CG – T9 – Illumination

L:CC, MI:ERSI

Miguel Tavares Coimbra

(course and slides designed by Verónica Costa Orvalho)

U.PORTO FC

How should we illuminate objects?

illumination

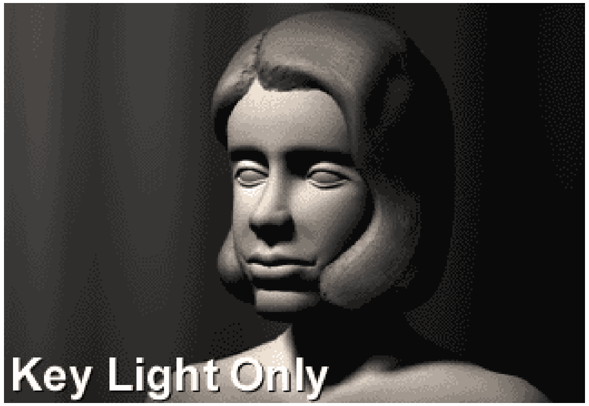
- key element to add realism to a scene
- how many lights we need to illuminate and object?

U.PORTO

http://www.iluminategames.com/vatan.htm

illumination: how to illuminate

A basic illumination technique requires at least 3 lights per object: key light, fill light, rim light



http://www.3drender.com/light/3point.html

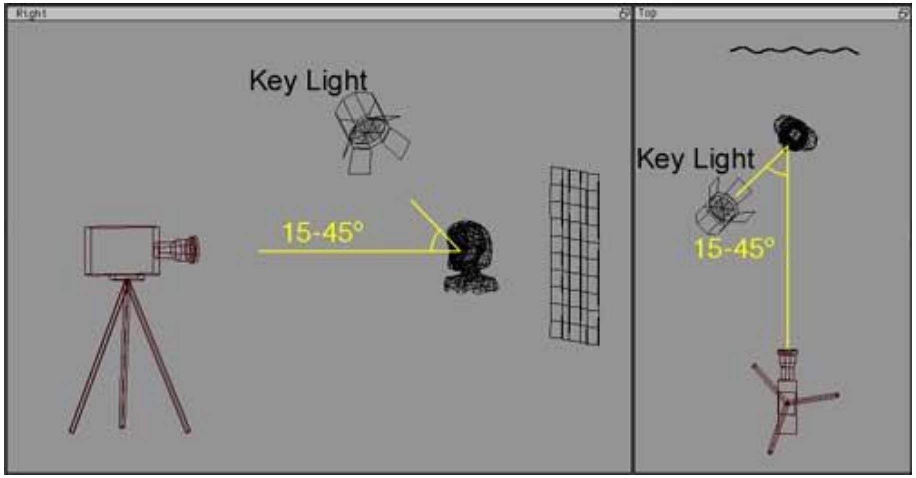
illumination: how to illuminate

key light:

