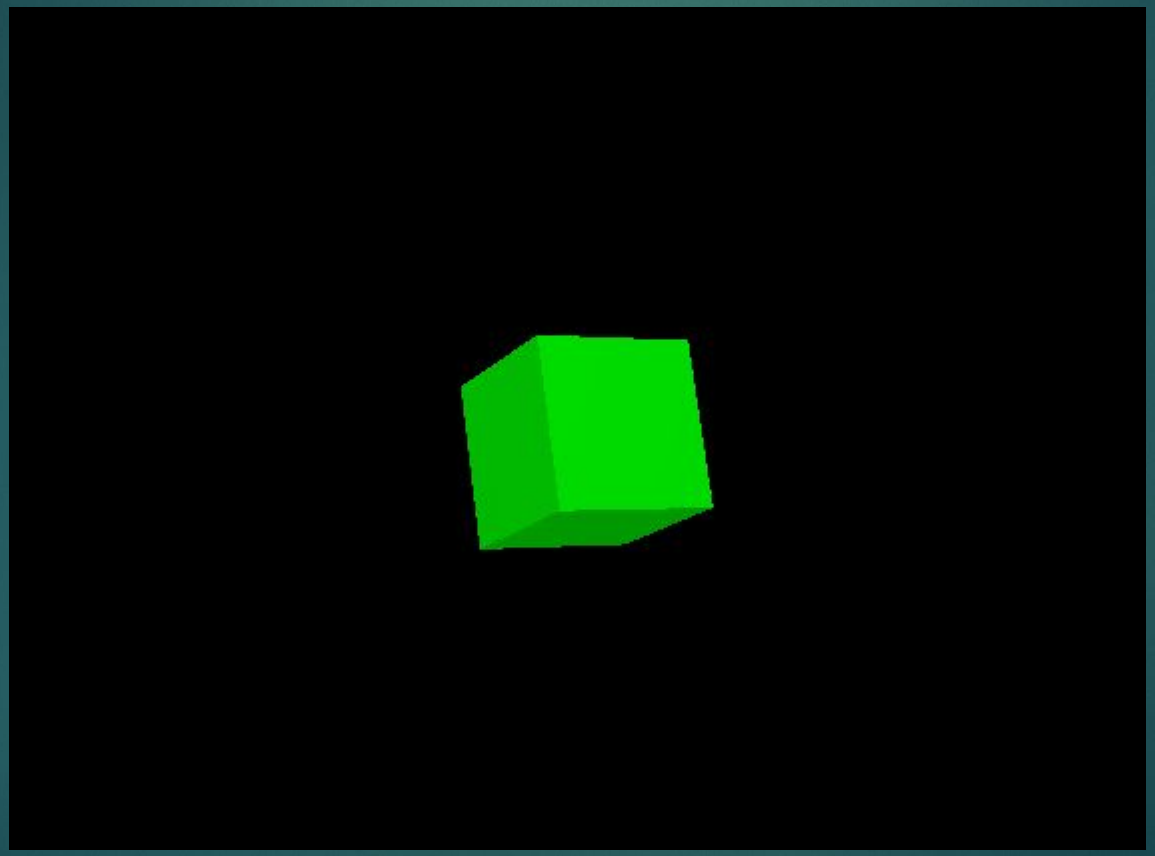
OpenGl Course 2020

Lighting

Lighting



Lighting

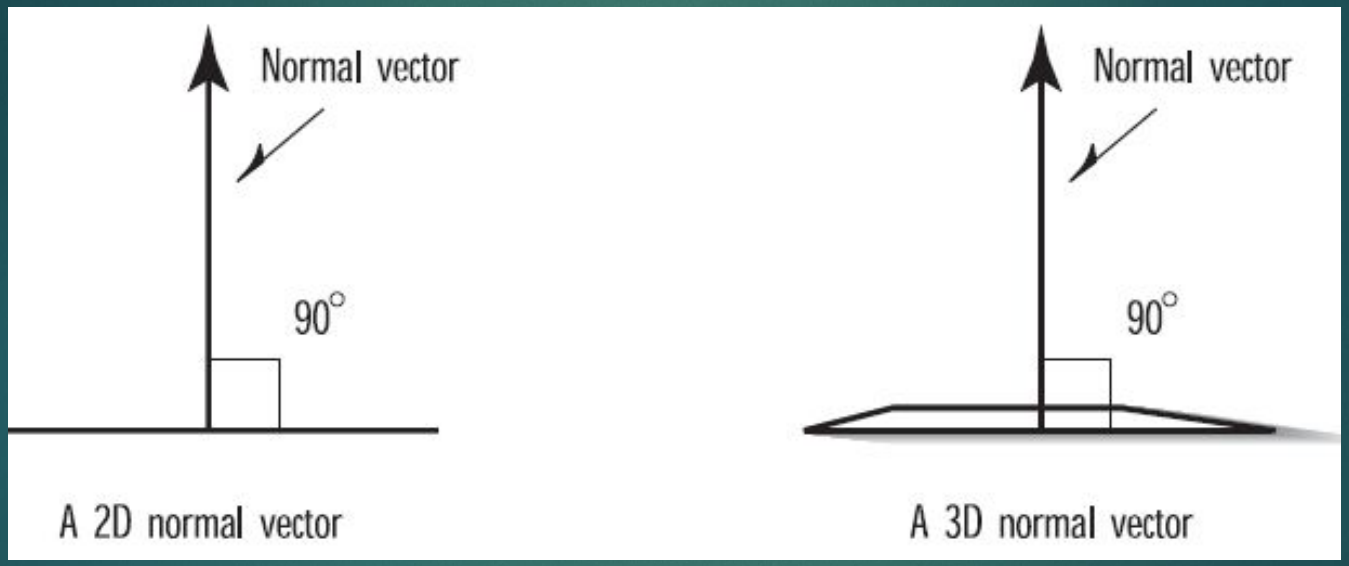
- Lighting is an important effect that supplies the realistic look of the scene. So we must realize that it's an extra effect !
- Which means that it's not necessary to add a light to the scene. However, without lighting your scene would look like cartoons and so far from the reality.
- The final color of the body that we see on the screen is not just about color of the body but also the lighting body.
- In OpenGL there are 8 lights, we can change their position or their properties (color, ambient, …)

Lighting
- Normal

Normal

Lighting - Normal

- Normal vector is a line pointed in a direction



