LIGHTING

JEFWA NGOMBO NICHOLAS
ngombonj@gmail.com

content

Illumination

Lights

Lighting models

Illumination

- The transportation of luminous flux between points along direct or indirect paths

Lighting

- Computation of the light intensity reflected from a 3D point

Components of illumination

- Light sources
  - emission spectrum(color)
  - geometry(position ,direction)
  - direction attenuation
- Surface properties
  - Reflectance spectrum(color)
  - Geometry(position, orientation, microstructure)
  - Absorption
  - transmission

Ambient light model

- An indirect illumination model
- Independent of spatial and direction characteristics
- Amount of light incident on each object is a constant for all surfaces in the scene
- Amount of light reflected depends on the surface properties, independent of position and orientation

Directional light source model

Example the sun

Amount of light incident on each point of every surface in the scene depends on a common direction of the source of light for all the surfaces

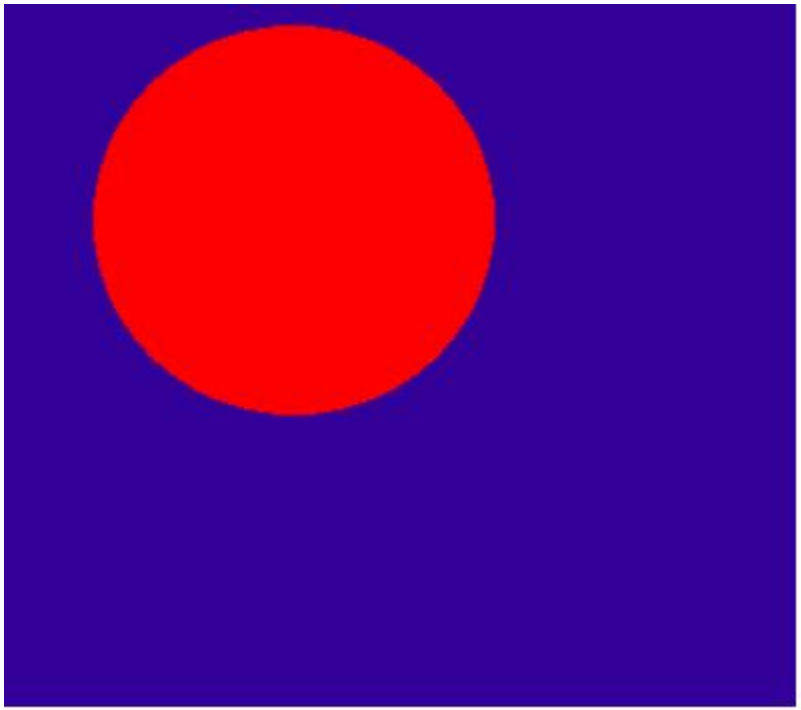
Point light source model

- Example local source such as bulb
- The direction of light changes for each point on the surface
- Amount of light incident on a point depends on the normalised vector to the light emitter
- Normal is used in the computation of color of a pixel as a function of light direction

