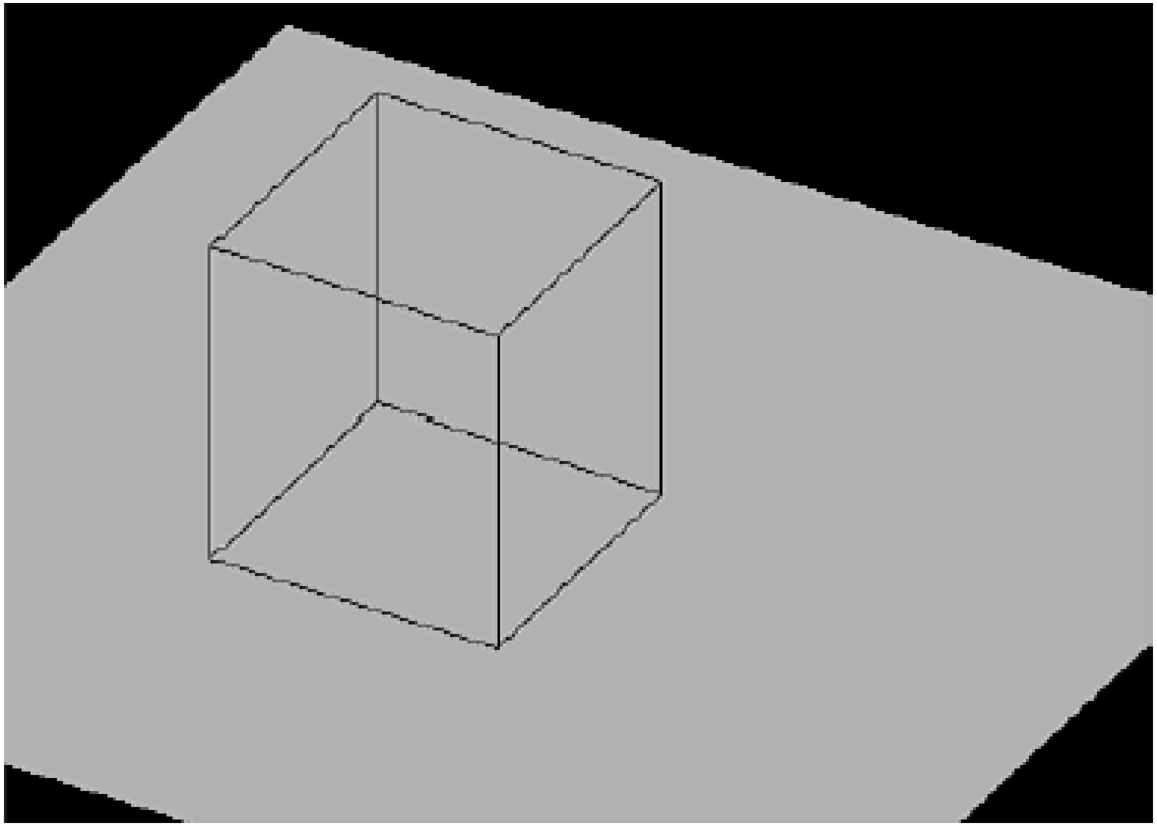
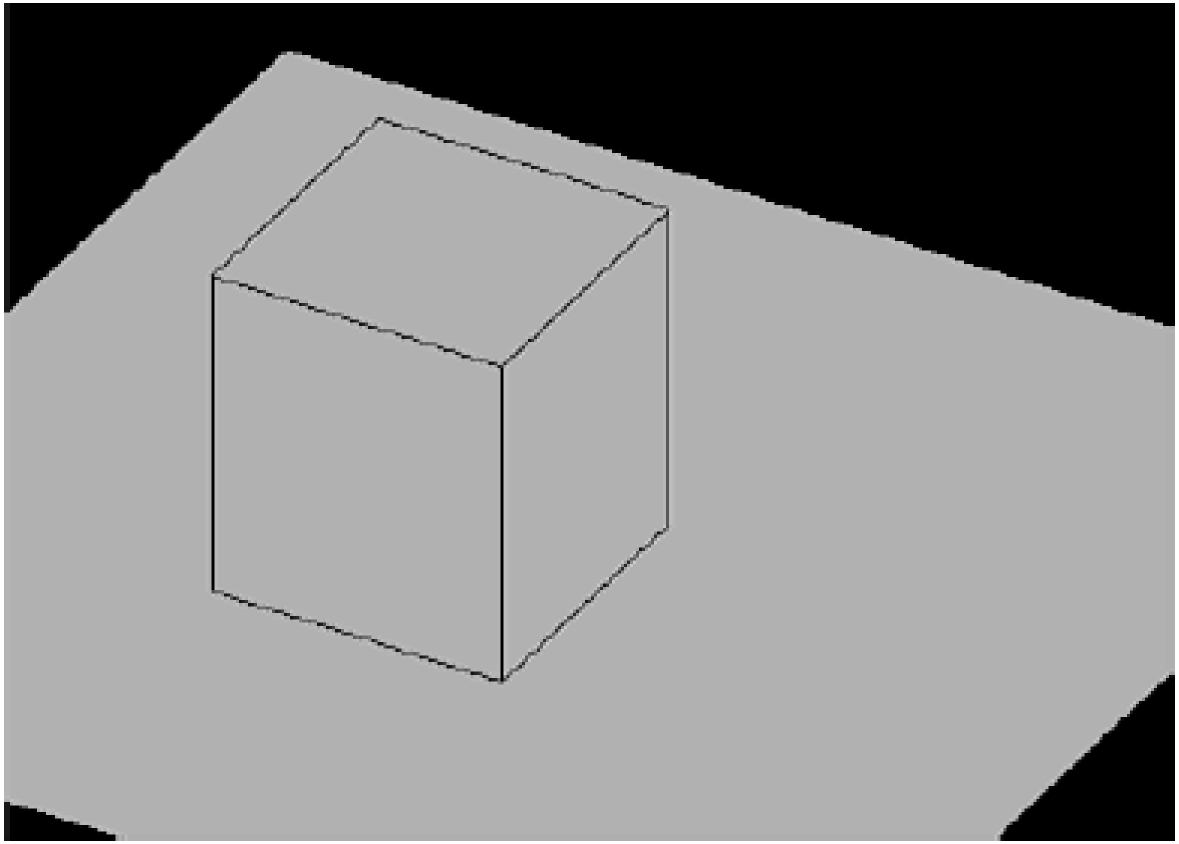
3D EFFECTS

