CSCI3260 PRINCIPLES OF COMPUTER GRAPHICS

Tutorial 4

Colman Leung

Outline

- Lighting and Material
- Keyboard
- Animation

OpenGL Lighting

- The color of light sources is characterized by amount of Red, Green, and Blue light they emit
- The material of surfaces is characterized by the percentage of the incoming R, G, B components that is reflected in various direction
- The light in the scene comes from several light sources which can be individually turned on and off
- In OpenGL lighting model, the lighting is divided into four independent components: emissive, ambient, diffuse, and specular
- All four components are computed independently and then added together

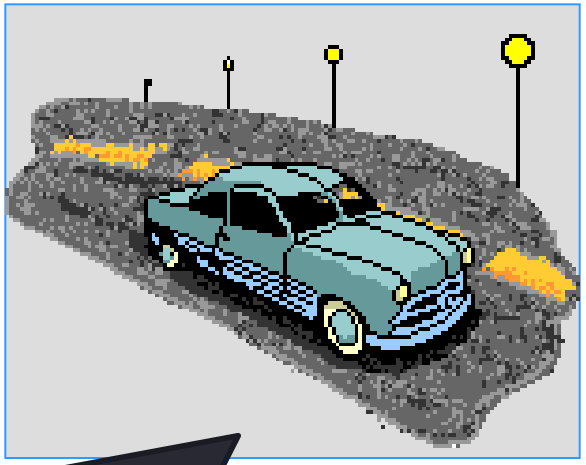
Ambient Light

- The light that's been scattered so much by the environment that its direction is impossible to determine
- It seems to come from all directions

Backlighting in a room

- **Large ambient component**
- **Light reaches you after bouncing off many surfaces**

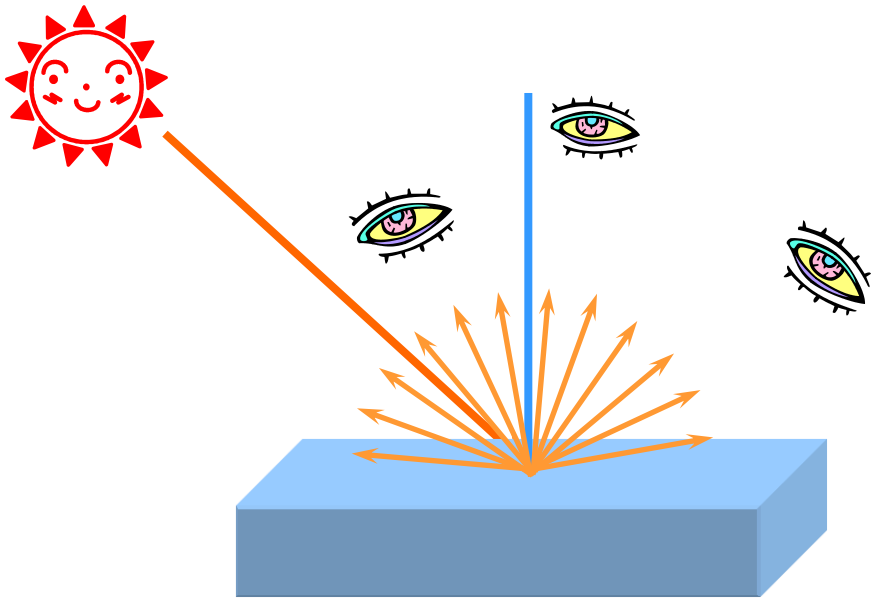
Spotlight outdoors



- **Tiny ambient component**
- **Most of the light travels in the same direction**
- **Very little of the light reaches your eyes after bouncing off other objects**