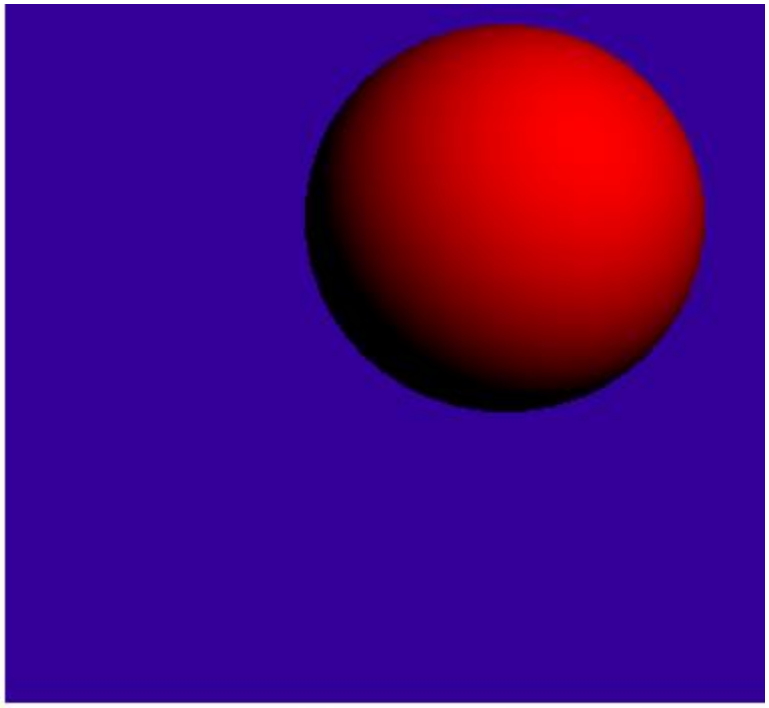
Normal vector

- Unit vector that is perpendicular to the surface
- Also called surface normal vector
- Normal is used in the computation of color of a pixel as a function of light direction
- It is used for shading-makes object look 3D

Normal vector



object color only



Diffuse Shading

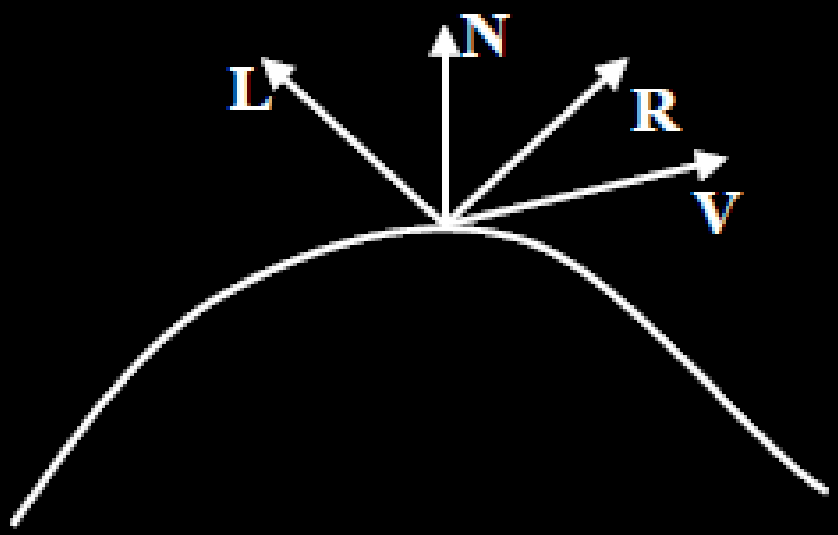
Surface normal vector is used for shading-makes object look 3D

Defining color of pixel in drawing

```
glBegin(GL_polygon);
Glcolor3f(1.0,0.0,0.0)
glNormal3f(0.0,0.0,1.0) //normal perpendicular
glVertex3f(2,3,4)
.
.
glEnd()
```

MODELS FOR COMPUTING LIGHT

L: direction to light
N: normal vector
R: reflection of light about normal
V: direction to viewer (i.e. reflection direction of interest)



POINT LIGHT

Specified by position(x,y,z)

Intensity(r,g,b)