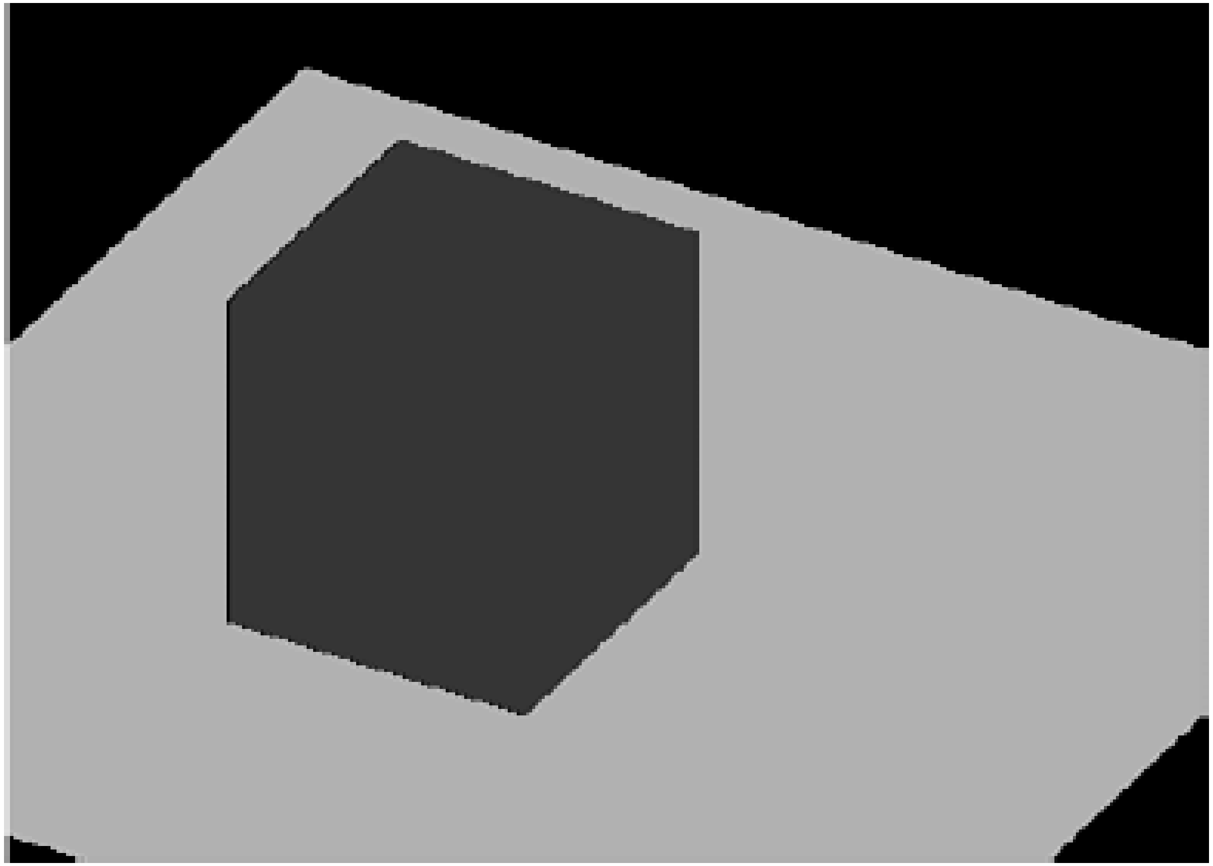
A line-drawn 3D cube



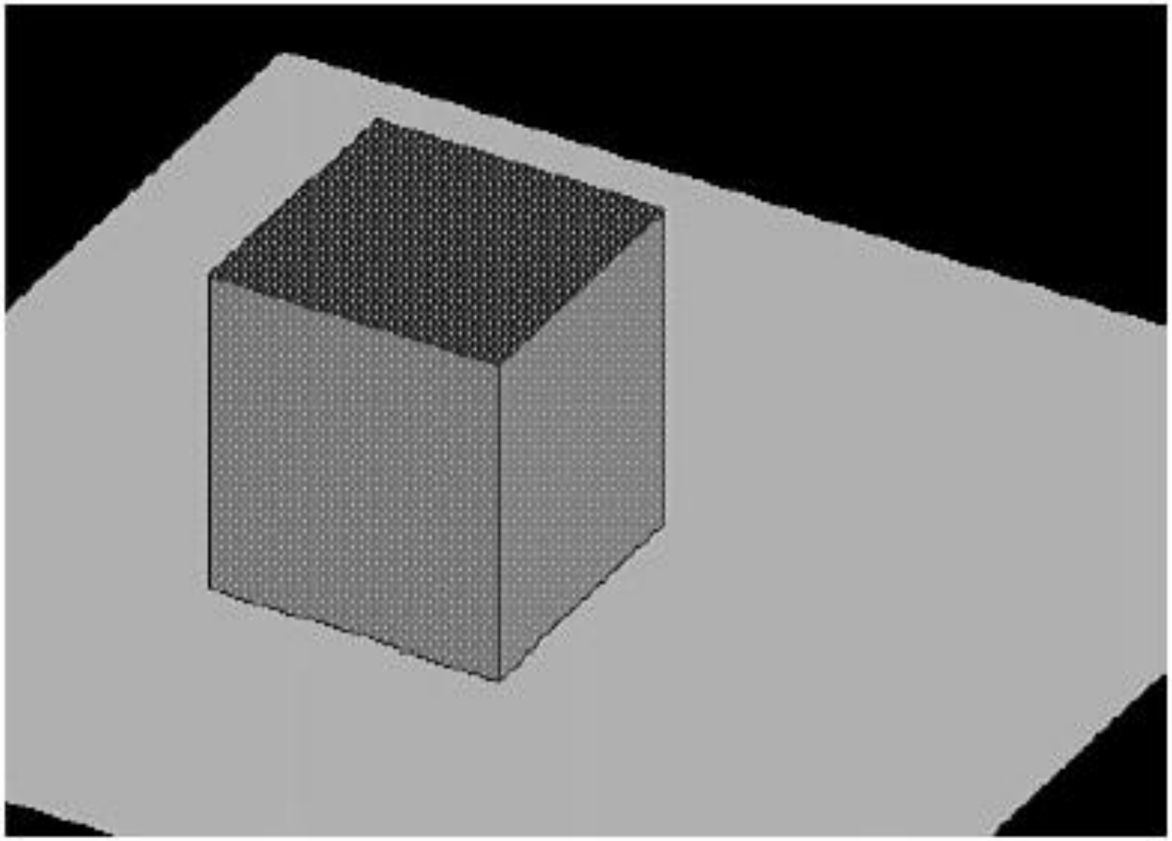
A more convincing solid cube



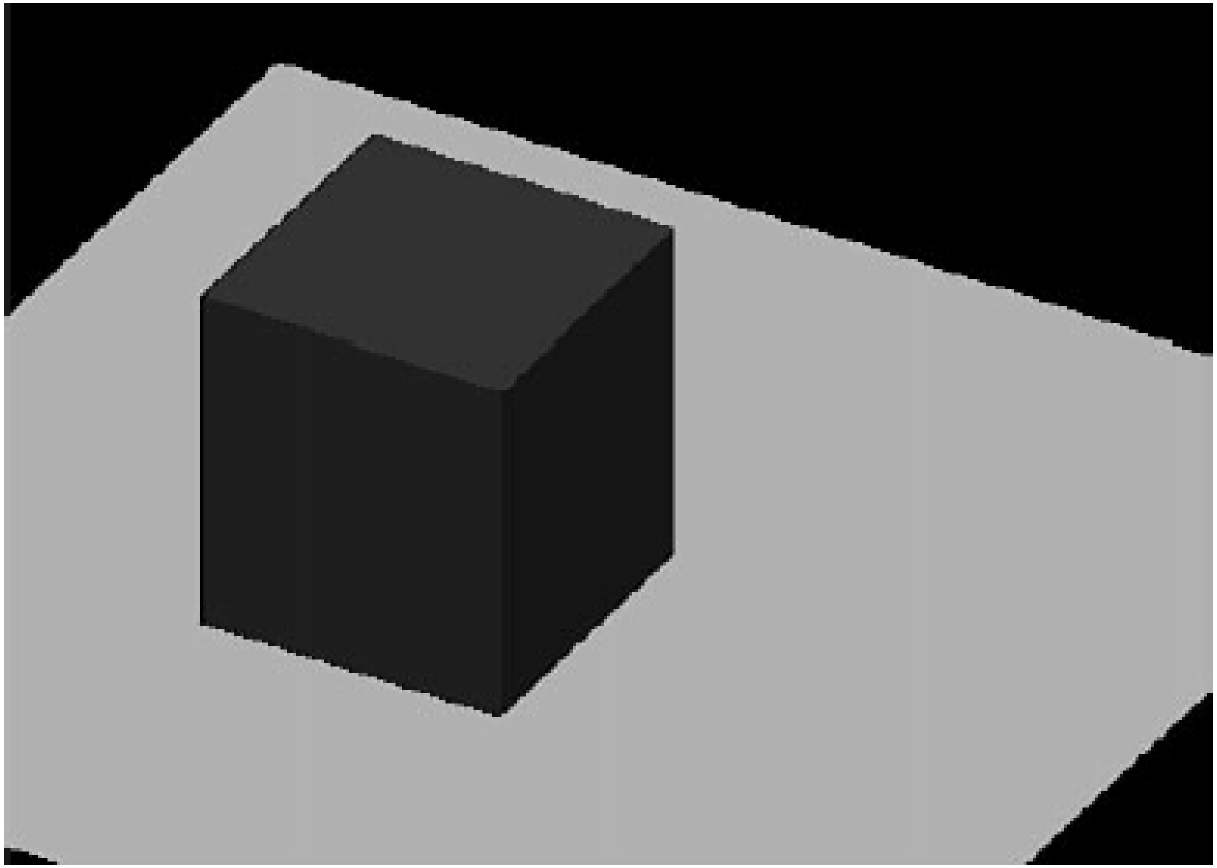
Adding color alone can create confusion



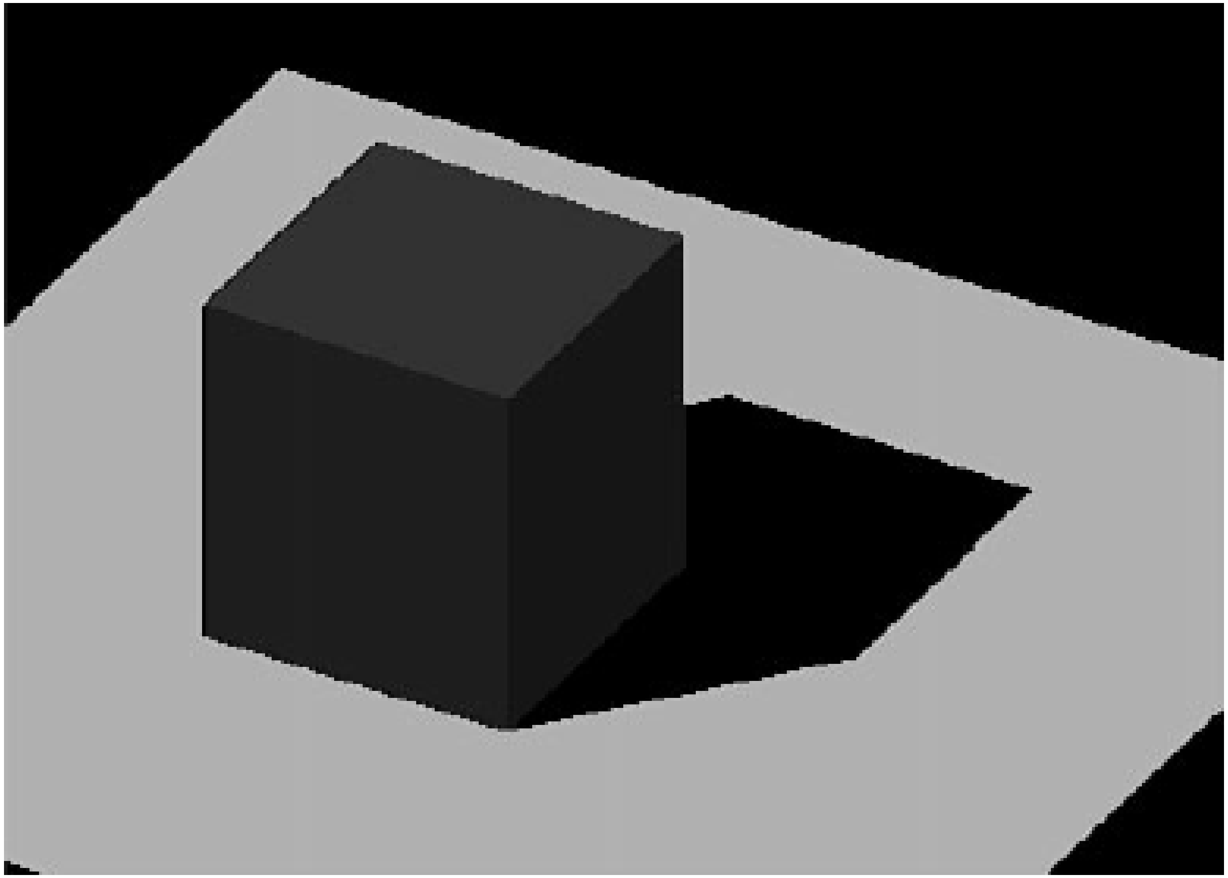
Adding different colors increases the illusion of three dimensions