- Normal is very useful in lighting. We must specify the normal of the polygon we're drawing in order to get a nicer result in lighting.

Lighting - Normal

- OpenGL statement which specifies the normal is:

```
glNormal3f(X, Y, Z);
```

- For example:

```
glBegin(GL_TRIANGLES);
    glNormal3f(0.0f, 0.0f, 1.0f);
    glVertex3f(-1.0f, 0.0f, 0.0f);
    glVertex3f(1.0f, 0.0f, 0.0f);
    glVertex3f(0.0f,1.0f, 0.0f);
glEnd();
```

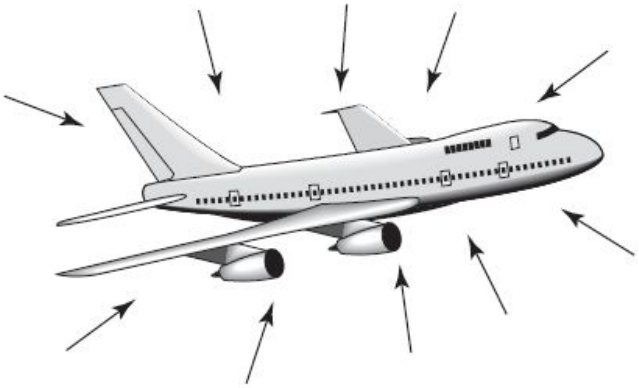
- Guess why the normal is pointing towards us ? ☺

Lighting
- Ambient

Ambient

Lighting - Ambient

- Ambient light doesn't come from any particular direction. It has an original source somewhere, but the rays of light have bounced around the room or scene and become directionless.



An object illuminated purely by ambient light.

