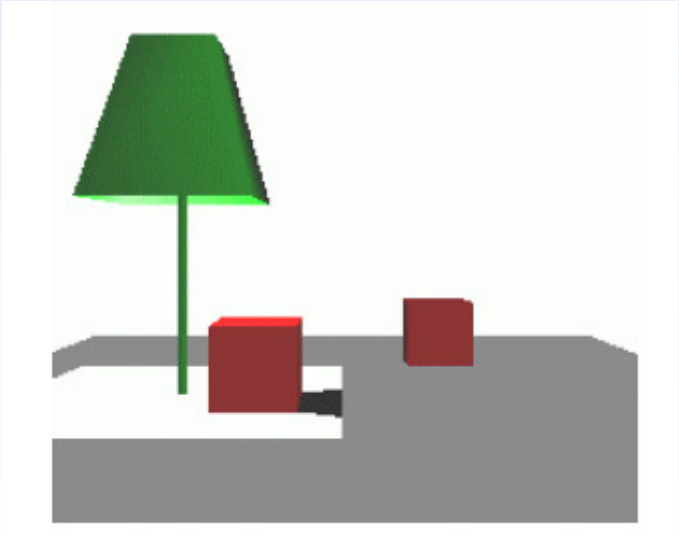
Computer Game Technologies

Java 3D Lighting

Lighting in Java 3D

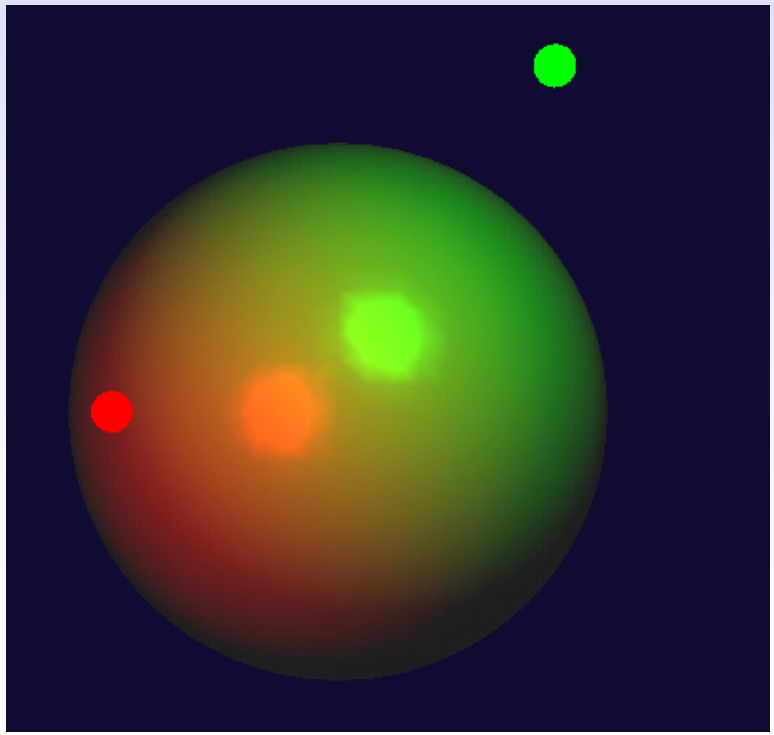
- Realistic 3D requires a scene to be lit
- An object's colour is then determined by its *material* plus the lights shining on it
  - *Shading* model
- A scene can have both lit and unlit objects



(Java Tutorial Chapt. 6)

Lighting Model

- How an object in the real world looks depends on:
  - Physical properties of object
  - Characteristics of light sources
  - Object's position relative to the light sources
  - Viewer's position relative to the object



Java 3D Lighting Model

- Material of object
- Characteristics of light sources
- Three vectors:
  - surface normal (N)
  - light direction (L)
  - viewer direction (E)

