3D Environment Modeling

- This is really a subset of overall 3D object modeling, but is sufficiently distinct to be discussed on its own.

- So far, we’ve discussed the 3D object model in terms of representing the geometry of an object: vertices, edges, faces, shapes, and how they combine.

- If you recall, a component of that model does not have to do with geometry — instead, it deals with colors, textures, and other non-geometric elements.
Rendering in OpenGL

• So far you’ve only been using glColor*()
  – “Absolute” color: independent of lighting
  – In reality, perceived color is highly dependent on the lighting environment
    • red object under white light looks red
    • cyan object under green light looks green
    • yellow object under red light looks — gasp! — red
    • blue object under blue light looks blue
    • blue object under cyan light looks blue
    • red object under blue light looks black
    • etc.

• Welcome to glMaterial*() and glLight*()
  – Based on OpenGL’s model for light
  – Think of glMaterial*() as “professional” glColor*()

The OpenGL Light Model

• Is based on, but not the same as the real world’s light model
  – Why not?

• Light is broken up into three components:
  – Ambient: Light that is so scattered as to appear to be coming from all
directions and going in all directions
  – Diffuse: Light coming from a specific direction
  – Specular: Light that is reflected back in a specific direction; specular light
corresponds to the notion of “shininess”

• Lighting (and therefore shading) in OpenGL is based on the
interaction of light sources on materials, according to
combinations of their respective ambient, diffuse, and specular
components
Setting Up a Lit Scene

- Define your pure model so that it captures the data that influences the 3D environment
  - light sources: colors, positions, directions
  - material settings: colors, other properties
- Prepare your geometric model to interact properly with lighting
  - normal vectors using `glNormal*()`
    - the GLUT quickie shapes do this for you already; if you build your own objects, you’ll need to do this yourself
  - ambient, diffuse, and specular material properties using `glMaterial*()`
- “Translate” the pure model into OpenGL’s light model using:
  - `glEnable(GL_LIGHTING)` to activate lighting in the first place
  - Turn on individual light sources — `glEnable(GL_LIGHT0)`
  - Set up the light sources — `glLight*()`

Material Details

- Anatomy of `glMaterial*()` — check the red book for details

```c
void glMaterialf(GLenum face, GLenum pname, GLfloat param);
void glMaterialfv(GLenum face, GLenum pname, const GLfloat *params);
void glMateriali(GLenum face, GLenum pname, GLint param);
void glMaterialiv(GLenum face, GLenum pname, const GLint *params);
```

- Which side of the current face?
  - GL_FRONT
  - GL_BACK
  - GL_FRONT_AND_BACK

- Which material property?
  - GL_AMBIENT
  - GL_DIFFUSE
  - GL_AMBIENT_AND_DIFFUSE
  - GL_SPECULAR
  - GL_SHININESS
  - GL_EMISSION

- Which property? RGBA most of the time; for GL_SHININESS, a single scalar value from 0 to 128 (128 being “shiniest”)
- Set the property to what value?
Light Details

• Anatomy of glLight*()

```c
void glLightf (GLenum light, GLenum pname, GLfloat param);
void glLightfv (GLenum light, GLenum pname, const GLfloat *params);
void glLighti (GLenum light, GLenum pname, GLint param);
void glLightiv (GLenum light, GLenum pname, const GLint *params);
```

Which light?
GL_LIGHT0 to GL_LIGHT7
...some implementations of OpenGL may have more

Which light property?
GL_AMBIENT
GL_DIFFUSE
GL_SPECULAR
GL_POSITION
GL_*.ATTENUATION
GL_SPOT_ *

Set the property to what value?
RGBA most of the time; for GL_POSITION, an (x, y, z, w) tuple, w = 0 implies direction instead of position; attenuation are scalars, and spotlight values are scalars except for GL_SPOT_DIRECTION

Even More Details

• While glLight*() and glMaterial*() specify what OpenGL processes to do lighting/shading calculations, there are also configurable options on how to do these calculations

• This tweaking can be done with glLightModel*() — check the red book for details

• But in general, the defaults for the light model will suffice