Lighting - Diffuse

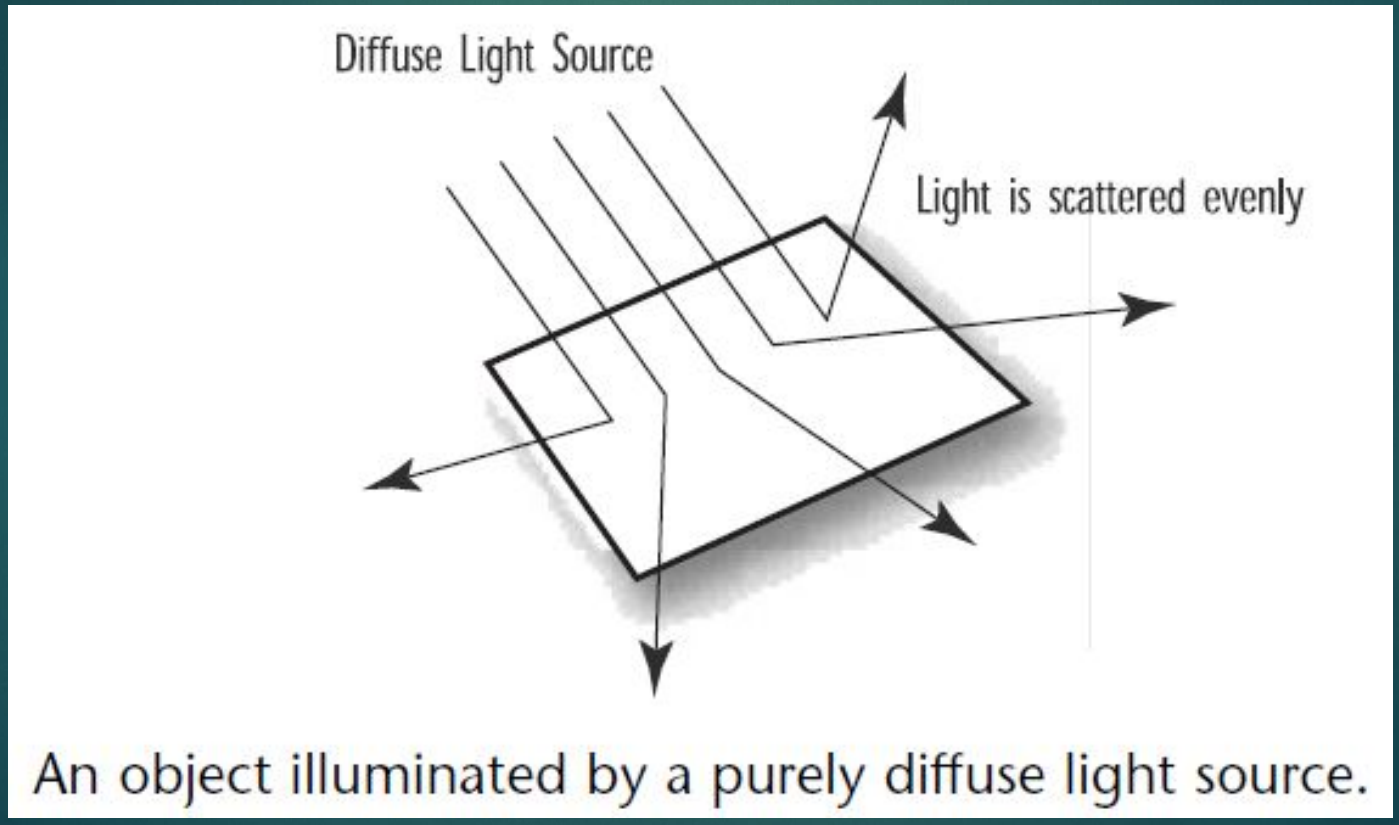
Diffuse

Lighting - Diffuse

The diffuse part of an OpenGL light is the **directional component** that appears to come from a particular direction and is reflected off a surface with an intensity proportional to the angle at which the light rays strike the surface. Thus, the object surface is brighter if the light is pointed directly at the surface than if the light grazes the surface from a greater angle. Good examples of diffuse light sources include a lamp, candle, or sunlight streaming in a side window at noon. Essentially, it is the diffuse component of a light source that produces the shading (or change in color) across a lit object's surface.