. creates the subject main illumination

. defines the most visible lighting and shadows

. is the dominant light source (eg. sun,



representad by ambient light

http://www.3drender.com/light/3point.html

U. PORTO FC

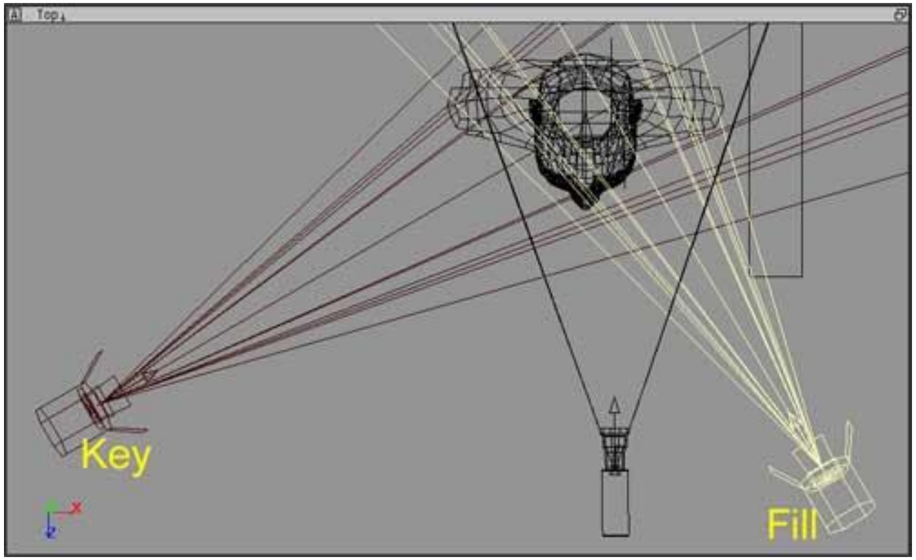
illumination: how to illuminate

fill light:

. softens and extends the illumination of fill light

. makes more of the subject visible

. are secondary light sources (eg. table lamp...)



U.PORTO FC

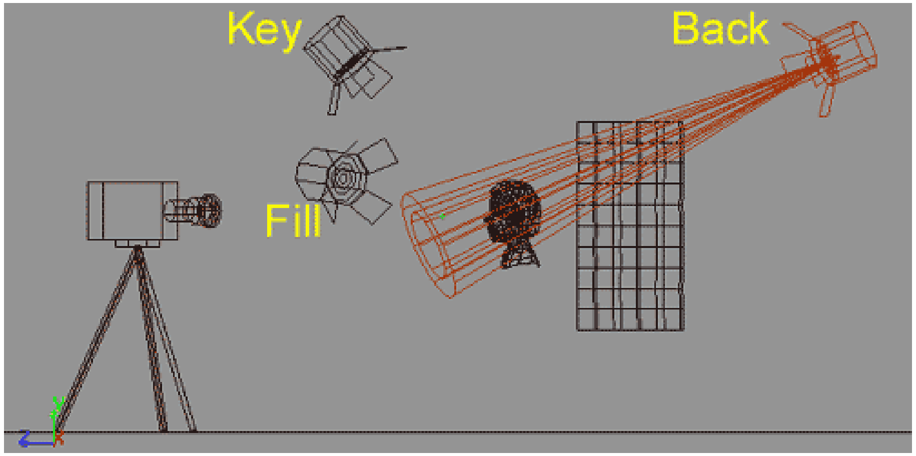
representad by spot or point light

illumination: how to illuminate

rim light (or back light):

. creates a bright line around the edge of the object

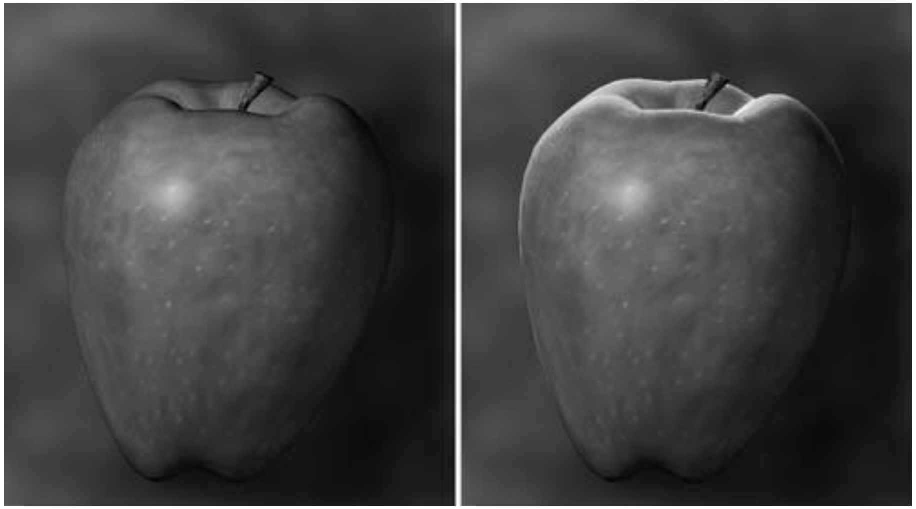
. helps visually separates the object from the background