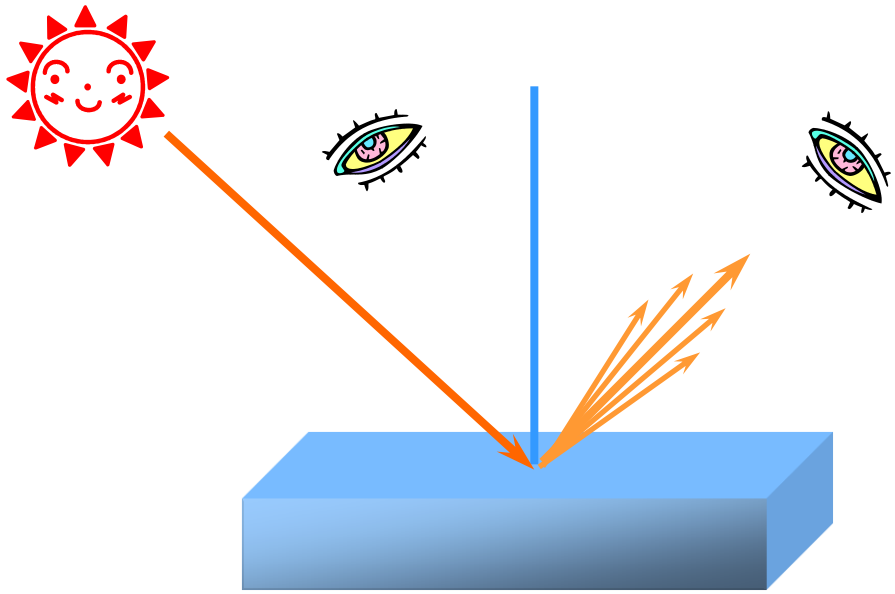
Diffuse Component

- Comes from one direction
- Once it hits a surface, it's scattered equally in all directions
- The object appears equally bright, no matter where the eye is located



Specular light

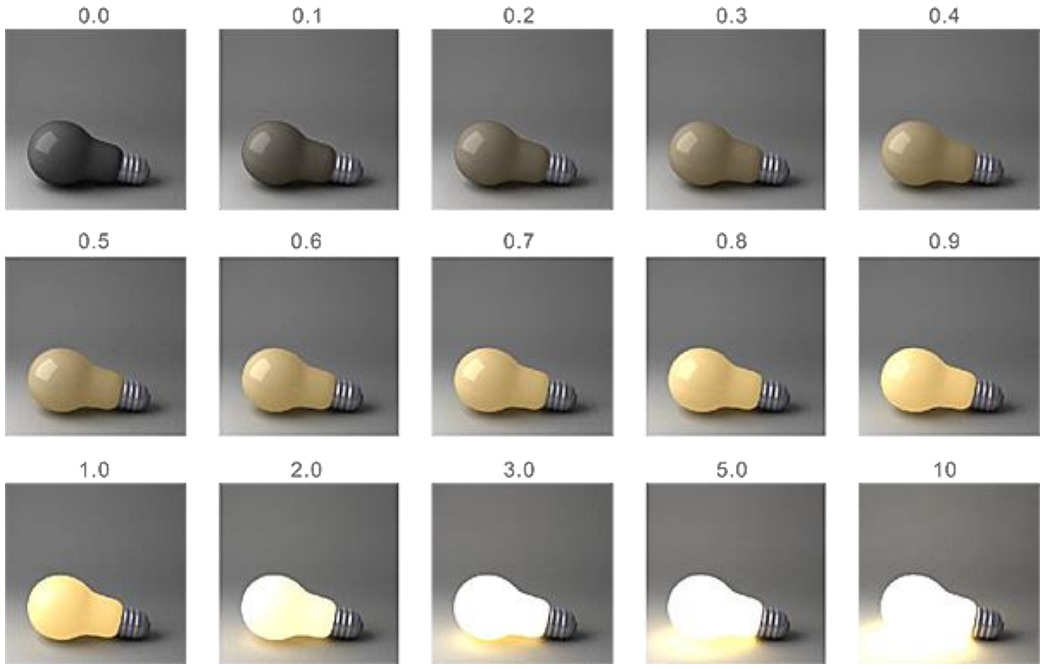
- Comes from a particular direction
- Tends to bounce off the surface in a preferred direction
- Think of specular as shininess



