**✅ 1. Modeling Virtual Appearance in VR**

To make virtual objects look realistic, we work on:

* **Material** (what it’s made of)
* **Lighting** (how light falls on it)
* **Shading** (how shadows and light behave)
* **Textures** (surface details like smooth, rough, etc.)

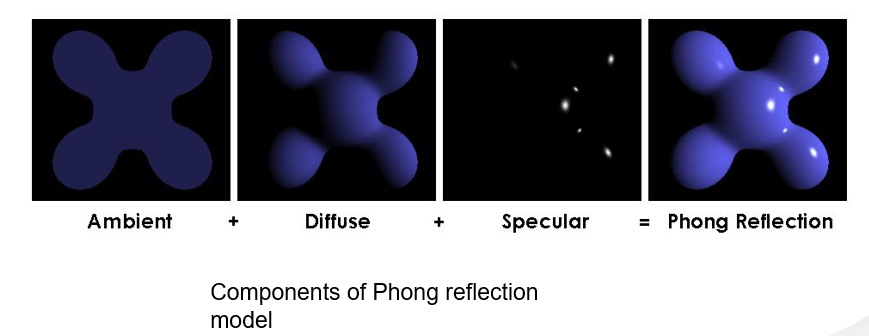
**✅ 2. Phong Reflection Model**

This model explains how light reflects from objects. It has 3 parts:

**🔸 Ambient Reflection**

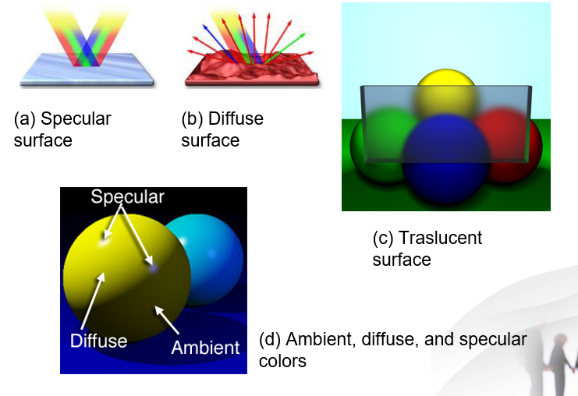
* Light that exists in the environment (like general background light).
* Material reflects some amount of it (controlled by kar, kag, kab for red, green, blue).

**🔸 Diffuse Reflection**

* Happens on dull surfaces.
* Light spreads out evenly in all directions.
* Depends on how strong the light is and how close it is.

**🔸 Specular Reflection**

* Shiny reflections (like glass or water).
* Depends on viewing angle and light angle.
* Has a **shininess factor**.

**✅ 3. Types of Surfaces**

|  |  |
| --- | --- |
| Type | Behavior |
| Specular | Smooth and shiny, reflects clearly (e.g. mirror) |
| Diffuse | Rough and dull, light scatters (e.g. wall) |
| Translucent | Let light pass through partially (e.g. glass, water) |

**Ambient Color** = How the object looks in shadow (when no light directly falls on it).

**✅ 4. Material Modeling in OpenGL**

In OpenGL, materials are described using 3 arrays:

GLfloat ambient[] = {r, g, b, a};

GLfloat diffuse[] = {r, g, b, a};

GLfloat specular[] = {r, g, b, a};

You use:

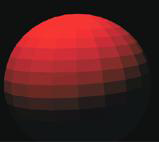
//Face, Property, Value

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);

to apply material to object surfaces.  
Use GL\_SHININESS to make it more shiny.

**✅ 5. Light Sources in OpenGL**

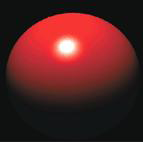
|  |  |
| --- | --- |
| Light Type | Description |
| Ambient | Light everywhere equally (e.g. bulb with cover) |
| Point | Light from one point in all directions (e.g. lamp) |
| Spotlight | Light in cone shape, focused in the center |
| Distant | Light rays are parallel (like sunlight) |
| Colored | Lights can be colored (using RGB values) |

**✅ 6. Shading Techniques**

**🔸 Flat Shading**

* Each polygon (flat face) has one color.
* Simple but looks less realistic.

**🔸 Smooth Shading**

* Colors change gradually.
* Uses interpolation between points (vertices).
* More realistic.

Use in OpenGL:

glShadeModel(GL\_FLAT); // for flat

glShadeModel(GL\_SMOOTH); // for smooth

**✅ 7. C++ Code: Light and Material Setup**

The init() function in OpenGL:

* Enables light, depth testing.
* Sets light position, type, and color.
* Sets material reflection: ambient, diffuse, specular.
* Sets GL\_SHININESS to control how shiny the surface is.

Example Object:

glutSolidTeapot(1.0);

**✅ 8. Street Light Example**

**Use:** Ambient Light  
Because street lights with rough covers spread light evenly.

C++ function to use:

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);

**✅ 9. Another Example Recap**

In the second code example:

* Light is **positional** (i.e., a point light)
* Materials are defined for how the object looks under light
* Perspective and background color are set

**🔚 Summary**

|  |  |
| --- | --- |
| Topic | Simple Meaning |
| Phong Model | Explains how light reflects (ambient, diffuse, specular) |
| Material Arrays | Tell OpenGL how to color the object |
| Light Types | Different ways light behaves in scene |
| Shading | Flat = simple, Smooth = realistic |
| OpenGL Functions | Used to enable lighting and apply material |