representad by directional light

http://www.3drender.com/light/3point.html

illumination: how to illuminate



no back light (left), back light added (right)

reference book: Jeremy Birn, “Digital Lighting & Rendering”, Second Edition, New Riders, April 27, 2006

http://www.3drender.com/light/3point.html

illumination

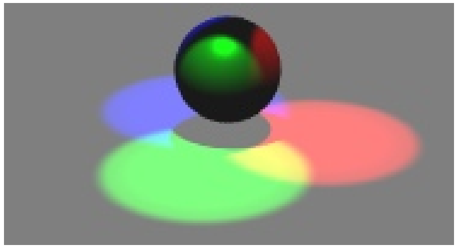
elements that influence the illumination computation:

. type of light, position and direction

. light component (ambient, diffuse, specular)

. vertices normal

. object material



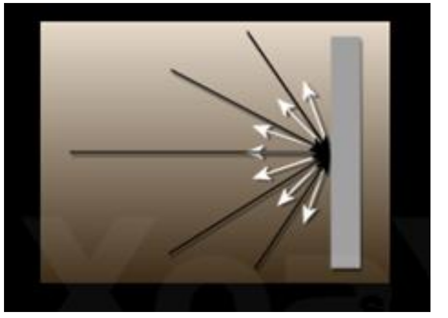
. additional object colors

some basics you MUST know

Types of Lights

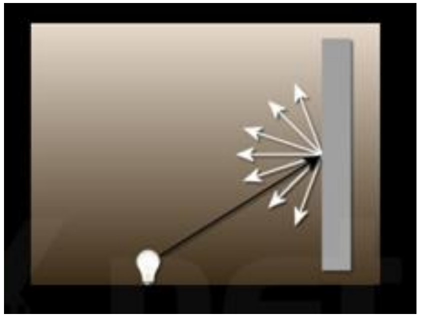
Ambient:

No source point; affects all polys independent of position, orientation and viewing angle; used as a 'fudge' to approximate 2nd order and higher reflections



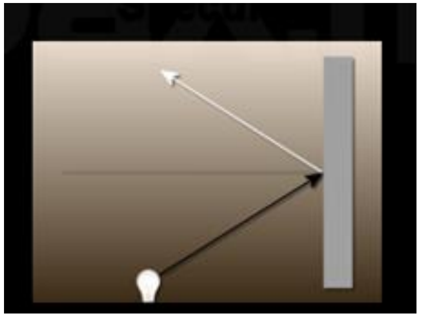
Diffuse:

Light scattered in all directions after it hits a poly; dependant upon incident angle



Specular:

'Shininess' ; dependant upon incident and viewing angles



U. PORTO FC

Images: http://xoax.net/comp/sci/graphics3D/BasLocIll.php

some basics you MUST know

Types of Lights

1. **Ambient**
2. **Diffuse**
3. **Specular**

4. Emissive: color of a surface adds intensity to the object, but is unaffected by any light sources. Does not introduce any additional light into the overall scene.