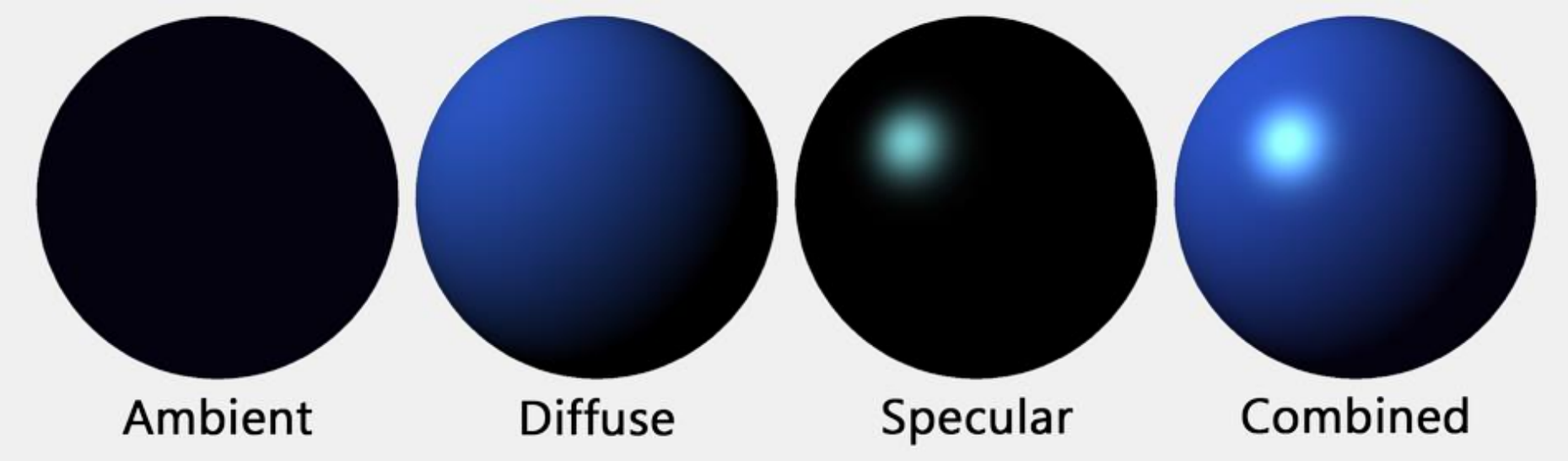
E.g. A laser beam bouncing off a mirror produces almost 100 percent specular reflection

Emissive Light

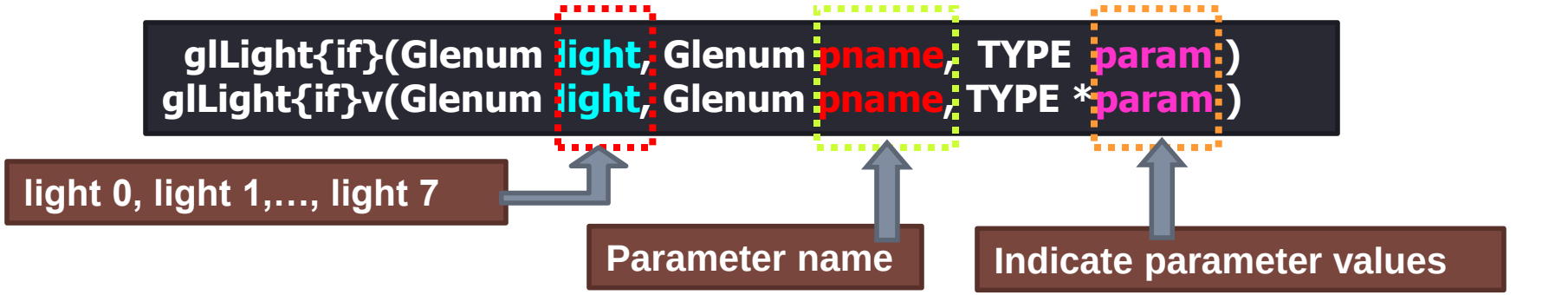
- This is the light that an object gives off by itself. A typical example of this is a light source itself.
- Specified by material property



The whole picture



Creating Light Sources



RGBA intensity of the ambient light that a light source adds to the scene

RGBA color of the diffuse light that a light source adds to the scene "the color of a light"