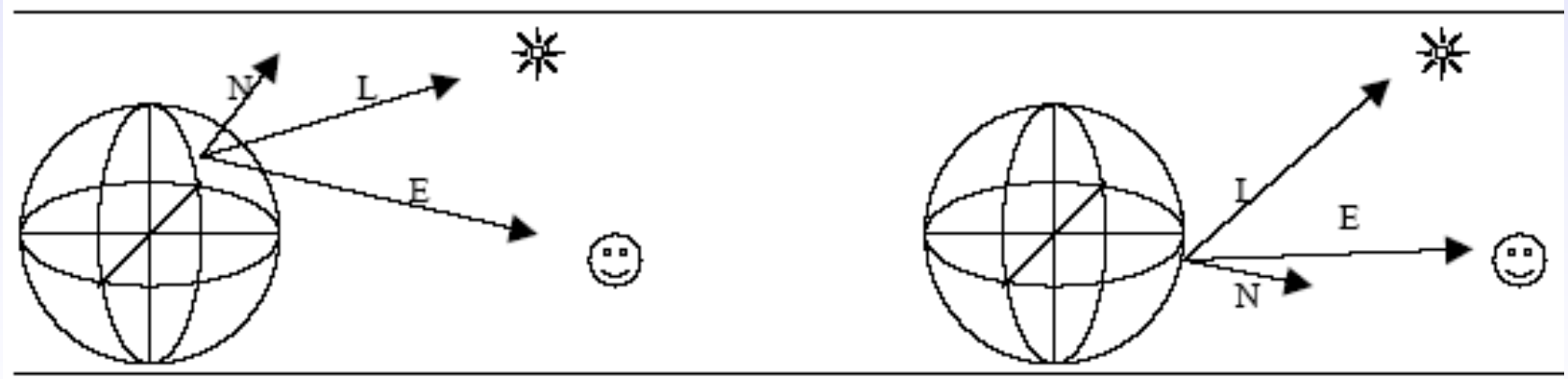


Figure 6-1 Light, Surface Normal, and Eye Vectors used to Shade Vertices.

Lighting Reflections

- Java 3D lighting model includes 3 types of light reflection from an object
  - *Ambient* results from low level background light
  - *Diffuse* is normal reflection from a light source
  - *Specular* are highlight reflections
- No shadows or interobject reflections!

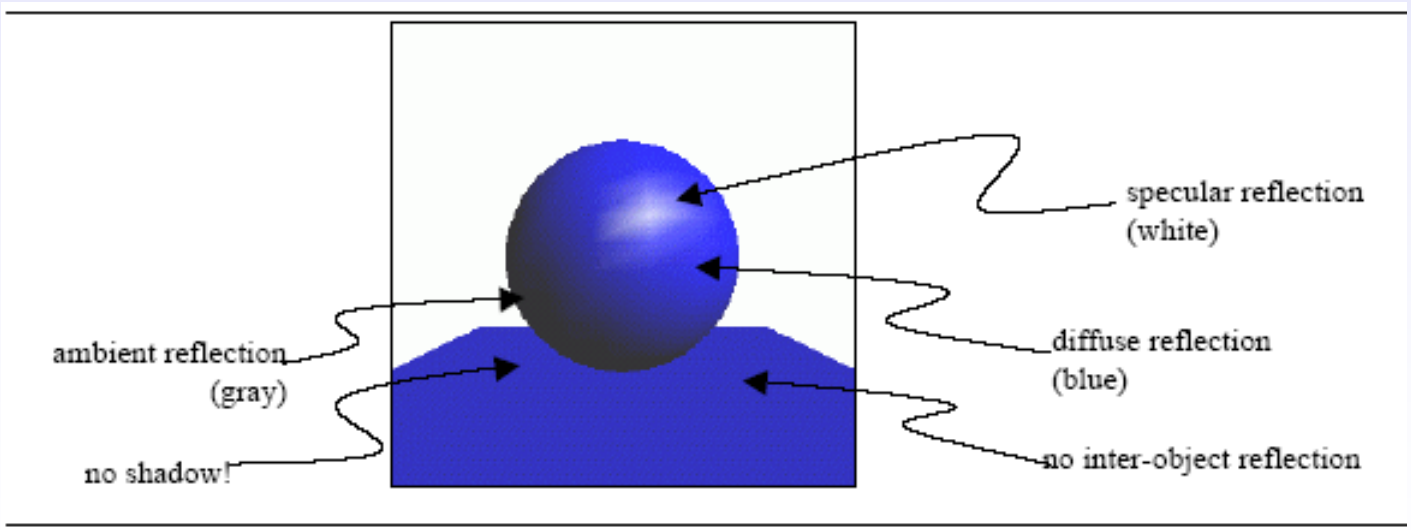


Figure 6-2 Shaded Sphere and Plane

Shading Model

- Lighting model first *shades* each vertex
  - Determines colour based on each light source
- Remaining pixel colours determined from vertices
  - Flat or Gouraud shading
- Flat shading choses colour of one vertex for all pixels in enclosed polygon
- Gouraud shading uses trilinear interpolation of colours from all vertices of a polygon

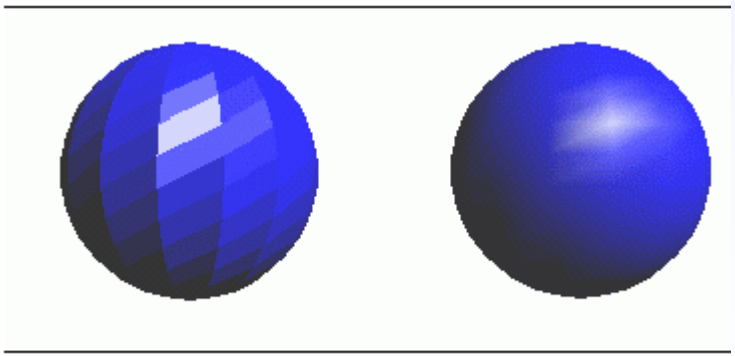
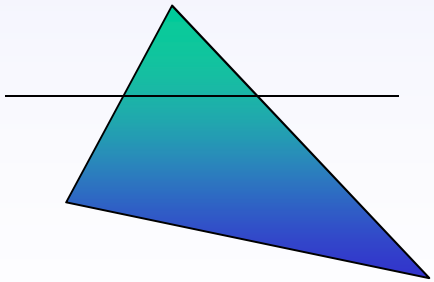
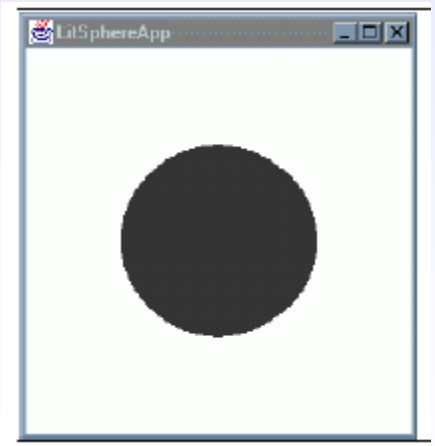