Images: http://xoax.net/comp/sci/graphics3D/BasLocIll.php

Ambient Light

ambient light

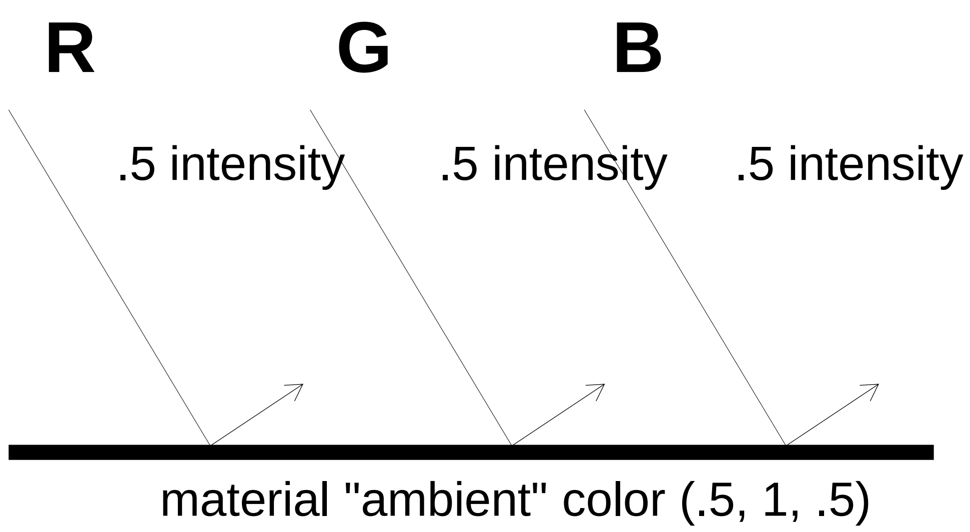
. light that doesn't come from any direction

. objects are evenly lit on all surfaces in all directions

ambient light

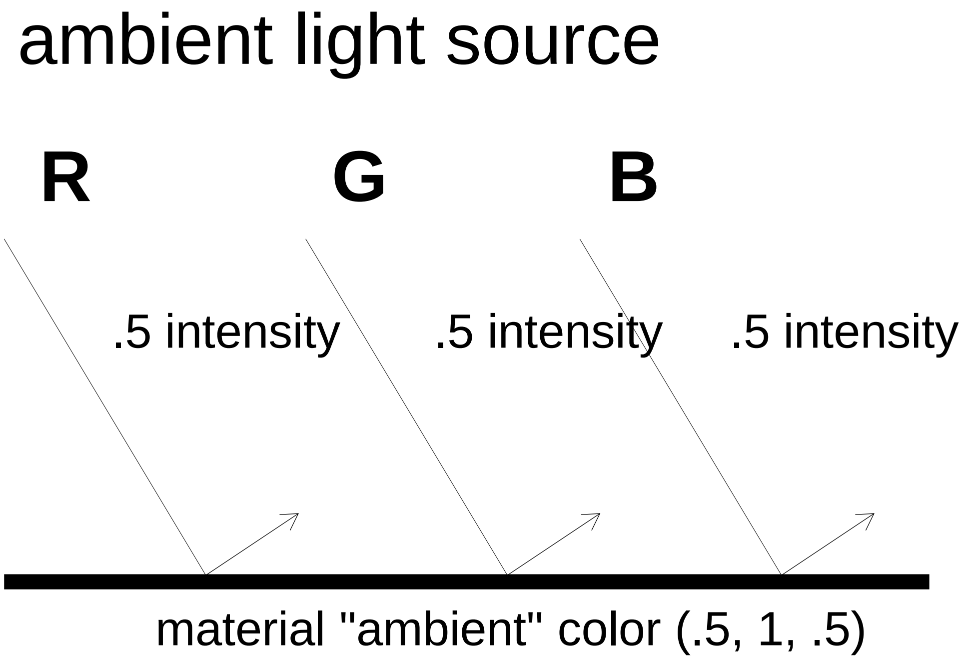
. has a source, but rays of light bounce around the scene and become directionless

ambient light source



ambient light

. has a source, but rays of light bounce around the scene and become directionless