Lighting - Diffuse

- In Figure, the object is illuminated by a



An object illuminated by a purely diffuse light source.

Lighting
- Specular

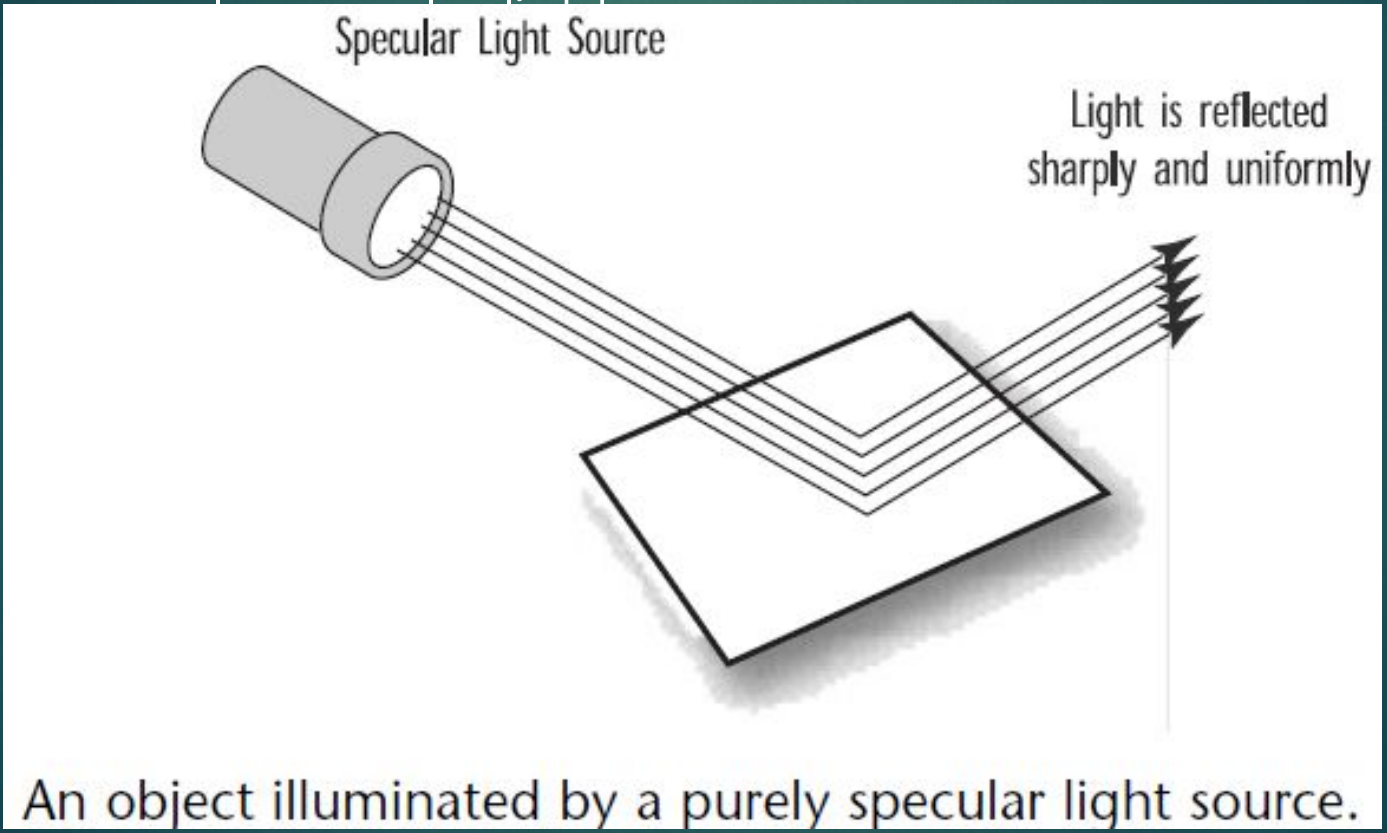
Specular

Lighting - Specular

- Like diffuse light, specular light is a **highly directional property**, but it interacts more sharply with the surface and in a particular direction. A highly specular light (really a material property in the real world) tends to cause a bright spot on the surface it shines on, which is called the *specular highlight*. Because of its highly directional nature, it is even possible that depending on a viewer's position, the specular highlight may not even be visible. A spotlight and the sun are good examples of sources that produce strong specular highlights.