Radiates eqully in all directions

$L=P_{light}-p_{surface}$

Directional light

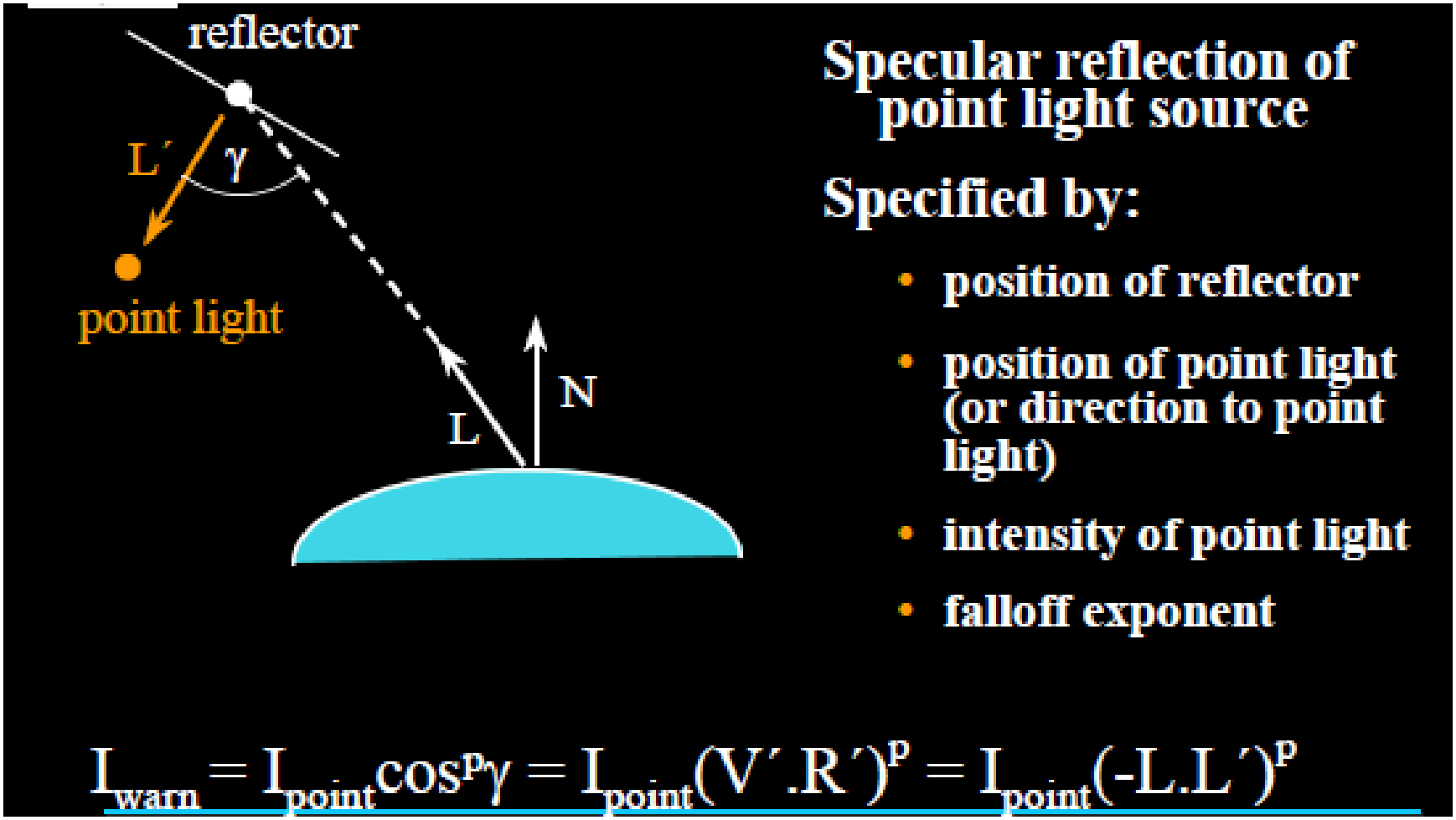
Specified by position(x,y,z)

Intensity(r,g,b)

All rays are parallel

L = -direction

Spot (or warn)light



Phong illumination model

Empirically divides reflection into 3 components

- **Ambient**
- **Diffuse (Lambertian)**
- **Specular**

Ambient component

Independent of location of viewer, location of light, and curvature of surface

$$I = I_a k_a$$

- I_a is intensity of ambient light
- k_a is ambient coefficient of surface

Note: this is a total hack, of course

Diffuse component

Component of reflection due to even scattering of light by uniform, rough surfaces