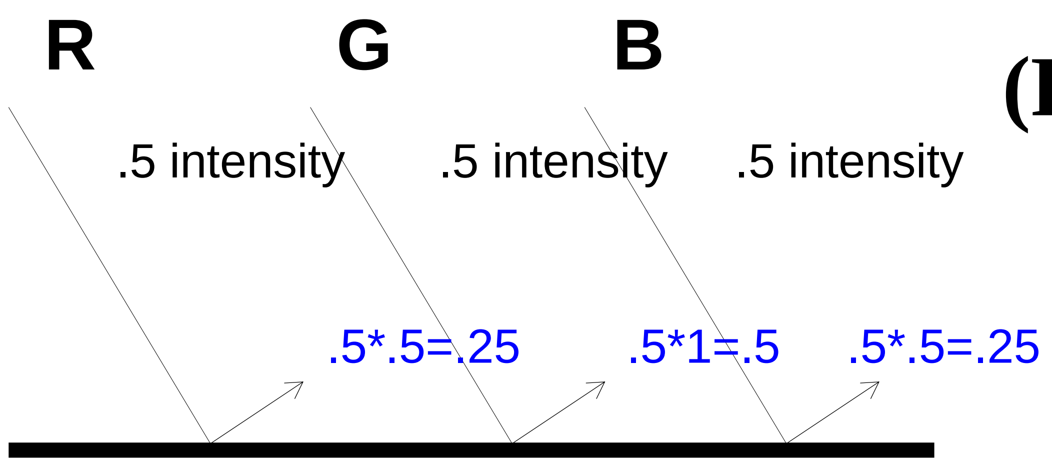


how do you calculate the ambient color component of an object?

ambient light

. has a source, but rays of light bounce around the scene and become directionless

ambient light source



material "ambient" color (.5, 1, .5)

color vector (R,G,B) = (.25, .5, .25)

Diffuse illumination

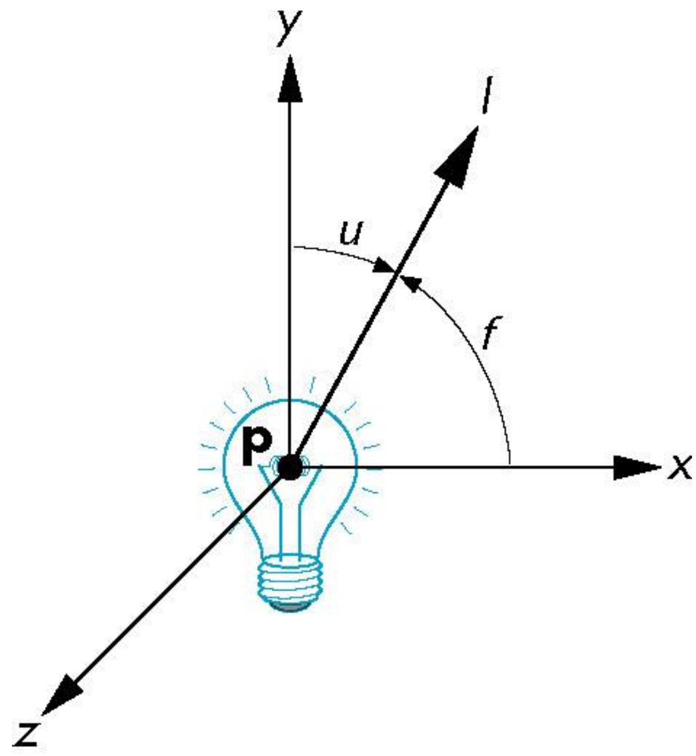
how can we create a light model?