Figure 6-3 Flat and Gouraud Shaded Spheres.



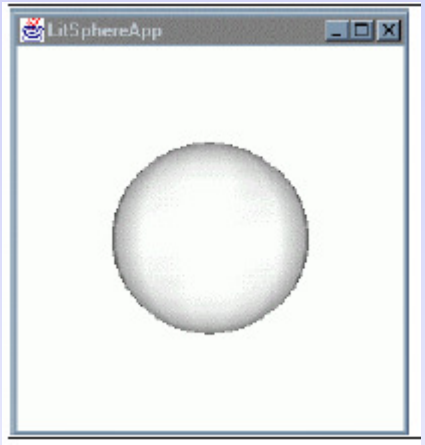
Lights

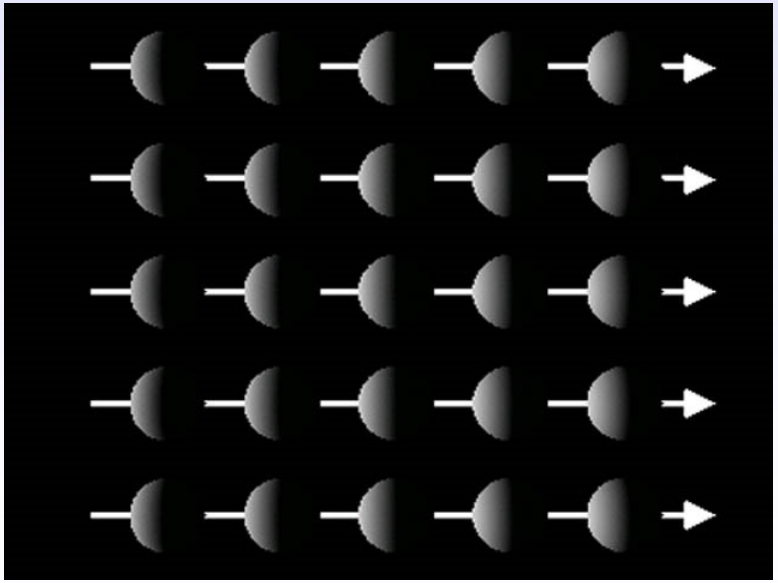
- Ambient
  - Light of same intensity at all places and in all directions
  - No location
  - Colour
  - Results in flat shading of lit objects
  - Used in combination with other types of light source



Lights (2)

- Directional
  - Light from one direction only
    - e.g. distant source such as the sun
  - Light vector (L) is constant
  - Direction and colour but no location





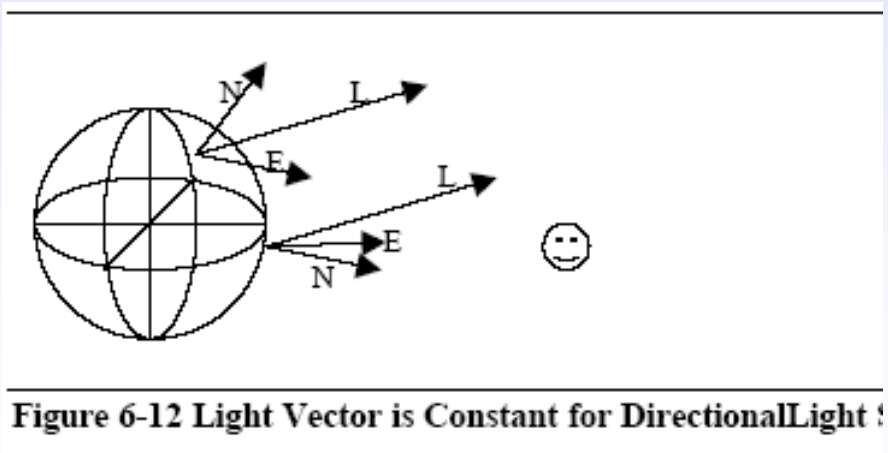


Figure 6-12 Light Vector is Constant for DirectionalLight S

Lights (3)

- Point
  - Omni directional light whose intensity decreases with distance from source (location)
  - Location and colour but no direction