Depends on direction of light and surface normal

$$I_d = I_p(L.N)$$

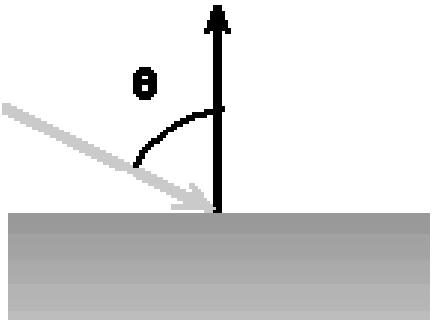
- **I_p is intensity of point light**

Computing diffuse reflection

- The angle between the surface normal and the incoming light ray is called the angle of incidence.
- I_{light} : intensity of the incoming light.
- k_d : represents the diffuse reflectivity of the surface at that wavelength.
- What is the range of k_d

$$I_{diffuse} = k_d I_{light} \cos\theta$$

$$I_{diffuse} = k_d I_{light} (\vec{n} \cdot \vec{l})$$



normal

Direction of light source

Diffuse component

When we write:

(N.L)

we often really mean:

max(N.L , 0)

- **The latter computes 1-sided lighting**
- **For 2-sided lighting, use:**

abs(N.L)

Specular component

Component of reflection due to mirror-like reflection off shiny surface

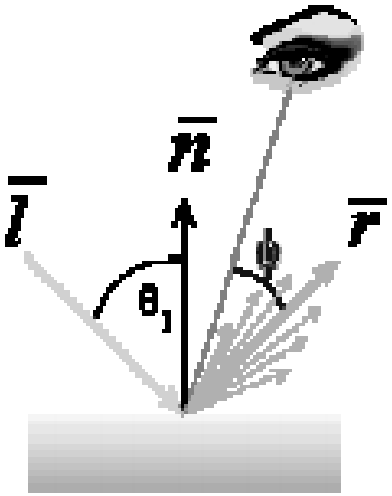
Depends on perfect reflection direction, viewer direction, and surface normal

$$I_s = I_p(R.V)^n$$

- **n is specular exponent, determining falloff rate**

Computing specular reflection

$$I_{specular} = k_s I_{light} \cos^{n_{shiny}} \boxed{\phi}$$



- The cosine term is maximum when the surface is viewed from the mirror direction and falls off to 0 when viewed at 90 degrees away from it. The scalar n_{shiny} controls the rate of this fall off.

light intensity reflected from a 3D point

Total I=sum of computed light components

=ambient+difuse+ specular

=

Illumunation with color

Surface reflection coefficients and light intensity may vary by wavelength

For RGB color

- **Light intensity specified for R, G, and B**
- **Surface reflection coefficients also for R, G, B**
- **Compute reflected color for R, G, and B**

Steps to add light to a scene

OpenGl steps to a light are as follows:

- Define normal vectors for each vertex for all objects
- Create, select and position one or more light source
- Create and select a light model
- Define the material properties for the objects in the scene

Define Normal Vectors for Each Vertex of Every Object

An object's normals determine its orientation relative to the light sources. For each vertex, OpenGL

uses the assigned normal to determine how much light that particular vertex receives from each

light source.

Create, Position, and Enable One or More Light Sources

light source location is specified by the **glLightfv() call. This**

example uses the default color for light zero (GL_LIGHT0), which is white; if you want a

differently colored light, use **glLight*() to indicate this**

After you've defined the characteristics of the lights you want, you have to turn them on with the

glEnable() command. You also need to call glEnable() with GL_LIGHTING as a parameter to

prepare OpenGL to perform lighting calculations

Select a Lighting Model

the **glLightModel*() command describes the parameters of a lighting model**

Define Material Properties for the Objects in the Scene

An object's material properties determine how it reflects light and therefore what material it seems to be made of

Light interacts with the material property of the object in light reflected to viewer

Material properties is specified with the **glMaterialfv() calls**

OpenGl Lighting code

```
void init(void){
//
GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };//define parameter,color model and intensity of light
GLfloat mat_shininess[] = { 50.0 };
GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };//define light position (x, y, z, w) position of light
glClearColor (0.0, 0.0, 0.0, 0.0);
glShadeModel (GL_SMOOTH);
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);//define material
glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);// define material
glLightfv(GL_LIGHT0, GL_POSITION, light_position);//define light
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_DEPTH_TEST);
}
```