$$I(x,y,z,\theta,\phi,\lambda)$$

. *(x,y,z): light source*

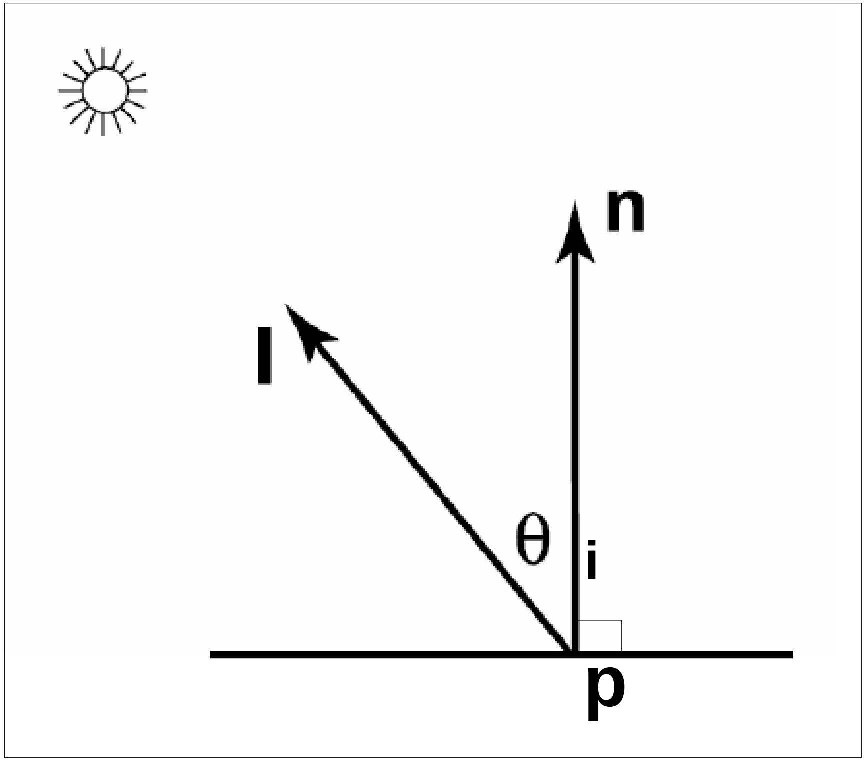
. *(θ,ϕ): emition direction*

. λ : light intensity



how can we create a light model?

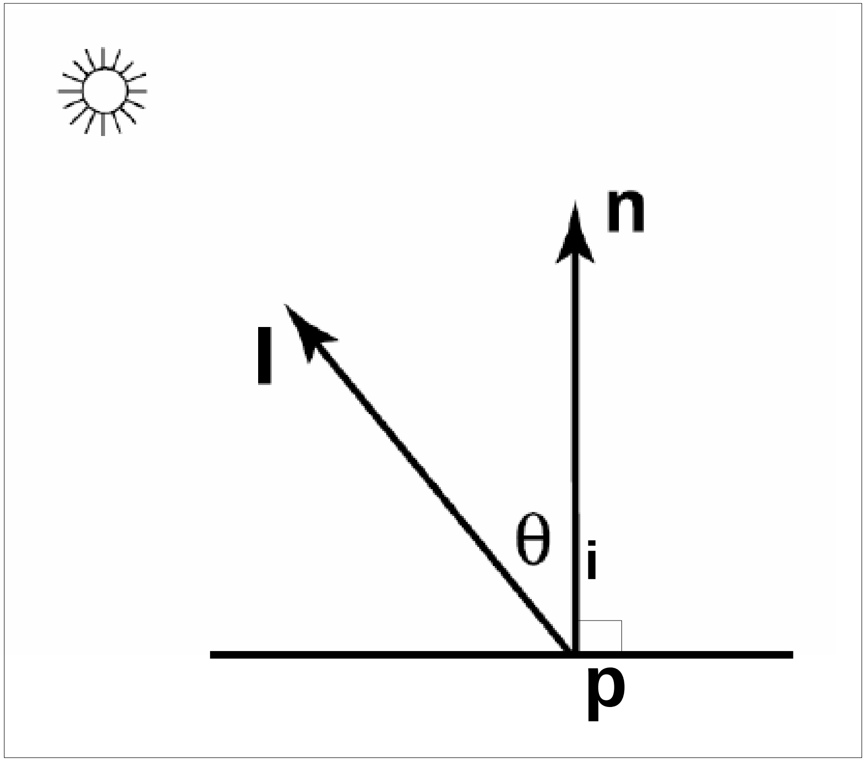
measuring **irradiance** at a **plane perpendicular** to **l** tells us how **bright** the light is in general



how can we create a light model?

measuring **irradiance** at a **plane perpendicular** to **l** tells us how **bright** the light is in general

how we calculate the illumination on a surface ?