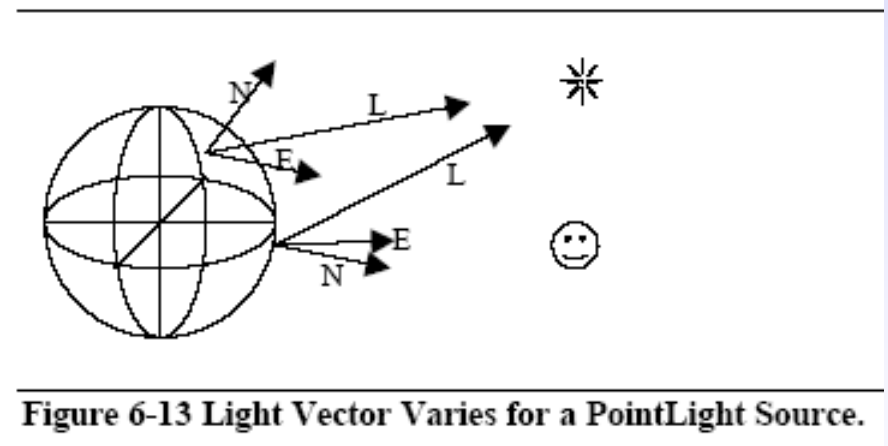
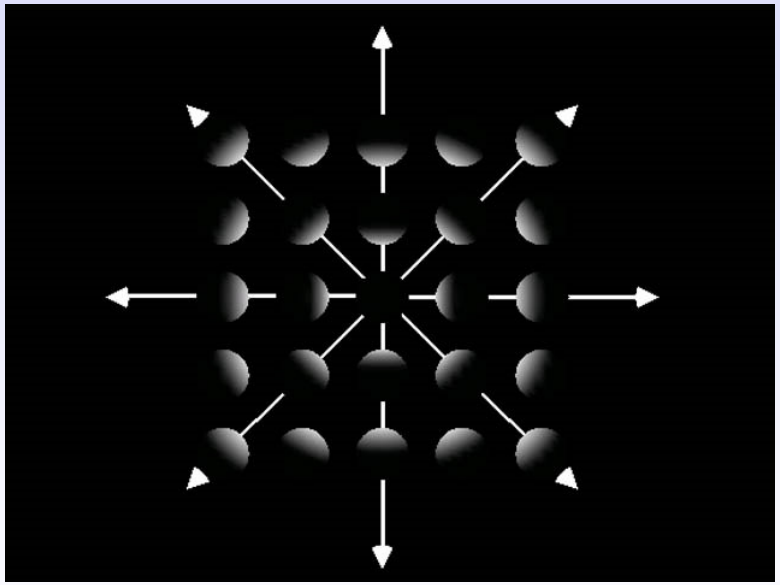


Figure 6-13 Light Vector Varies for a PointLight Source.



Lights (4)

- Spot
  - Subclass of point light with direction and concentration
  - Only light capable of lighting only a portion of an object

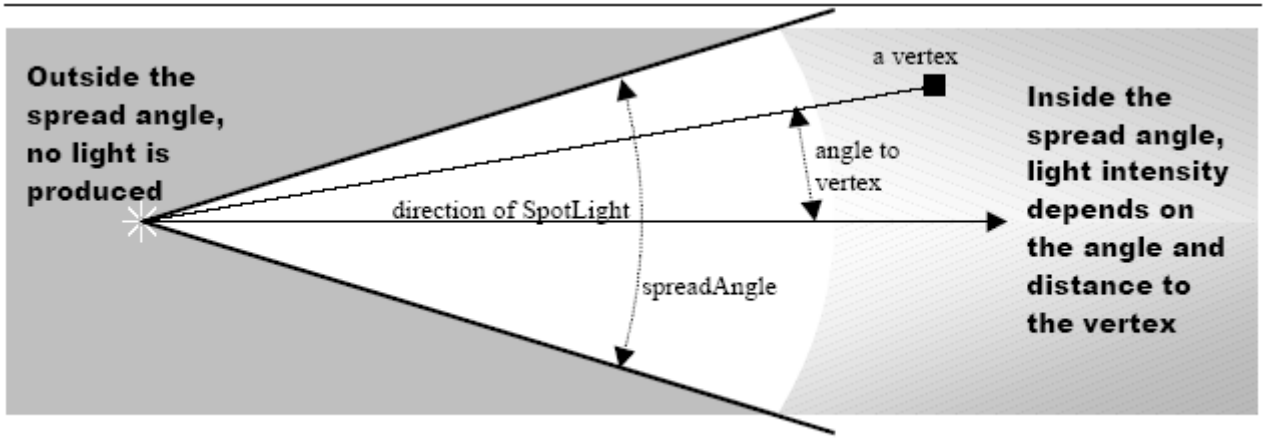
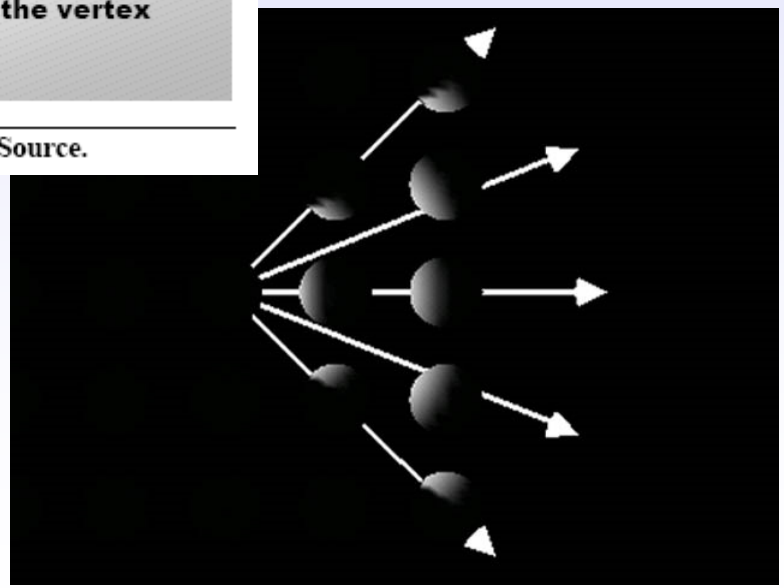
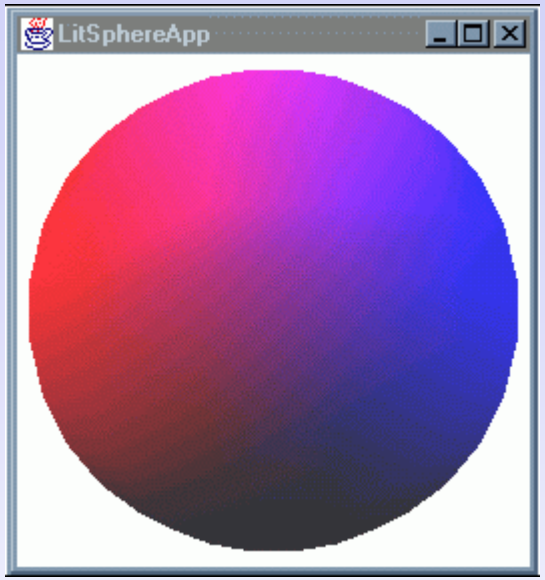