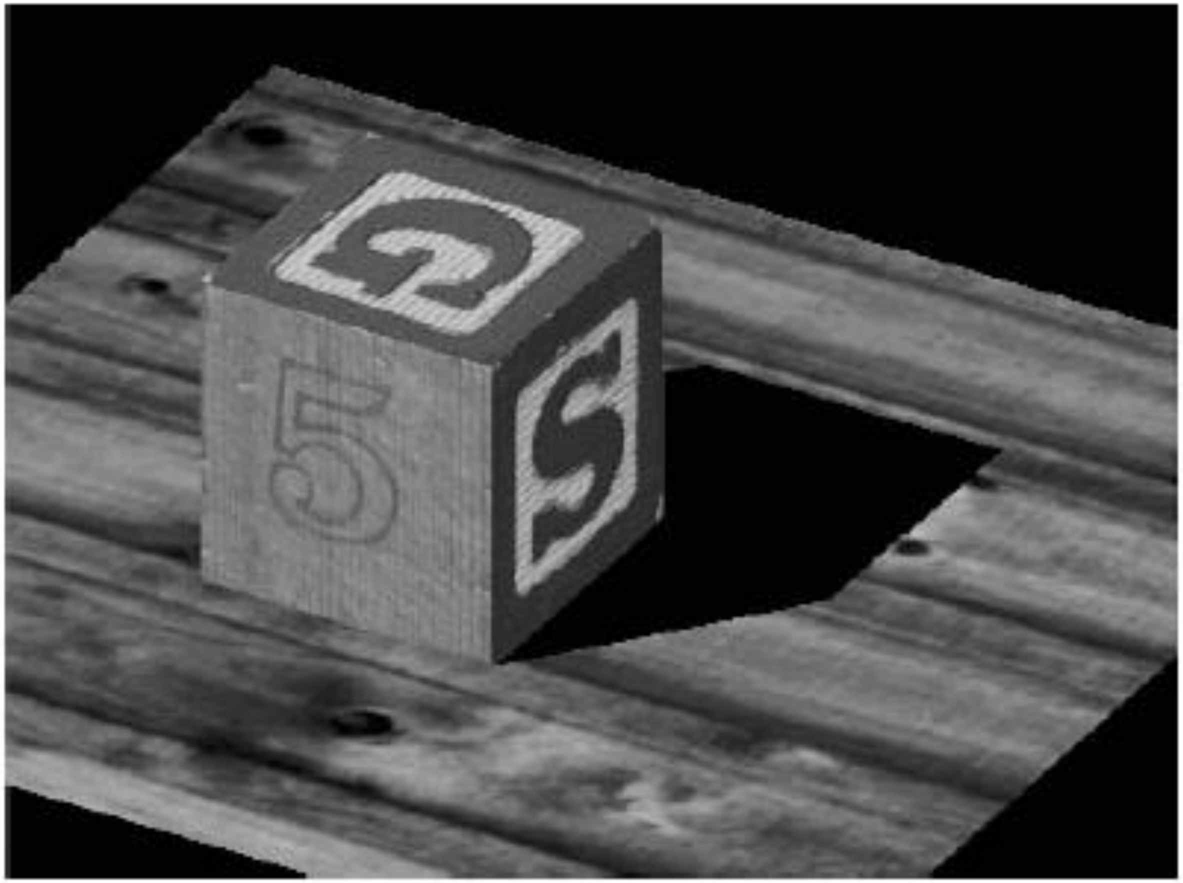
Proper shading creates the illusion of illumination



Adding a shadow to further increase realism



Texture mapping adds detail without adding additional geometry



Colors & Lighting

Lecture 11

Outline

- Colors in OpenGL
- Shading
- OpenGL lighting
- Light sources
- Materials

Setting the Color

- In RGBA mode, you specify colors by indicating the intensity of the red, green and blue components.
- There's also an optional 4th component, called alpha, which is usually used for transparency.

glColor*()

- To specify the primary color in OpenGL, we’ll use one of the many variations of glColor*():

```
void glColor{34}{bsifd ubusui}(T components);
void glColor{34}{bsifd ubusui}v(T components);
```

The primary color is used to determine vertex colors when lighting is not enabled.

glColor*()

```
//using floats
glColor3f(1.0, 1.0, 0.0); //yellow

//using unsigned bytes
glColor3ui(255, 255, 0);

//using signed bytes in an array
GLbyte yellow[ ]={127, 127, 0};
glColor3uiv(yellow);
```

Shading

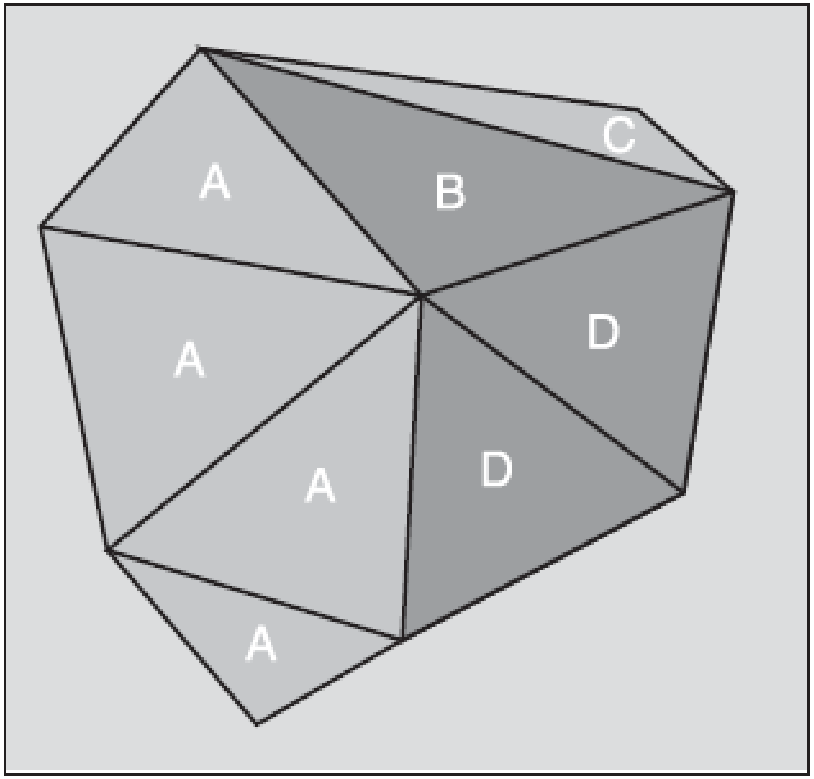
- So far, we used glColor() to set the color at each vertex.
- But how does OpenGL decide what color to use for pixels in the middle of a triangle?
  - If all three vertices are the same color, then it should be obvious that every pixel in the triangle should use that color.

Shading

- But what happens if you use a different color for each vertex of a primitive?
  - Let’s consider a line with two vertices of different colors.
- So what is the color of the line itself?
  - This answer comes from what is known as the ***shading model***.
  - There’re 2 types of shading: **flat** and **smooth**

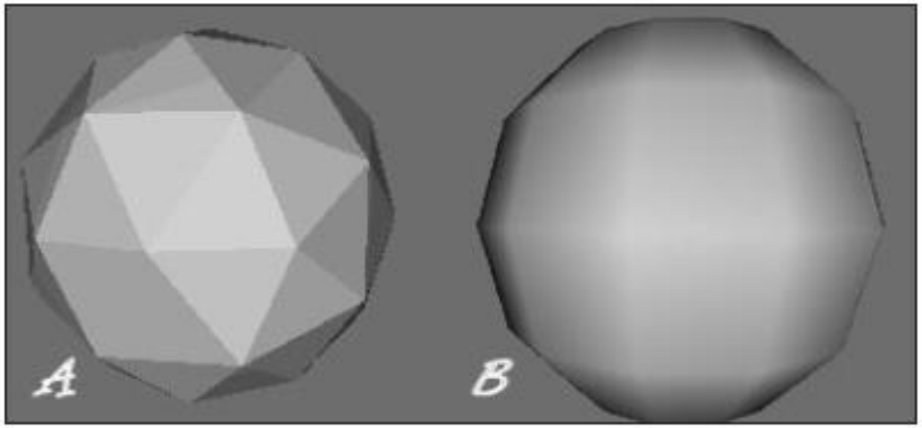
Flat Shading

- When we want to display 3D objects, we usually use some technique to apply color to the faces.
- The simplest method is **flat shading**.



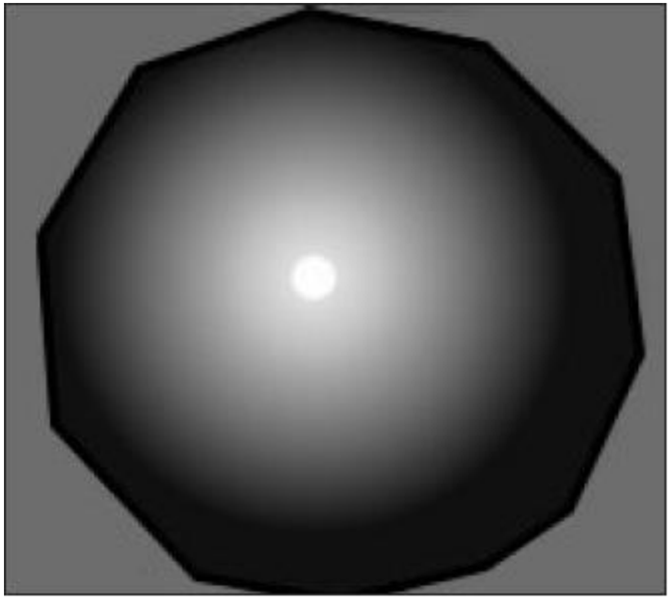
Gouraud Shading

- Gouraud shading smoothes the colors by averaging the normals of vertices of a surface.
- The normals are used to modify the color value of all the pixels in a face.
- Gouraud shading creates a much more natural appearance for the object.