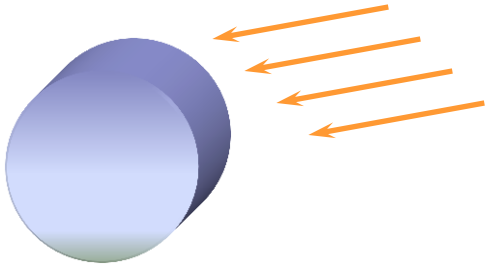
```
GLfloat light_ambient[] = {0.0, 0.0, 0.0, 1.0};
GLfloat light_diffuse[] = {1.0, 1.0, 1.0, 1.0};
GLfloat light_specular[] = {1.0, 1.0, 1.0, 1.0};
GLfloat light_position[] = {1.0, 1.0, 1.0, 0.0};

glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE,  light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

Affects the color of the specular highlight on an object

Position the light
Determine the light type

Directional and Positional Lights

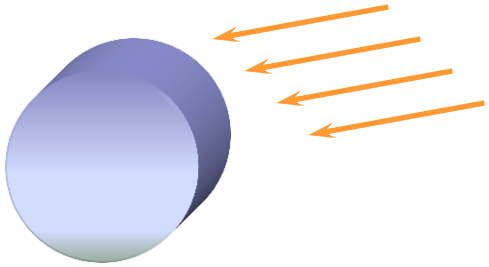
- Directional light: rays are parallel



```
GLfloat light_position[] ={x, y, z, 0.0);
```

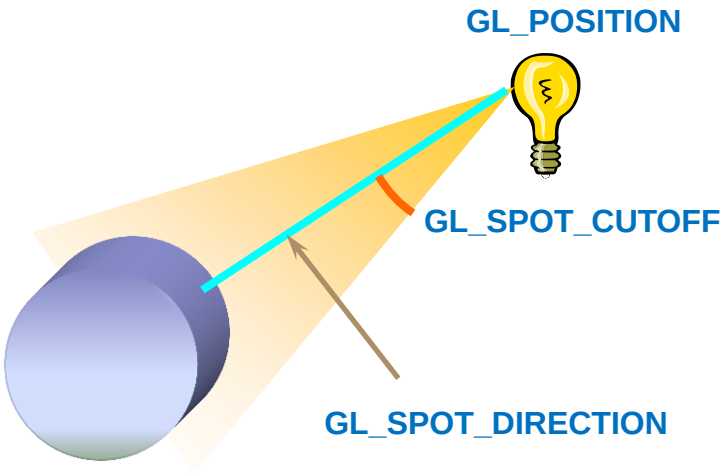
Directional and Positional Lights

- Directional light: rays are parallel



```
GLfloat light_position[] ={x, y, z, 0.0);
```

- Positional light: spotlights



w not equal 0

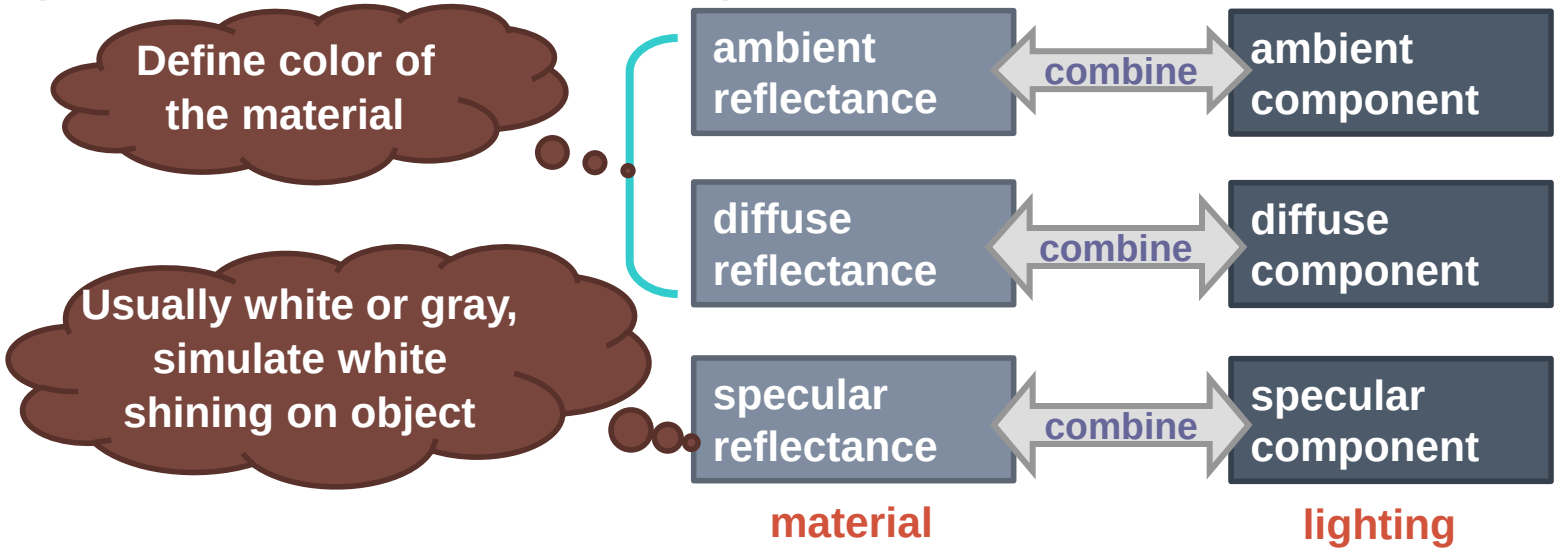
```
GLfloat spot_position[] ={x, y, z, w);
GLfloat spot_direction[] ={x', y', z');

glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);
glLightfv(GL_LIGHT0, GL_POSITION,
          spot_position);
glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION,
          spot_direction);
```

Sample code

```
/* Initialize Lighting */
GLfloat light_pos0[] = {2.0, 2.0, 2.0, 1.0};
GLfloat spot_dir[]   = {-1.0, -1.0, -1.0};
GLfloat light_amb0[] = {0.0, 0.0, 0.0, 1.0};
GLfloat light_dif0[] = {1.0, 1.0, 1.0, 1.0};
GLfloat light_spc0[] = {1.0, 1.0, 1.0, 1.0};

glLightfv(GL_LIGHT0, GL_POSITION, light_pos0);
glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 30.0);
glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spot_dir);

glLightfv(GL_LIGHT0, GL_AMBIENT, light_amb0);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_dif0);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_spc0);

glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0); // disabled by glDisable(GL_LIGHT0)
```

Material Color

- A material's color depends on the percentages of the incoming R, G, B light it reflects
- Materials also have different ambient, diffuse, and specular colors
- Determine the ambient, diffuse, and specular reflectance (values between 0 and 1) of the material



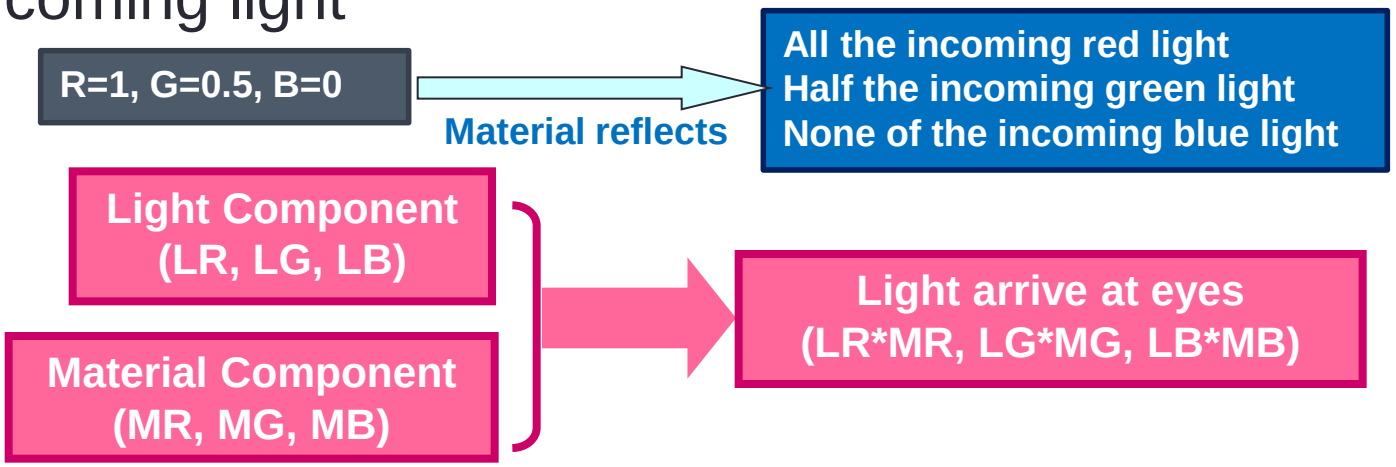
RGB Values for Lights and Materials

- For light, RGB numbers correspond to a percentage of full intensity for each color

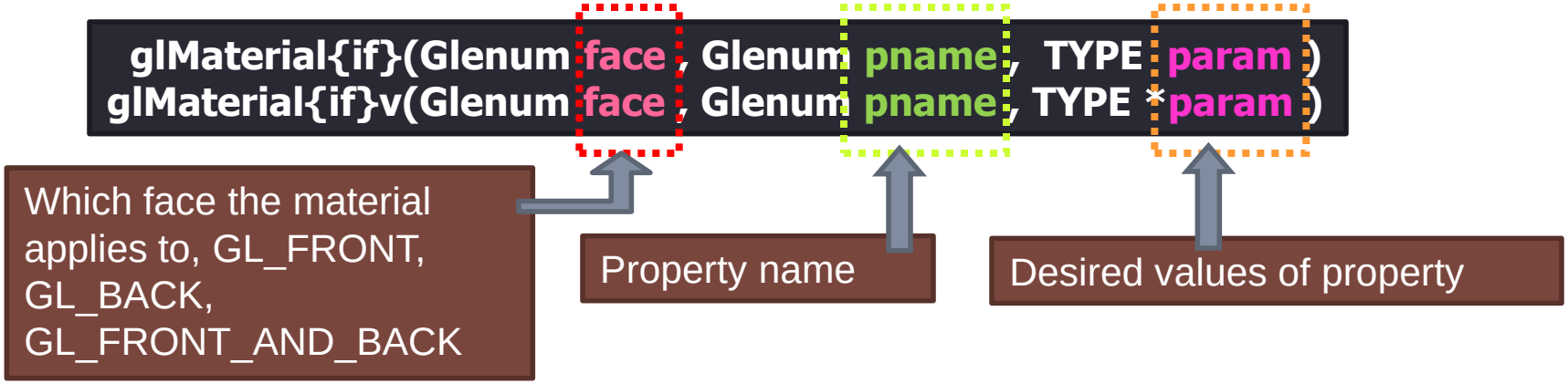
R=1, G=1, B=1
White

R=0.5, G=0.5, B=0.5
appear gray

- For material, RGB numbers correspond to the reflected proportions of RGB component from incoming light



Defining Material Properties



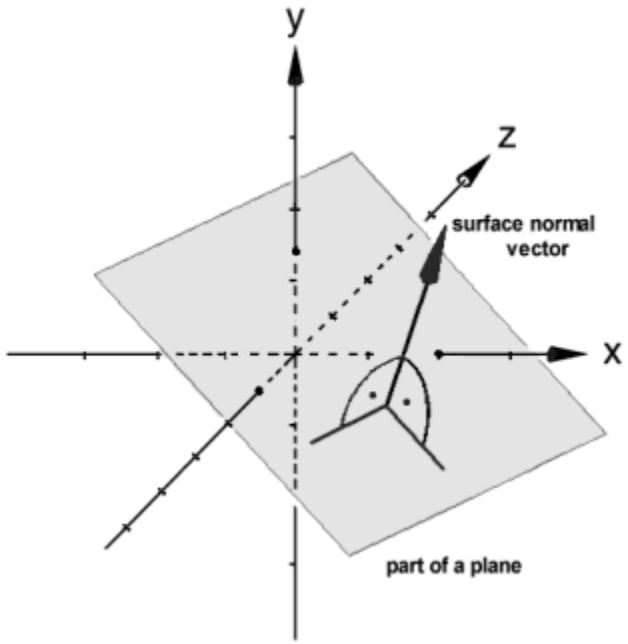
Parameter Name	Meaning
GL_AMBIENT	Ambient color of material
GL_DIFFUSE	Diffuse color of material
GL_AMBIENT_AND _DIFFUSE	Ambient and diffuse color of material, have same RGBA values
GL_SPECULAR	Specular color of material, highlight produced by reflection
GL_SHININESS	Specular exponent, control the size and brightness of the highlight
GL_EMISSION	Emission color of material, make an object appear to be giving off light of that color

Sample code