Figure 6-14 Light Intensity Varies with Distance and Orientation for a PointLight Source.

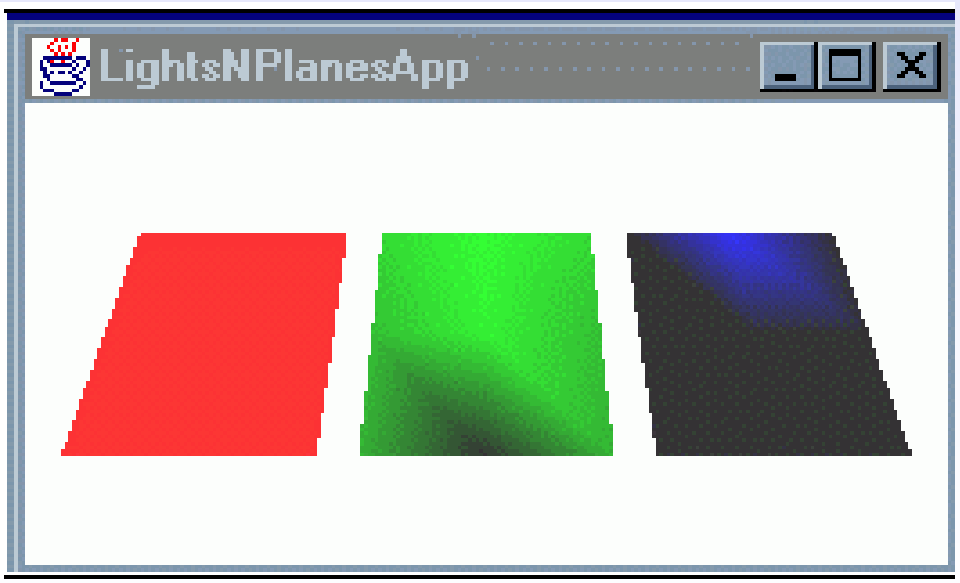


Examples

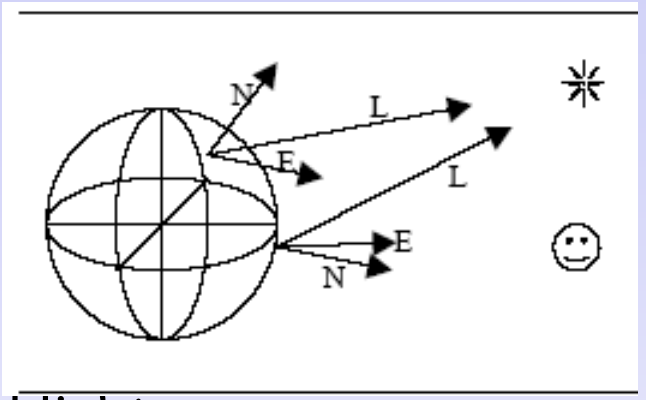
- Directional
  - Two lights: one red, one blue
  - White sphere



- Directional, Point and Spot



Light Vectors



- Eye vector (E) is assumed constant
  - For entire visible object
  - Much less computationally intensive
- Directional versus point light
  - Constant light vector (L) with directional light
  - Variable L with point light