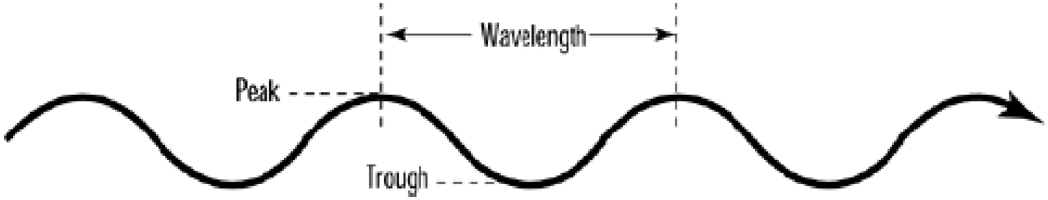
Phong Shading

- Phong shading is a much more complicated- and computation-intensive- technique for rendering a 3D object.
- Like gouroud shading, it calculates color or shade values for each pixel.
- Unlike gouraud shading, phong shading computes additional normals for each pixel between vertices and then calculates the new color values.

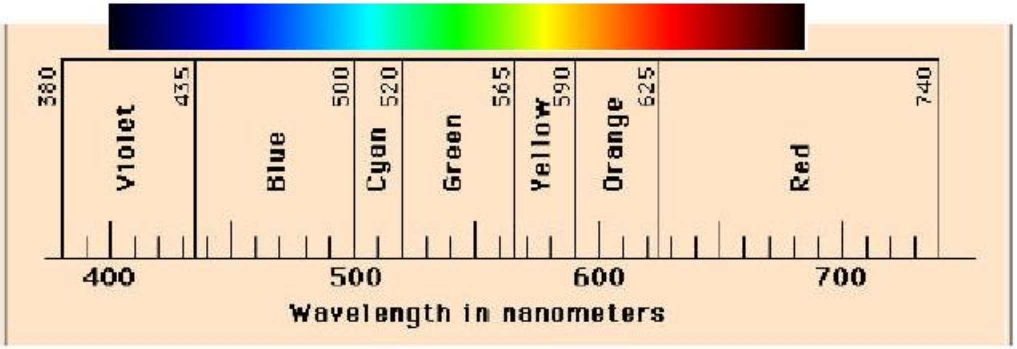


What is color?

- **Light as a Wave**
  - Color is simply a wavelength of light that is visible to the human eye.

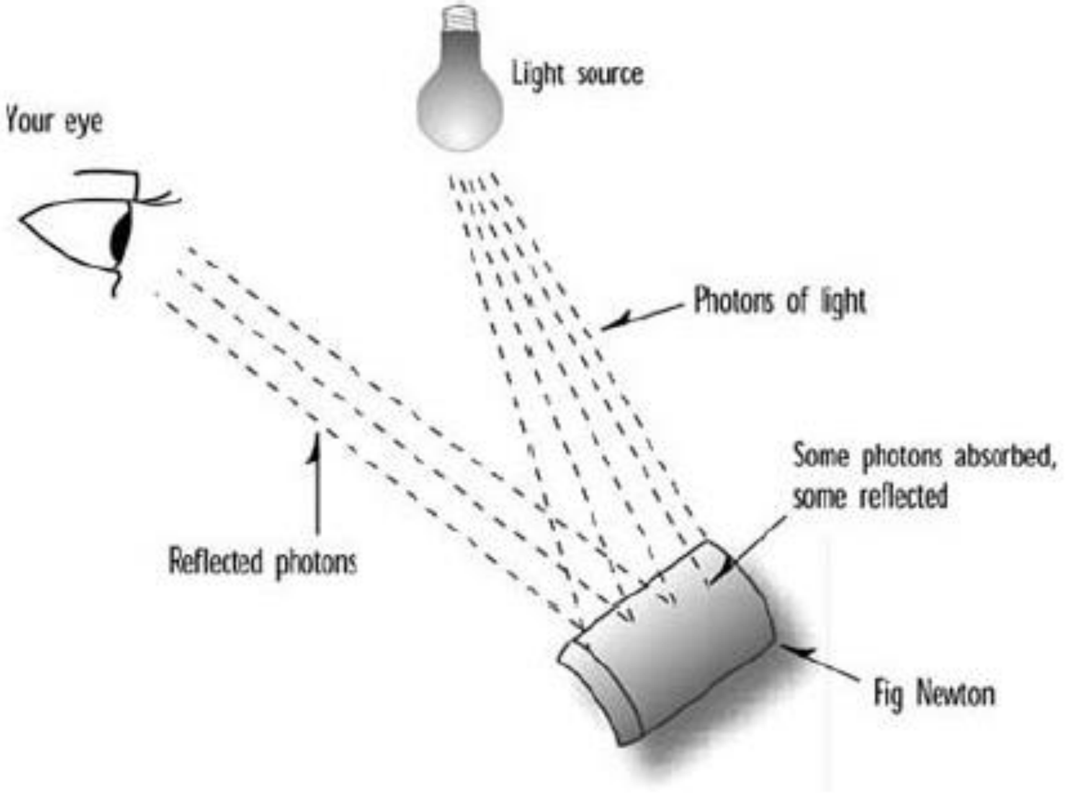


  - Visible spectrum of light



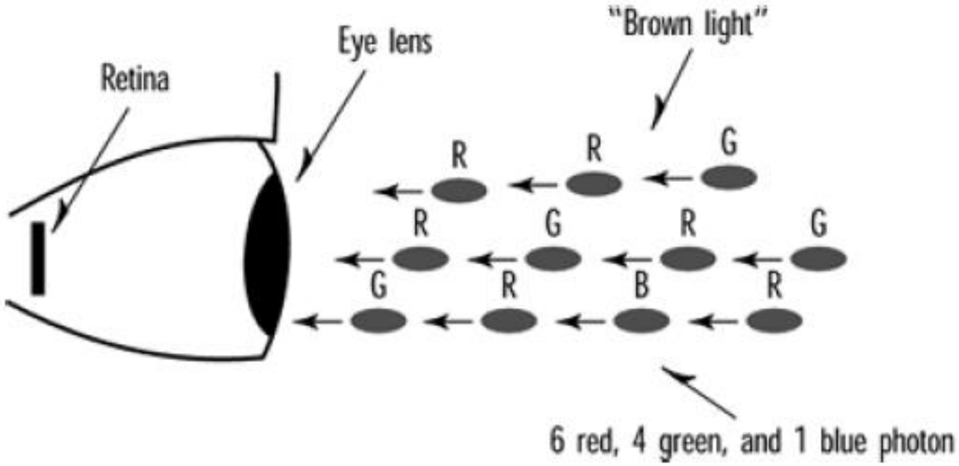
What is color?

- **Light as a Particle**
  - An object reflects some photons and absorbs others

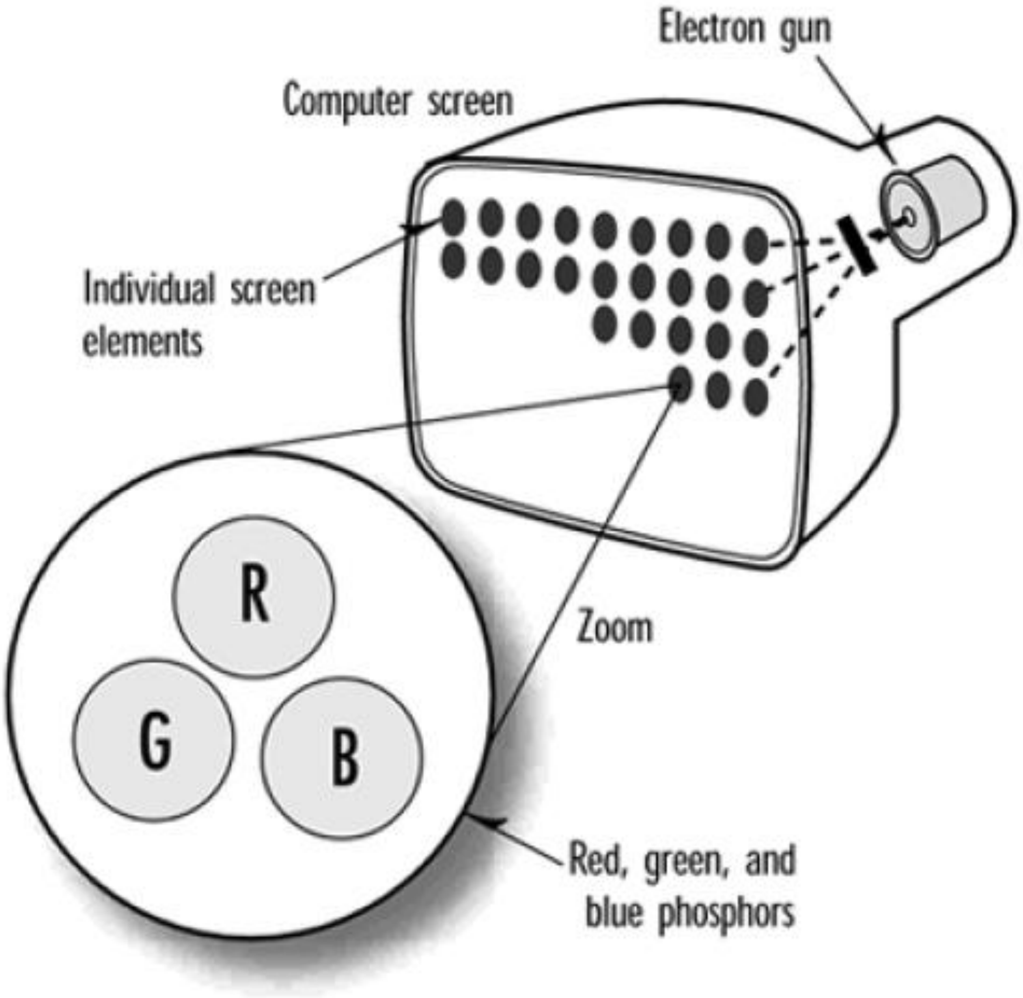


What is color?

- You can see that any “color” that your eye perceives actually consists of light all over the visible spectrum.
- For example, brown is composed of photon mix of 60% red photons, 40% green photons, and 10% blue photons.



How a computer monitor generates color?