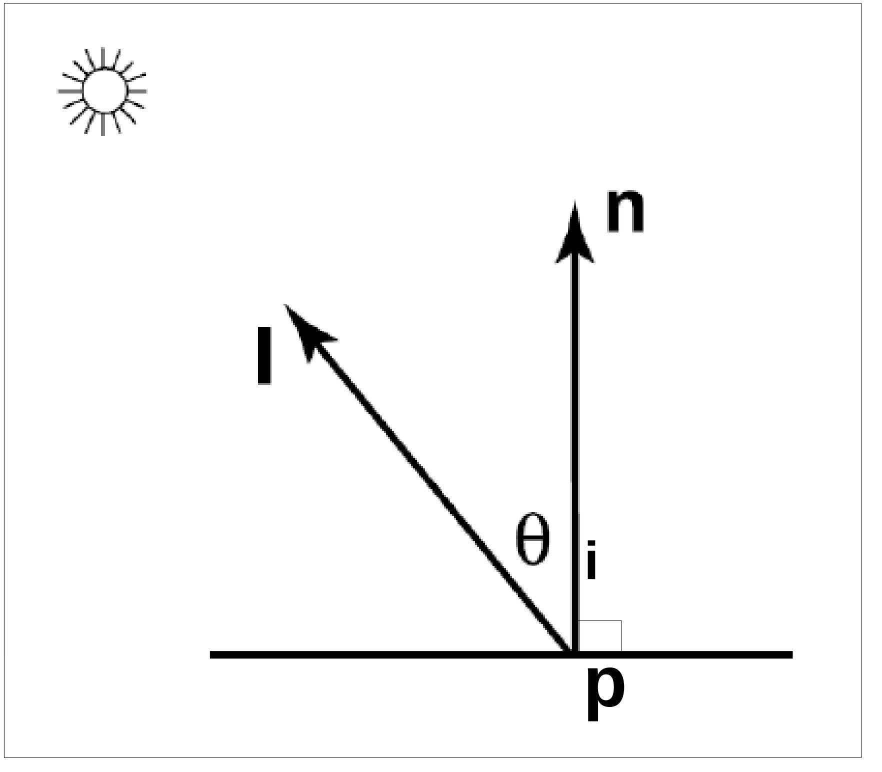
how can we create a light model?

measuring **<u>irradiance</u>** at a **<u>plane perpendicular</u>** to **l** tells us how **<u>bright</u>** the light is in general

how we calculate the illumination on a surface ?

meassure irradiance to a **<u>plane parallel</u>** to the surface.

(perpendicular to **n)**



U. PORTO FC

how can we create a light model?

