
- Material properties in OpenGL match up directly with the supported light sources and with the Phong reflection model.
- We can also specify different material properties for the front and back faces of the surface. All the reflection parameters are specified through the functions

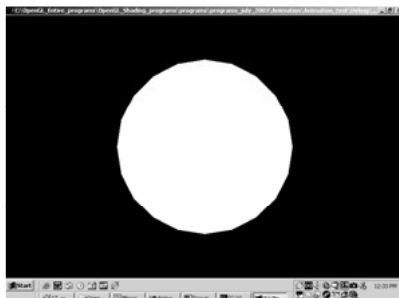
```
glMaterialf(face, pname, pointer_to_array);  
glMaterialf(face, pname, values);
```

## Lighting in OpenGL(1/6)

- OpenGL computes the color of each pixel in a final, displayed scene that's held in the framebuffer, part of this computation depends on what lighting is used in the scene and on how objects in the scene reflect or absorb that light.
- In an environment where there is no light at all, for example a windowless room, with the lights turned off, we can see nothing.
- The rendering of a three-dimensional computer graphics scene is similar. In order to "see" that is to render a scene or any objects within that scene, we must define lights. In fact, most objects don't even look three-dimensional until they are lit.

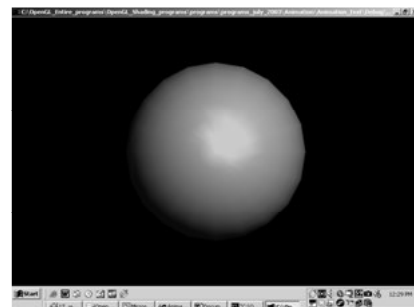
## Lighting in OpenGL(2/6)

3D Sphere without Light



## Lighting in OpenGL(3/6)

3D Sphere with Cyan Light



## Lighting in OpenGL(4/6)

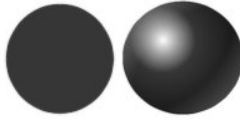
•The OpenGL lighting model considers the lighting to be divided into four independent components: ambient, diffuse, specular and emissive. All the four components are computed independently and then added together.

•OpenGL includes positional, directional, spotlight and global ambient light. We must enable or disable the light after the properties of each light is defined.

•Lighting calculations must be enabled, and each light source must be enabled individually. We can include upto eight different light sources of various colors in our scene. For a single source, we could use

```
glEnable(GL_LIGHTING);  
glEnable(GL_LIGHT0);
```

Shape 2-D      Form 3-D



## Lighting in OpenGL(5/6)

• OpenGL allows both local and global lighting. The OpenGL function used to invoke local lighting is

```
glLightfv(source, parameter, pointer_to_array);  
glLightf(source, parameter, value);
```

• To disable the light from the scene, we can use the command

```
glDisable(GL_LIGHTING);
```

• To specify the RGBA intensity of global ambient light, we can use the command

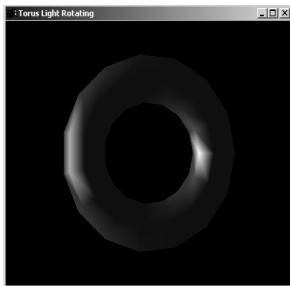
```
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,  
pointer_to_array);
```

## Lighting in OpenGL(6/6)

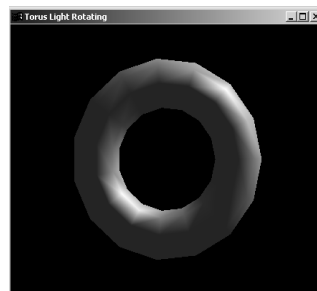
• The OpenGL lighting calculation is different for the two modes, and in fact the lighting capabilities are more limited in color-index mode. Thus RGBA is the preferred mode. We can also control a light's position and direction in three different manner such as

- A light position that remains fixed
- A light that moves around a stationary object
- A light that moves along with the viewpoint

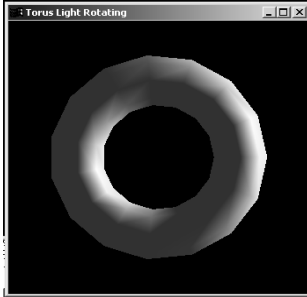
Snapshot-1



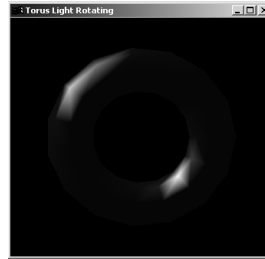
Snapshot-2



### Snapshot-3



### Snapshot-4



LightRotate.c

**LightRotate.c**

### Closing Thoughts

- An attempt is made to animate objects using regular deformation. In which dynamic motion is combined with the deformation process.
- An attempt is made to perform lighting, coloring and how to change material properties in OpenGL .
- With the help of these three effects we achieved animation by varying the position of the light in the object and changing the materials color property.
- This resembles that object is also rotating with the light, but the object remains stationary only the light is moving.
- Thus we have achieved the illusion of the object rotation with the help of light rotation and animation of object with regular deformation technique.

Q & A  
Thank You !