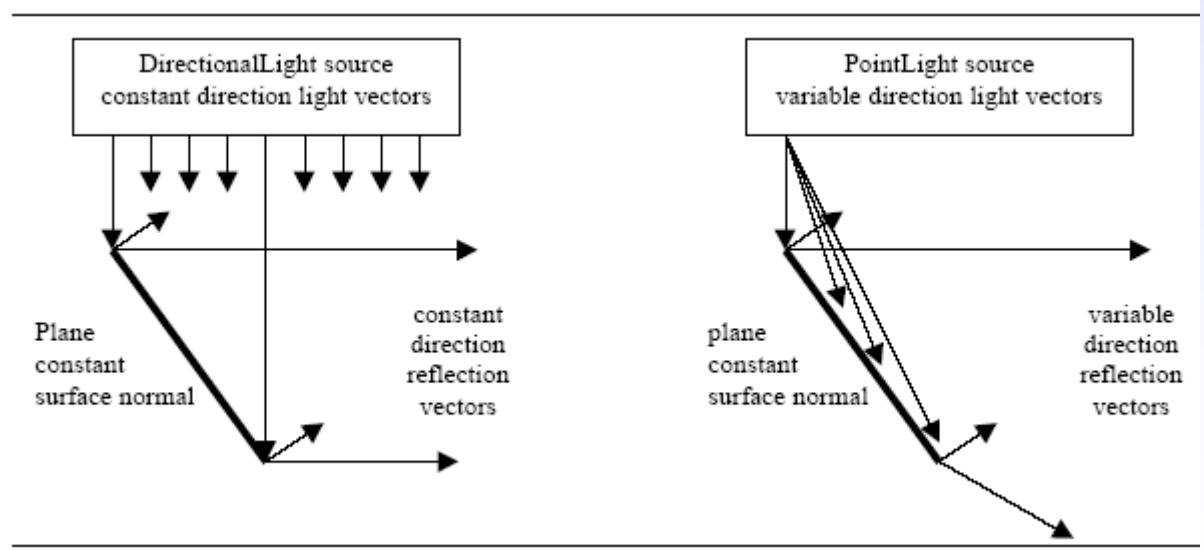
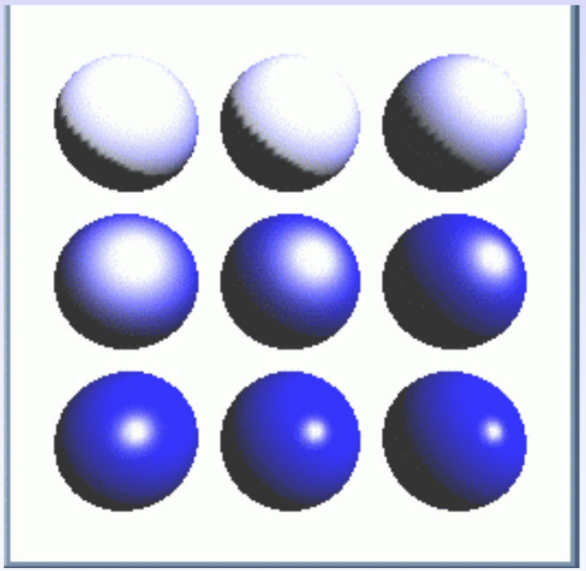


Figure 6-17 Geometry of Lighting Planes with Directional and Point Light Sources

Material Colours

- A visible object reacts to light through a *material*
- Material specifies colours and/or texture
  - We will consider textures later
- Ambient, diffusive
  - Colour of object
- Emissive
  - Glow-in-the-dark
- Specular, shininess
  - Reflective highlights



```
Material blueMat= new Material(blue, black, blue,
                                        specular, 25.0f);
       // sets ambient, emissive, diffuse, specular, shininess
blueMat.setLightingEnable(true);
```

Rendered Colours

- Phong lighting equation
  - Combines dull diffuse base with specular highlights
  - Approximates the look of plastic

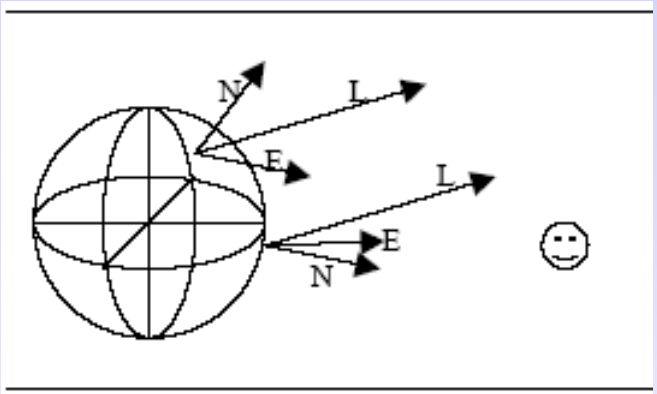
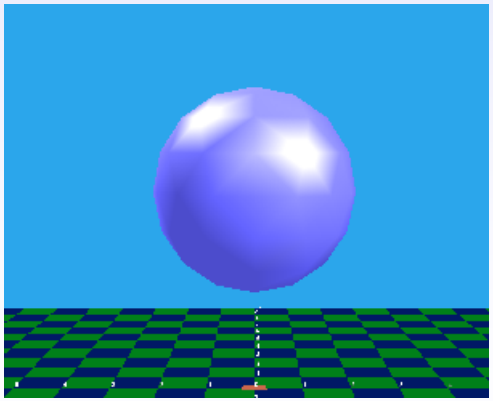