```
/* Initialize material properties */
GLfloat no_mat[] = {0.0,0.0,0.0,1.0};
GLfloat mat_diffuse[] = {0.8,0.2,0.5,1.0};
GLfloat mat_specular[] = {1.0,1.0,1.0,1.0};
GLfloat high_shininess[] = {100.0};

glMaterialfv(GL_FRONT,GL_DIFFUSE,mat_diffuse);
glMaterialfv(GL_FRONT,GL_SPECULAR,mat_specular);
glMaterialfv(GL_FRONT,GL_SHININESS,high_shininess);
glMaterialfv(GL_FRONT,GL_EMISSION,no_mat);

glEnable(GL_COLOR_MATERIAL);
```

Two side lighting

- glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);



User Input Callbacks

- Process user input
- glutKeyboardFunc(*keyboard*);
- glutMouseFunc(*mouse*);

User Input Callbacks

```
void keyboard( unsigned char key, int x, int y)
{
   switch( key ) {
      case'\033': // press button ’ESC’
               exit(0); // exit
               break;
      case ‘a':     //press ‘a’ to start animation
               glutIdleFunc(idlefunc); //use idle function
               break;
      case ‘b':
               glutIdleFunc(spinfunc); //use another idle function
               break;
      ……
     }
  }
```

Key: the generated ASCII character
x y: mouse coordinates in the window

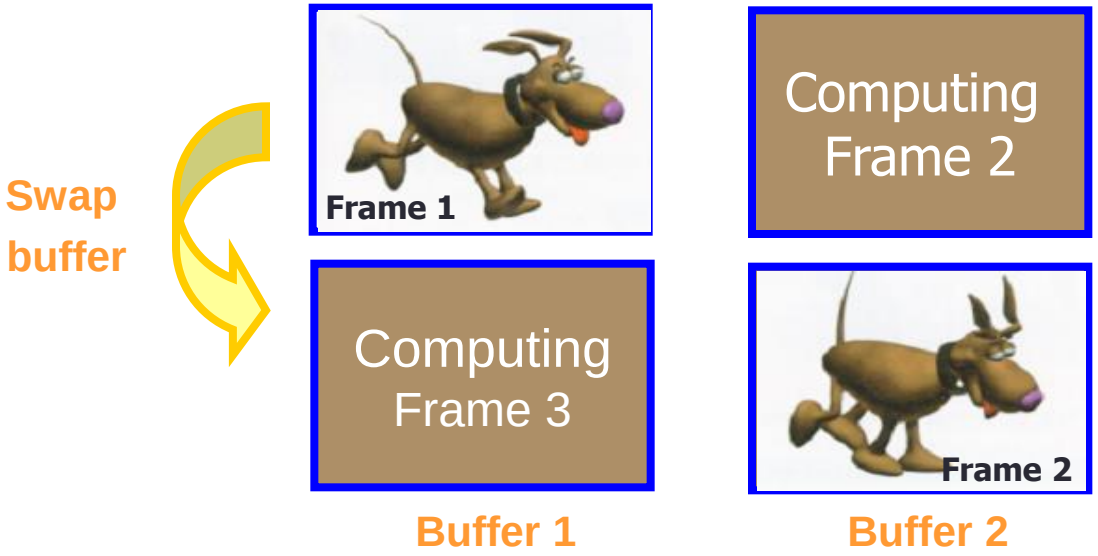
User Input Callbacks

```
void mouse(int button, int state, int x, int y)
{
    if(state == GLUT_DOWN && button == GLUT_LEFT_BUTTON){
            changeColor = true;
    }
}
```

button: the button of the mouse that triggered the event
state: the state of the event
x y: mouse coordinates in the window

Animation

- Computer graphics screen typically refresh 60 to 70 times per second
- Each frame is complete when it is displayed
- Double-buffering – two complete color buffers
  - One is displayed while the other is calculating next scene to be drawn
  - When the drawing of the next frame is completed, the two buffers are swapped



Motion =

Redraw + Swap

Animation

- Animation is just like drawing consecutive frames with small differences in each frame

- glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  each time before we draw the next frame, we should clear the frame buffer before drawing next frame

- glutPostRedisplay()
  sets a flag so that on the next iteration of the glut main loop, your registered display() function is called.

  If you do not call this function, the main loop will keep looping the idle() function instead. So your animation will look as if it is stuck even though you have update your scene.

End