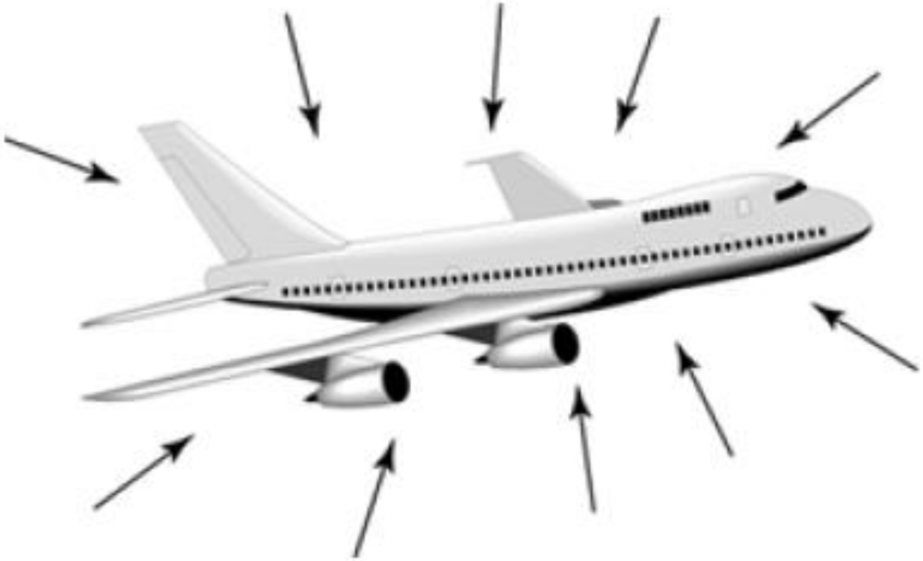


Color in the Real World

- Real objects don’t appear in a solid or shaded color based solely on their RGB values.
- OpenGL does a reasonably good job of approximating the real world in terms of lighting conditions.
- Unless an object emits its own light, it is illuminated by three different kinds of light:
    - ambient – орчны гэрэл
    - diffuse – тархсан гэрэл
    - specular –ойсон гэрэл

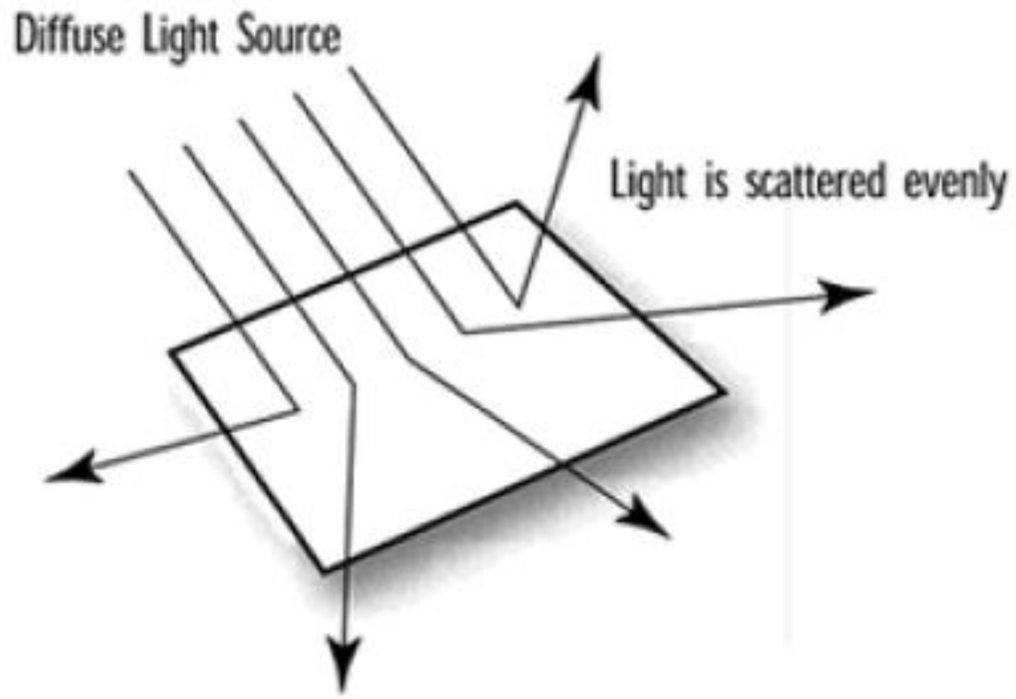
Ambient Light

- Ambient light doesn't come from any particular direction.
- It has a source, but the rays of light have bounced around the room or scene and become directionless.
- Objects illuminated by ambient light are evenly lit on all surfaces in all directions.



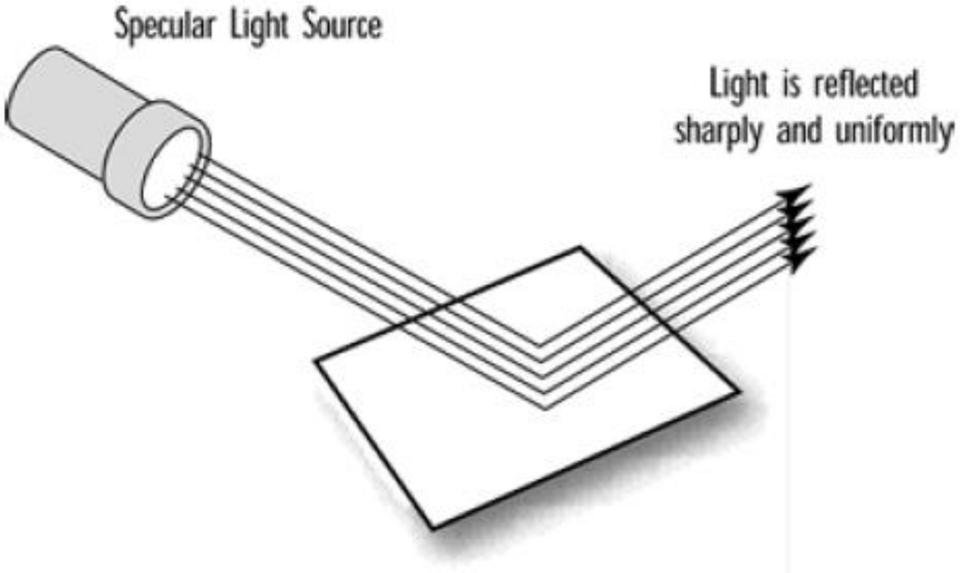
Diffuse Light

- Diffuse light comes from a particular direction but is reflected evenly off a surface.

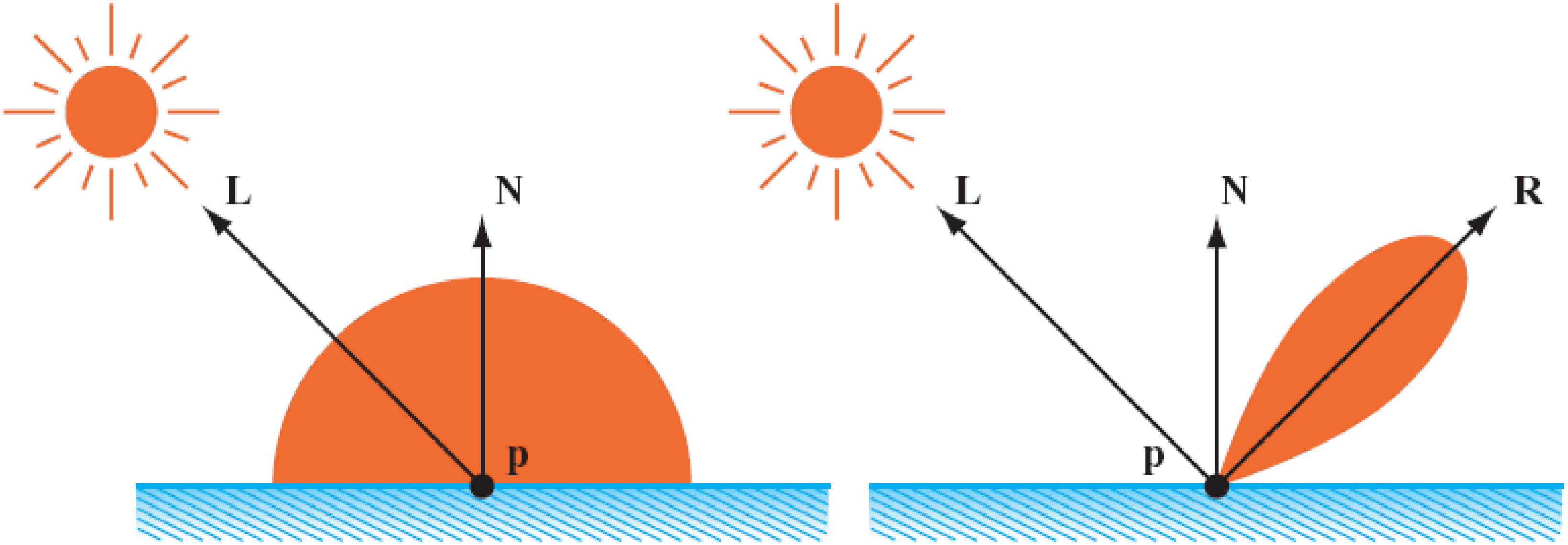


Specular Light

- Like light comes from a particular direction but is reflected evenly off diffuse light, specular light is directional, but it is reflected sharply and in a particular direction.