Lighting - Specular

- Figure below shows an object illuminated by a



An object illuminated by a purely specular light source.

Lighting
- Putting It All Together

Putting It All Together

Lighting - Putting It All Together

No single light source is composed entirely of any of the three types of light just described. Rather, it is made up of varying intensities of each. For example, a red laser beam in a lab is composed of almost a pure-red specular component producing a very bright spot where it strikes any object. However, smoke or dust particles scatter the beam all over the room, giving it a very small ambient component. This would produce a slight red hue on other objects in the room. If the beam strikes a surface at a glancing blow, a very small diffuse shading component may be seen across the surface it illuminates (although in this case it would be largely overpowered by the specular highlight).

Lighting - Putting It All Together

Thus, a light source in a scene is said to be composed of

three lighting components: ambient, diffuse, and specular. Just like the components of a color, each lighting component is defined with an RGBA value that describes the relative intensities of red, green, and blue light that make up that component (for the purposes of light color, the alpha value is ignored). For example, our red laser light might be described by the component values in Table:

	Red	Green	Blue	Alpha
Specular	0.99	0.0	0.0	1.0
Diffuse	0.10	0.0	0.0	1.0
Ambient	0.05	0.0	0.0	1.0

Material

Material

Light is only part of the equation. In the real world

objects do have a color of their own. Generally, the color of an object as defined by its reflected wavelengths of light. A blue ball reflects mostly blue photons and absorbs most others. This assumes that the light shining on the ball has blue photons in it to be reflected and detected by the observer. Generally, most scenes in the real world are illuminated by a white light containing an even mixture of all the colors. Under white light, therefore, most objects appear in their proper or "natural" colors. However, this is not always so; put the blue ball in a dark room with only a yellow light, and the ball appears black to the viewer because all the yellow light is absorbed and there is no blue to be reflected.

Material

When we use lighting, we do not describe polygons

as having a particular color, but rather as consisting of materials that have certain reflective properties. Instead of saying that a polygon is red, we say that the polygon is made of a material that reflects mostly red light. We are still saying that the surface is red, but now we must also specify the material's reflective properties for ambient, diffuse, and specular light sources. A material might be shiny and reflect specular light very well, while absorbing most of the ambient or diffuse light. Conversely, a flat colored object might absorb all specular light and not look shiny under any circumstances. Another property to be specified is the emission property for objects that emit their own light, such as taillights or glow-in-the-dark watches.

Adding Light To Material

Adding Light To Material

Setting lighting and material properties to achieve the desired effect takes some practice. There are no color cubes or rules of thumb to give you quick and easy answers. This is the point at which analysis gives way to art, and science yields to magic. When drawing an object, OpenGL decides which color to use for each pixel in the object. That object has reflective "colors," and the light source has "colors" of its own. How does OpenGL determine which colors to use? Understanding these principles is not difficult, but it does take some simple grade-school multiplication. (See, that teacher told you you'd need it one day!)

Adding Light To Material

Each vertex of your primitives is assigned an RGB color value based on the net effect of the ambient, diffuse, and specular illumination multiplied by the ambient, diffuse, and specular reflectance of the material properties. Because you make use of smooth shading between the vertices, the illusion of illumination is achieved.