Figure 6-12 Light Vector is Constant for Di

$$C = C_d.C_l(N.L) + C_s.C_l(R.E)^s$$

  - R is the reflection vector (from N and L)

- Contributions from individual lights are summed



Constructing a Lit Scene

- Construct lights
  - Typically an ambient light plus 1 or 2 directional lights
  - Set influencing bounds for each light
  - Add lights to scene graph
- Construct visible objects
  - Specify surface normals
  - Add light-enabled material
  - Add objects to scene graph
- If any of these steps is missed out, then an object will not be lit
  - E.g. easy to forget the influencing bounds for a light, or even to add the light to the scene graph

Influence of Lights

- Region of influence of a light
  - Determined by its *influencing bounds*
  - Region that is affected by the light
    - Relative to light's location for point and spot lights
  - Usually specified as a sphere
- When a light's influencing bounds intersects the bounds of a visual object the light is used to shade the *entire* object

Influence of Lights (2)

- Light's position in the scene graph does not affect its region of influence
- However the bounds object referenced by the light is affected by scene graph location

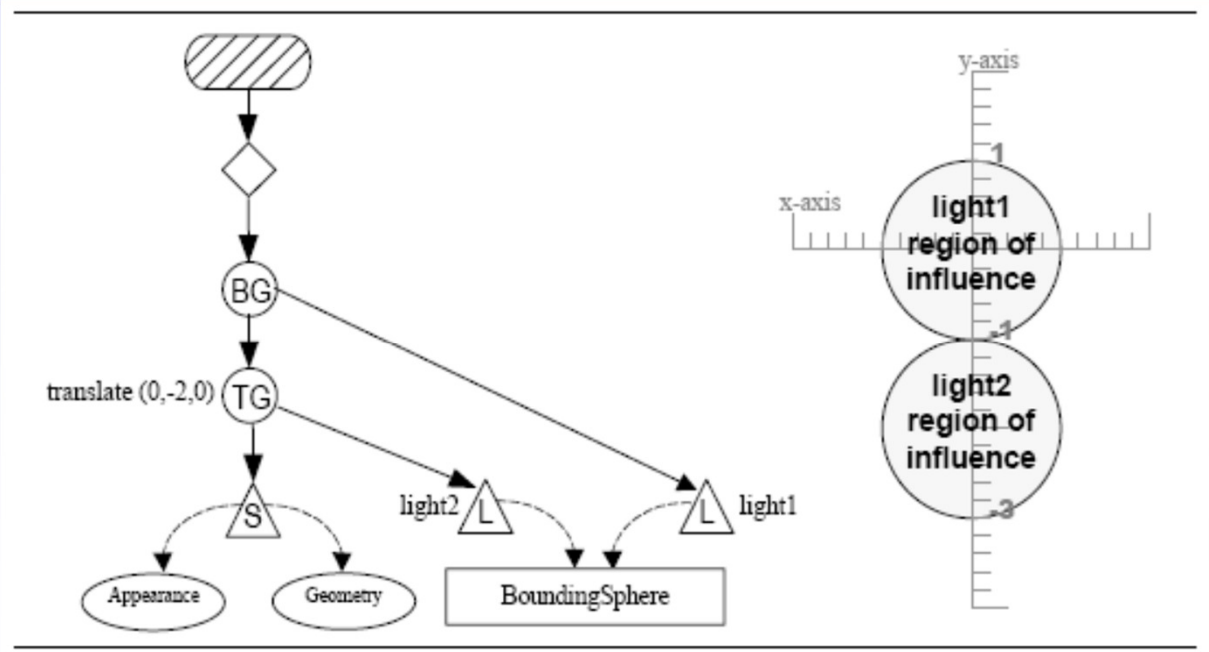
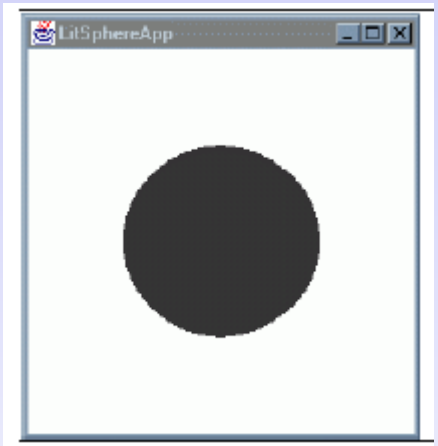