Diffuse and Specular Reflection Pattern

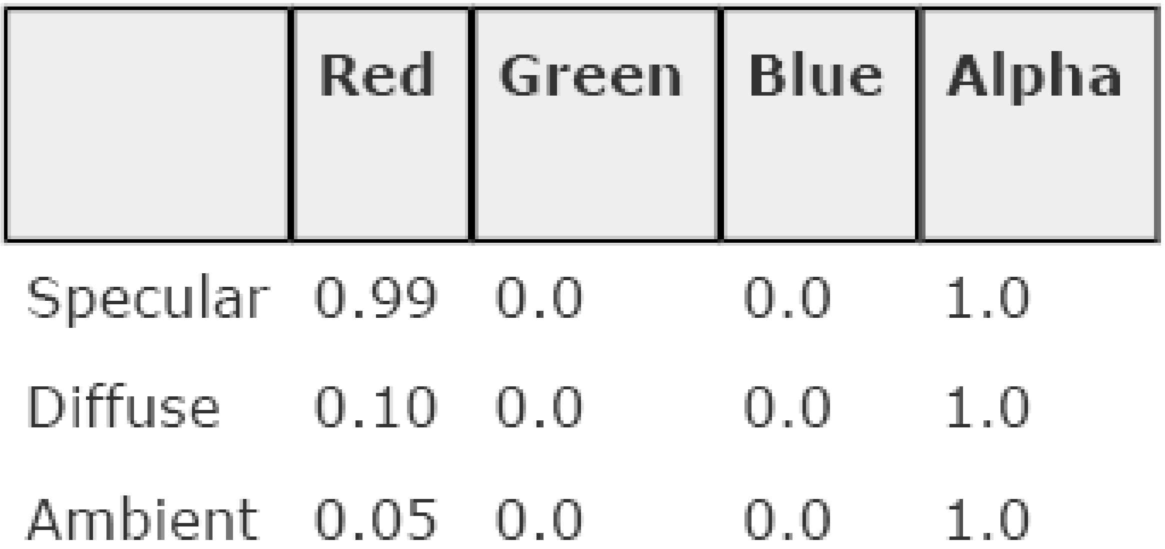


DIFFUSE REFLECTION

SPECULAR REFLECTION

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



	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

Materials in Real World

- In real world, objects do have a color of their own.
- A blue ball reflects mostly blue photons and absorbs most others.
- Under white light, 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 ball appears black to the viewer.

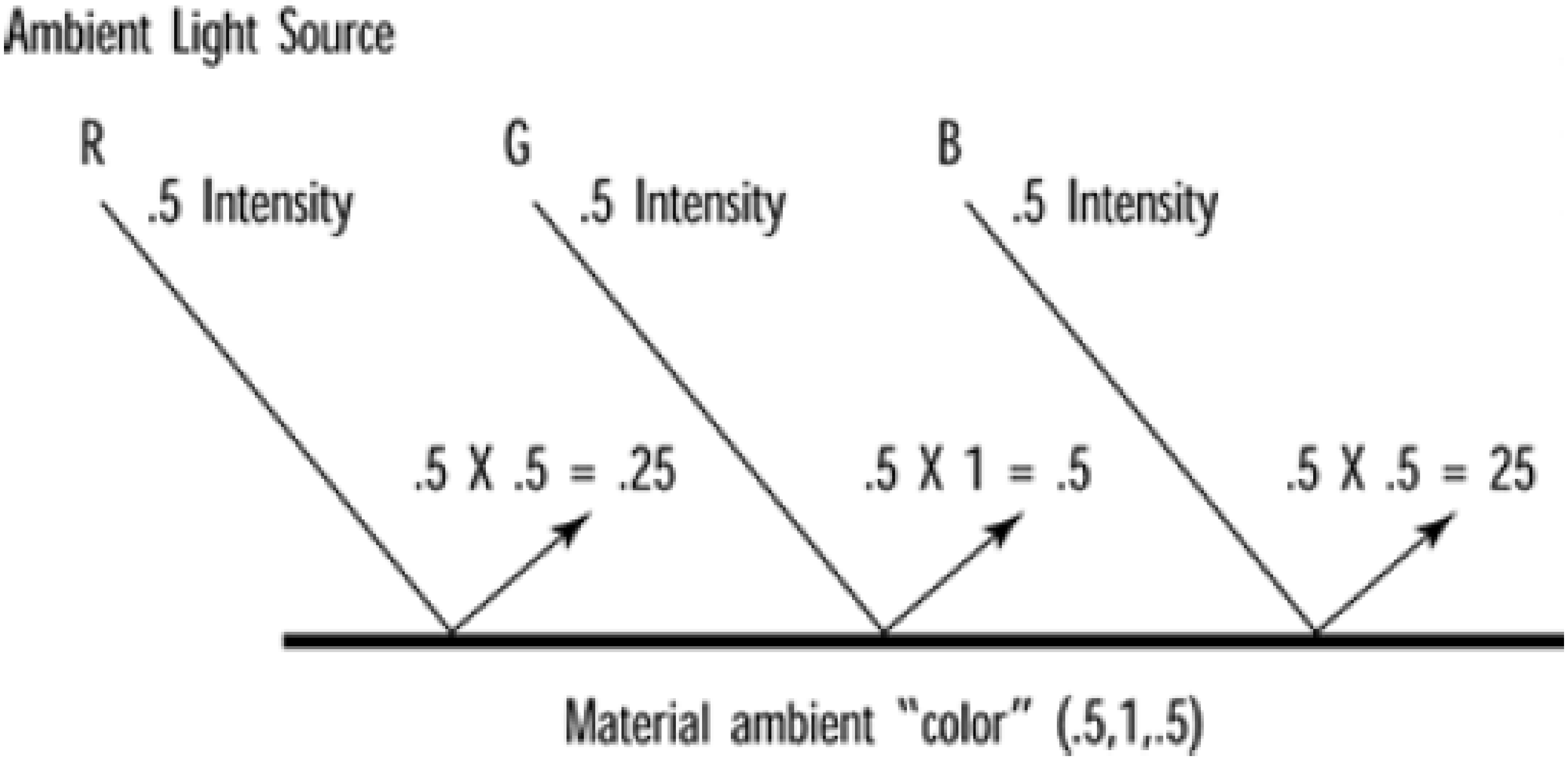
Material Properties

- 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.

Adding light to Materials

- 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?

Calculating ambient light effects



Enable the Lighting

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

  glEnable(GL_LIGHTING);

Setting up the Lighting Model

```
// Bright white light – full intensity RGB values
GLfloat ambientLight[] = { 1.0f, 1.0f, 1.0f, 1.0f };
// Enable lighting
glEnable(GL_LIGHTING);
// Set light model to use ambient light specified by ambientLight[]
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,
                                ambientLight);
```

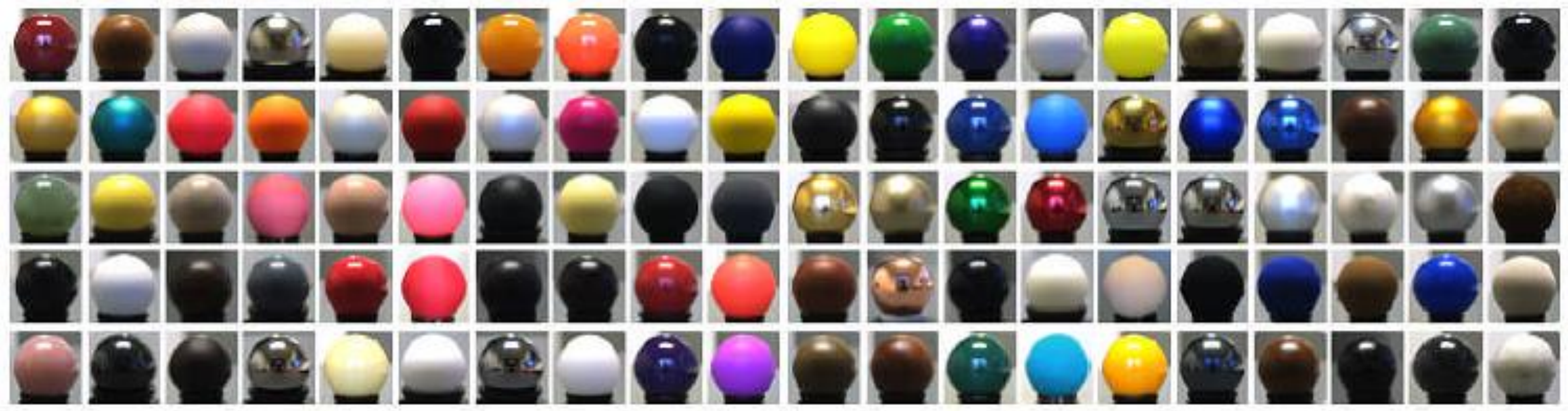
Setting Material Properties

```
Glfloat gray[] = { 0.75f, 0.75f, 0.75f, 1.0f };
...
glMaterialfv(GL_FRONT,
   GL_AMBIENT_AND_DIFFUSE, gray);
glBegin(GL_TRIANGLES);
   glVertex3f(-15.0f,0.0f,30.0f);
   glVertex3f(0.0f, 15.0f, 30.0f);
   glVertex3f(0.0f, 0.0f, -56.0f);
glEnd();
```