Adding Light To Material

Calculating Ambient Light Effects:

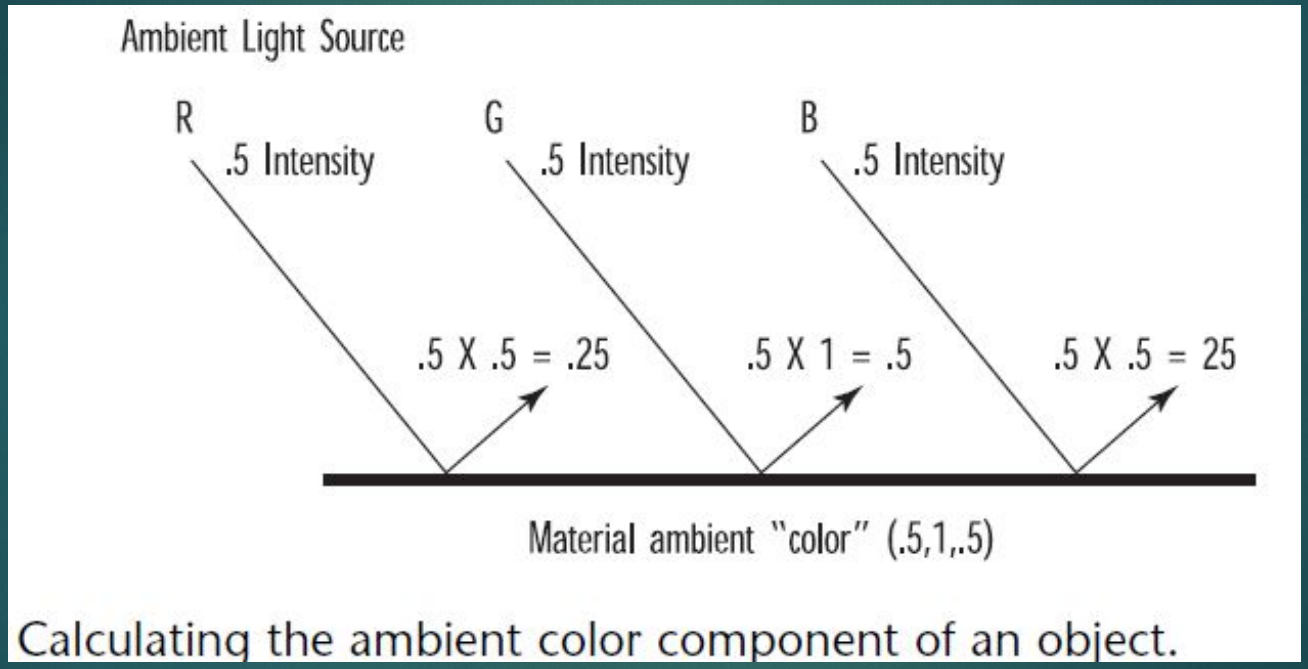
- To calculate ambient light effects, you first need to put away the notion of color and instead think only in terms of red, green, and blue intensities. For an ambient light source of half-intensity red, green, and blue components, you have an RGB value for that source of (0.5, 0.5, 0.5). If this ambient light illuminates an object with ambient reflective properties specified in RGB terms of (0.5, 1.0, 0.5), the net "color" component from the ambient light is

  (0.5 * 0.5, 0.5 * 1.0, 0.5 * 0.5) = (0.25, 0.5, 0.25)

- This is the result of multiplying each of the ambient light source terms by each of the ambient material property terms
- See figure in the next slide.

Adding Light To Material

- **Calculating Ambient Light Effects:**



Calculating the ambient color component of an object.

Adding Light To Material

Diffuse and Specular Effects:

- Calculating ambient light is as simple as it gets. Diffuse light also has RGB intensities that interact in the same way with material properties. However, diffuse light is directional, and the intensity at the surface of the object varies depending on the angle between the surface and the light source, the distance to the light source.
- the same goes for specular light sources and intensities.
- finally, all three RGB terms are added to yield a final color for the object.