Figure 6-10 BoundingSphere Affected by Transformation

Simple Example - Ambient

```
1. Appearance createAppearance() {
2. Appearance appear = new Appearance();
3. Material material = new Material();
4. appear.setMaterial(material);
5.
6. return appear;
7. }
8.
9. BranchGroup createScene (){
10. BranchGroup scene = new BranchGroup();
11.
12. scene.addChild(new Sphere(0.5f, Sphere.GENERATE_NORMALS,
13. createAppearance()));
14.
15. AmbientLight lightA = new AmbientLight();
16. lightA.setInfluencingBounds(new BoundingSphere());
17. scene.addChild(lightA);
18.
19. return scene;
20. }
```



Simple Example - Directional

- Add following code fragment:

```
1. DirectionalLight lightD1 = new DirectionalLight();
2. lightD1.setInfluencingBounds(new BoundingSphere());
3. // customize DirectionalLight object
4. scene.addChild(lightD1);
```

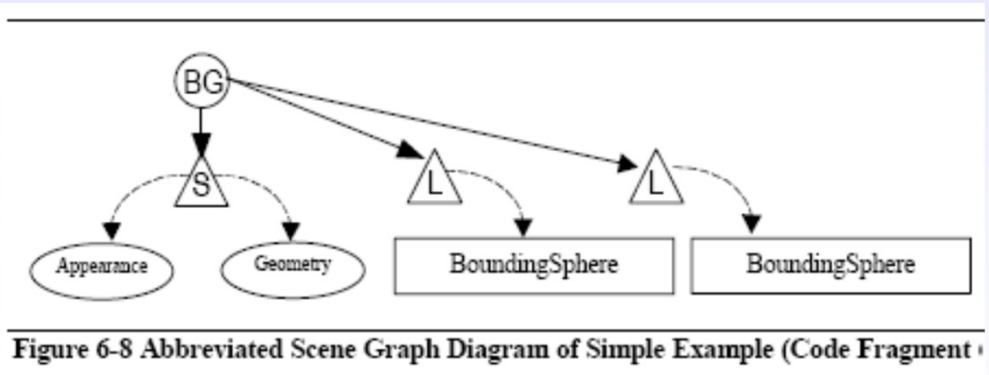
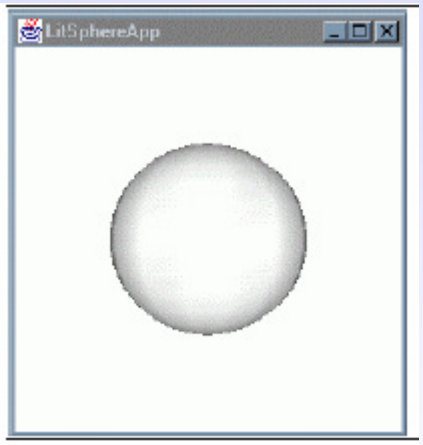


Figure 6-8 Abbreviated Scene Graph Diagram of Simple Example (Code Fragment



Realistic Lighting

- Realistic lighting can involve a bit of work…
  - Light scoping
    - Only enable lighting on objects that should be lit
  - Fake shadows

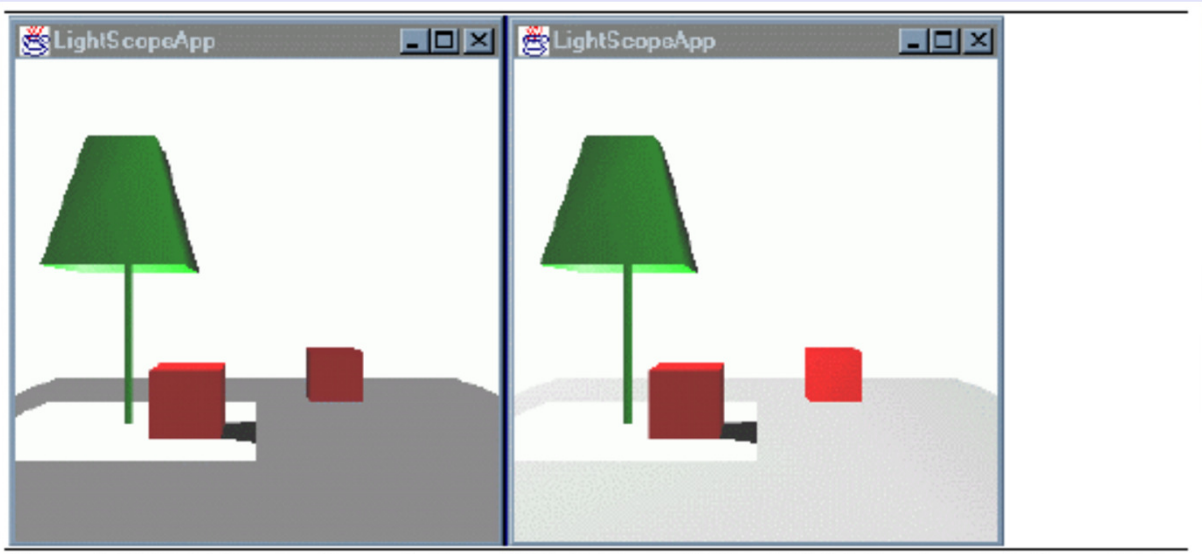


Figure 6-25 Light Scoping Example Scene With (left) and Without (right) Scoping of the Lamp Light.

(Java Tutorial Chapt. 6)

The End