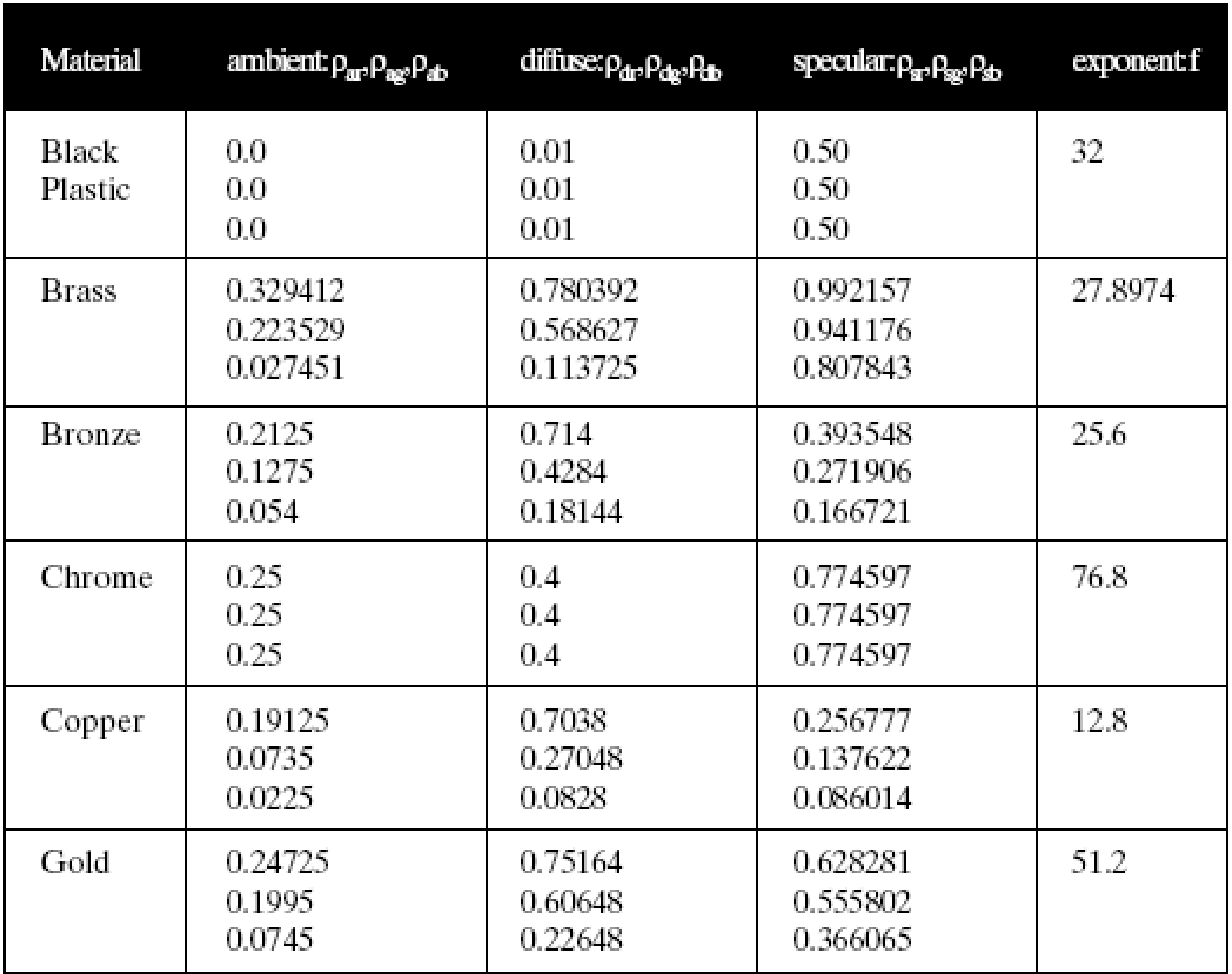
2nd preferred way of setting material properties

```
// Enable color tracking
glEnable(GL_COLOR_MATERIAL);
// Front material ambient and diffuse colors track glColor
glColorMaterial(GL_FRONT,GL_AMBIENT_AND_DIFFUSE);
...
...
glColor3f(0.75f, 0.75f, 0.75f);
glBegin(GL_TRIANGLES);
    glVertex3f(-15.0f,0.0f,30.0f);
    glVertex3f(0.0f, 15.0f, 30.0f);
    glVertex3f(0.0f, 0.0f, -56.0f);
glEnd();
```

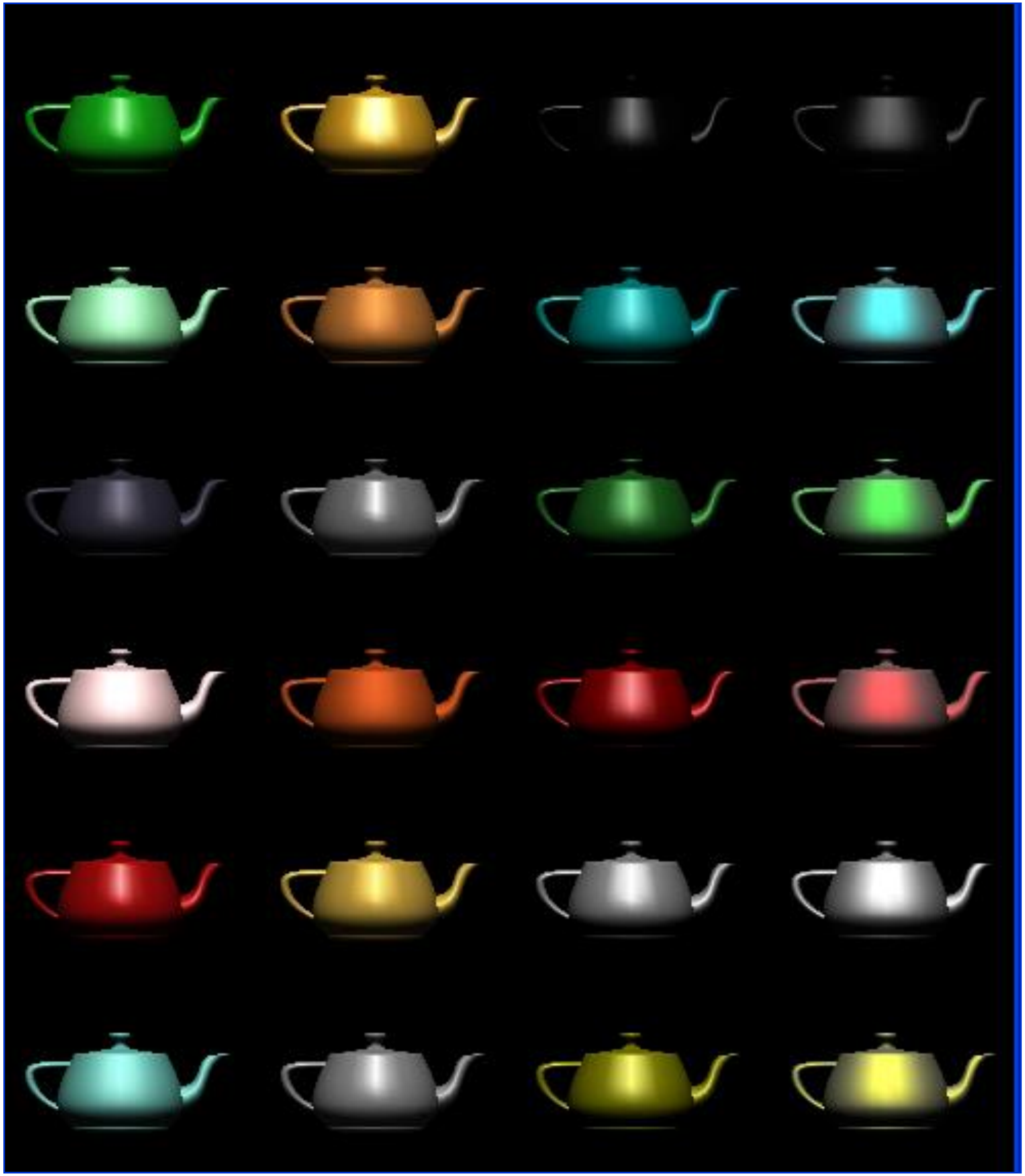
Materials



Parameters for common materials



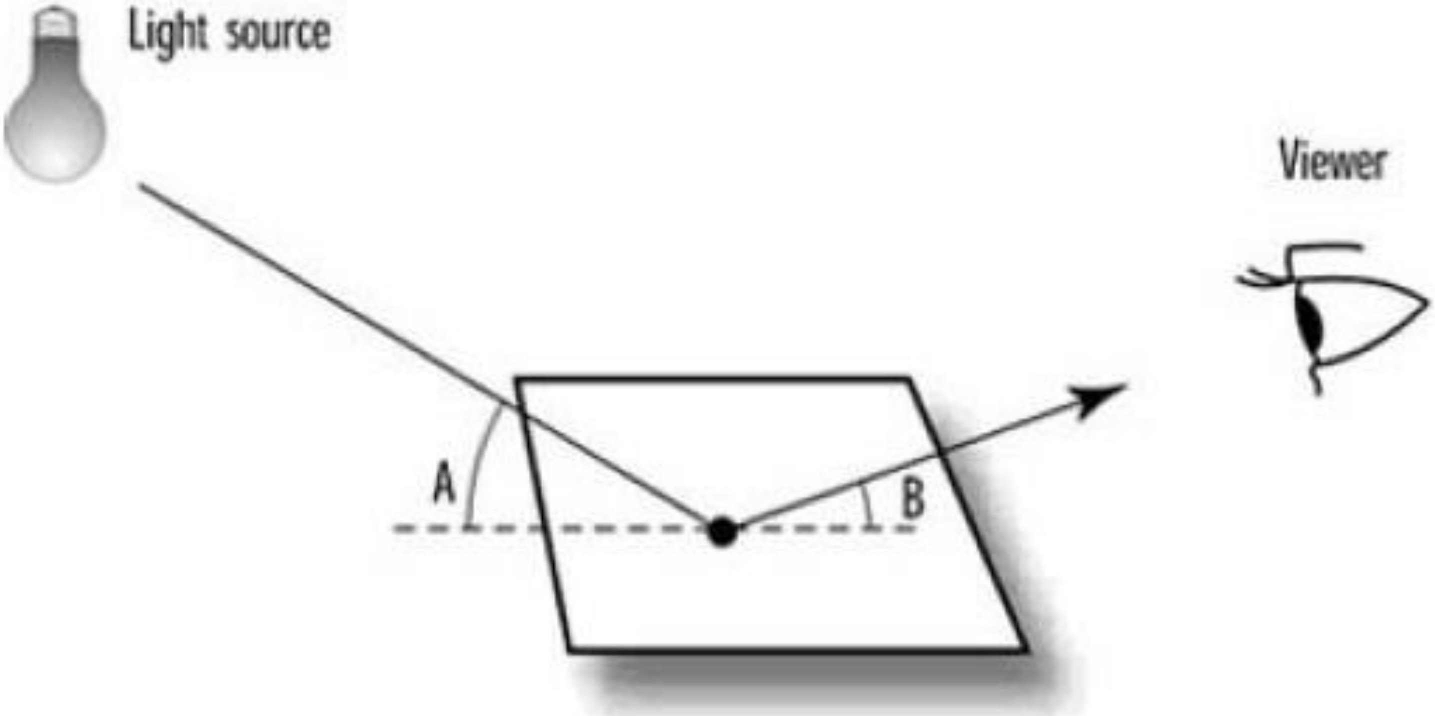
Material	ambient: $\rho_{ar}, \rho_{ag}, \rho_{ab}$	diffuse: $\rho_{dr}, \rho_{dg}, \rho_{db}$	specular: $\rho_{sr}, \rho_{sg}, \rho_{sb}$	exponent: f
Black Plastic	0.0 0.0 0.0	0.01 0.01 0.01	0.50 0.50 0.50	32
Brass	0.329412 0.223529 0.027451	0.780392 0.568627 0.113725	0.992157 0.941176 0.807843	27.8974
Bronze	0.2125 0.1275 0.054	0.714 0.4284 0.18144	0.393548 0.271906 0.166721	25.6
Chrome	0.25 0.25 0.25	0.4 0.4 0.4	0.774597 0.774597 0.774597	76.8
Copper	0.19125 0.0735 0.0225	0.7038 0.27048 0.0828	0.256777 0.137622 0.086014	12.8
Gold	0.24725 0.1995 0.0745	0.75164 0.60648 0.22648	0.628281 0.555802 0.366065	51.2