Syntax

Syntax

- **Enabling the Lighting**
  - To tell OpenGL to use lighting calculations, call glEnable with the GL_LIGHTING parameter:

```
glEnable(GL_LIGHTING);
```

  - This call alone tells OpenGL to use material properties and lighting parameters in determining the color for each vertex in your scene. However, without any specified material properties or lighting parameters, your object remains dark and unlit

Syntax

- As we said before, there are 8 lights in OpenGL.
- To enable light0 just write simply:

```
glEnable(GL_LIGHT0);
```

- We can enable the others in same way.
- The others are named GL_LIGHT1 till GL_LIGHT7.

Syntax

- To change the properties of lights. We can use the following statement:

```
glLightfv (,, GLfloprolightpertyat[4] value);
```

For example:

```
glLightfv(_LIGLGHT0, GL_POSITION, LightPos);
```

- Where LightPos is a pre-defined array before changing the properties of light0:

```
GLfloat LightPos[]= {x, y, z, 1.0f};
```

- There are too many properties we can change in light like:
  - GL_POSITION , GL_AMBIENT , GL_DIFFUSE , GL_SPECULAR
  - GL_AMBIENT_AND_DIFFUSE

Syntax

- All the previous properties take a float array of length 4.
- In order to change the properties of material,

glMaterialfv (face(s), property, Glfloat[4] value);

glMaterialfv(GL_FRONT, GL_DIFFUSE, MatDif);

- All the properties of light are properties of material too, like (GL_AMBIENT , GL_DIFFUSE , GL_SPECULAR, GL_AMBIENT_AND_DIFFUSE) . But there is no position in material.

Syntax

There is another property in the material is called shininess. It can be changed using the function:

- The range of its values is [0, 128].

```
glMateriali (GL_FRONT, GL_SHININESS, 128);
```

Syntax

- Shininess of material is strongly related with specular light. For example, iron is very shiny so its shininess value is very high. In the other hand, wood is not shiny a lot. So its shininess is very low.
- After changing the properties of the material. We must enable this material on some primitive. We can do that using the function:

```
glEnable(GL_COLOR_MATERIAL);
```

Syntax

When we enable this material before drawing some

shape then this shape will be affected with this material. To stop affecting shapes with this material we can disable this material using function:

```
glDisable(GL_COLOR_MATERIAL);
```

Example

Example

Example

- In this example we will draw a cube with some material

and put a light in a particular position. Then we'll change the properties of both light and material to notice the differences.

- Firstly, we'll declare the properties values before the InitGL function:

```
GLfloat LightPos[]    = {0.0f,5.0f,4.0f,1.0f};
GLfloat LightAmb[]  = {1.0f,1.0f,1.0f,1.0f};
GLfloat LightDiff[]   = {0.6f,0.6f,0.6f,1.0f};
GLfloat LightSpec[] = {0.2f,0.2f,0.2f,1.0f};
GLfloat MatAmb[]   = {1.0f,1.0f,1.0f,1.0f};
GLfloat MatDif[]     = {0.6f,0.6f,0.6f,1.0f};
GLfloat MatSpec[]  = {0.2f,0.2f,0.2f,1.0f};
GLfloat MatShn[]    = {128.0f};
```

Example

- Then, we will use these values in the InitGL function

to change the properties of the light and material:

```
glLightfv(GL_LIGHT1,GL_POSITION,LightPos);
glLightfv(GL_LIGHT1,GL_AMBIENT,LightAmb);
glLightfv(GL_LIGHT1,GL_DIFFUSE,LightDiff);
glLightfv(GL_LIGHT1,GL_SPECULAR,LightSpec);
glEnable(GL_LIGHT1);
glEnable(GL_LIGHTING);
glMaterialfv(GL_FRONT,GL_AMBIENT,MatAmb);
glMaterialfv(GL_FRONT,GL_DIFFUSE,MatDif);
glMaterialfv(GL_FRONT,GL_SPECULAR,MatSpec);
glMaterialfv(GL_FRONT,GL_SHININESS,MatShn);
glEnable(GL_COLOR_MATERIAL);
```

Example

Finally, we'll draw the cube in the

```
glTranslatef(0, 0, -5);
glColor3f(0,0.5,0);
glRotatef(30,0,1,0);
glRotatef(-15,1,0,0);
auxSolidCube(0.5);
```