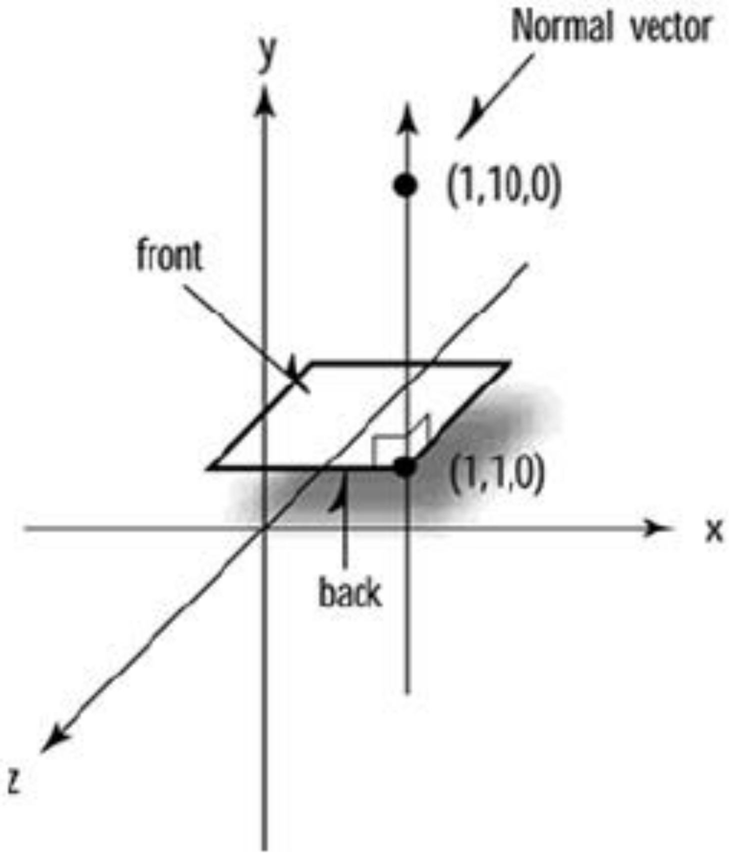
Using Light Source

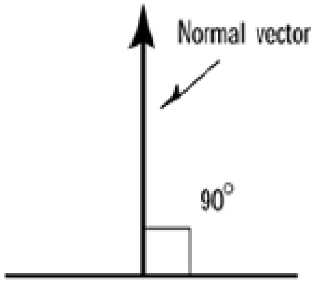
- For most applications attempting to model the real world, you must specify one or more specific sources of light.
- In addition to their **intensities** and **colors**, these sources have a **location** and/or a **direction**.

Which way is up?

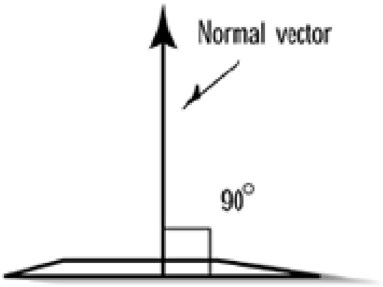


Surface normals



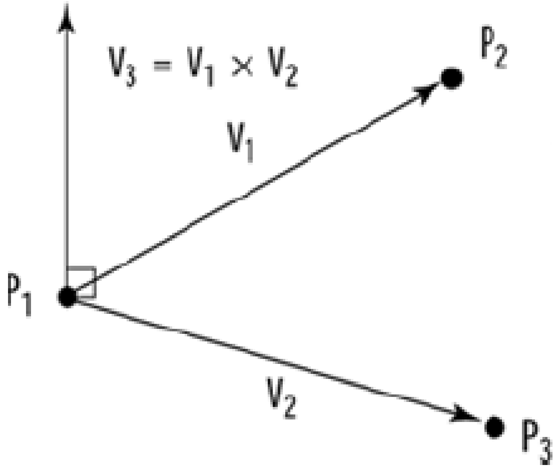


A 2D normal vector



A 3D normal vector

Surface normals



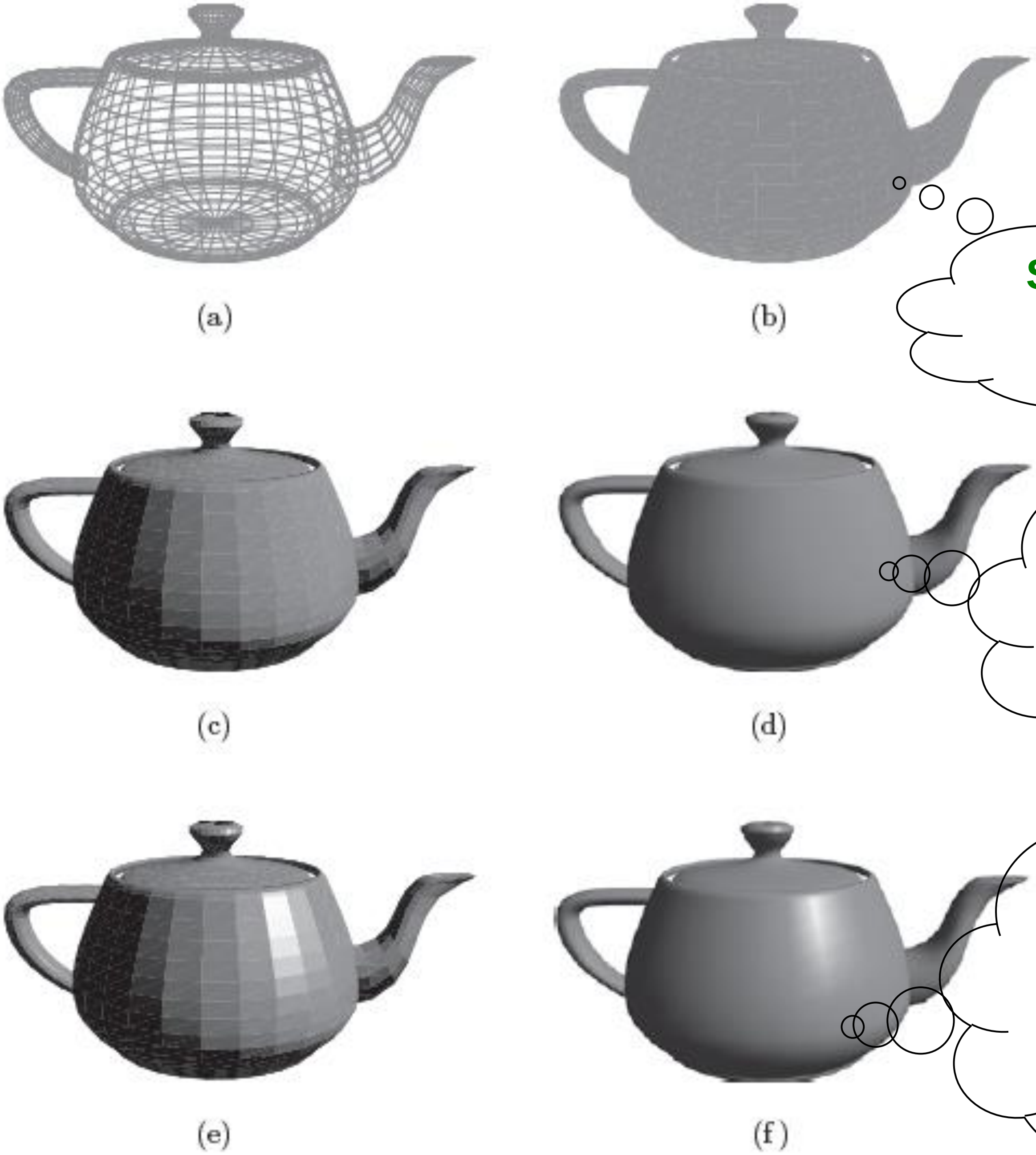
```
glBegin(GL_TRIANGLES);
        glNormal3f(0.0f, -1.0f, 0.0f);
        glVertex3f(0.0f, 0.0f, 60.0f);
        glVertex3f(-15.0f, 0.0f, 30.0f);
        glVertex3f(15.0f,0.0f,30.0f);
glEnd();
```

Setting up a Source

```
GLfloat ambientLight[] = { 0.3f, 0.3f, 0.3f, 1.0f };
GLfloat diffuseLight[] = { 0.7f, 0.7f, 0.7f, 1.0f };
...
...
// Set up and enable light 0
glLightfv(GL_LIGHT0,GL_AMBIENT,ambientLight);
glLightfv(GL_LIGHT0,GL_DIFFUSE,diffuseLight);

glEnable(GL_LIGHT0);

GLfloat lightPos[] = { -50.f, 50.0f, 100.0f, 1.0f };
...
...
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```



(a)

(b)

(c)

(d)

(e)

(f)

Solid color but no lighting or shading

Teapot drawn with Gouraud shading with only ambient and diffuse

Teapot drawn with Gouraud shading with only ambient and diffuse and specular lighting