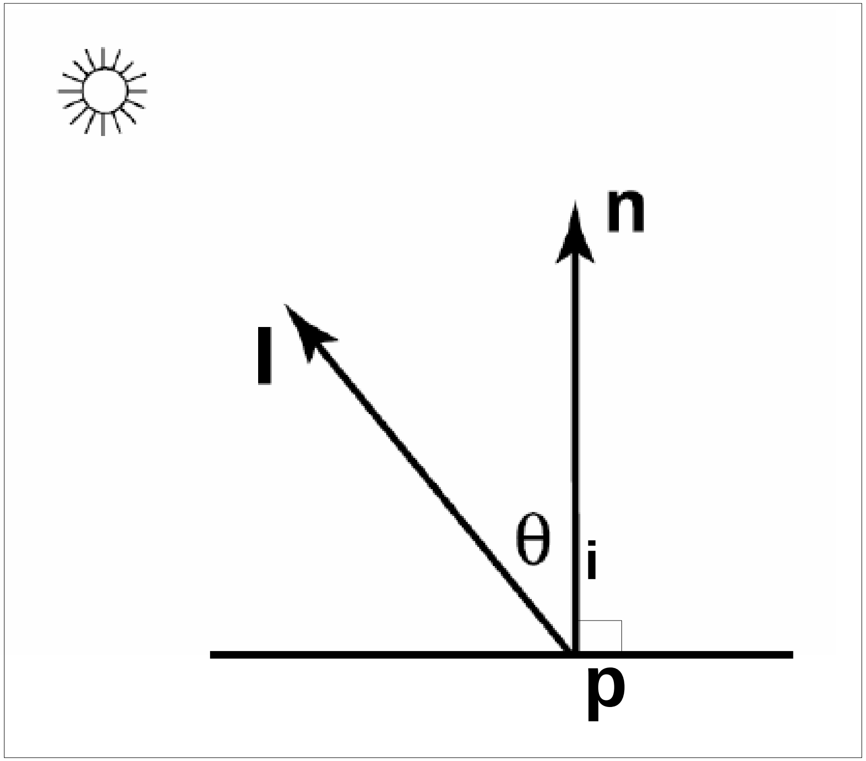
measuring **irradiance** at a **plane perpendicular** to **l** tells us how **bright** the light is in general

how we calculate the illumination on a surface ?

meassure irradiance to a **plane parallel** to the surface.

(perpendicular to **n)**

surface irradiance equal to the irradiance meassured perpendicular to **l * cosine** θ *(between **n** and **l**)*



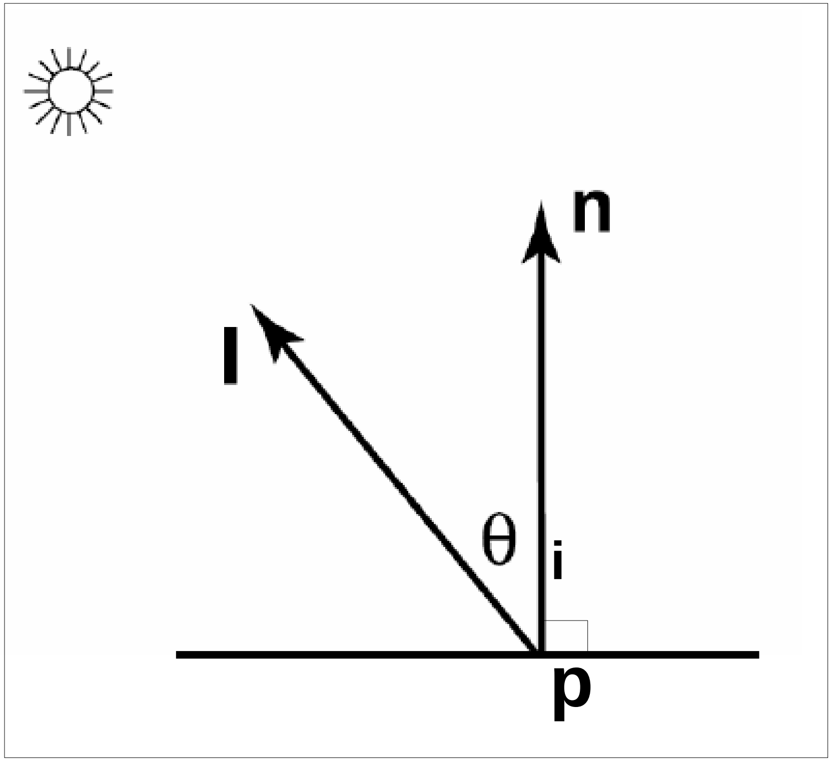
how can we create a light model?

measuring **irradiance** at a **plane perpendicular** to **l** tells us how **bright** the light is in general

how we calculate the illumination on a surface ?

$$E = E_L \cos \theta$$

E_L irradiance perpendicular to **l**



how can we create a light model?

measuring **irradiance** at a **plane perpendicular** to **l** tells us how **bright** the light is in general

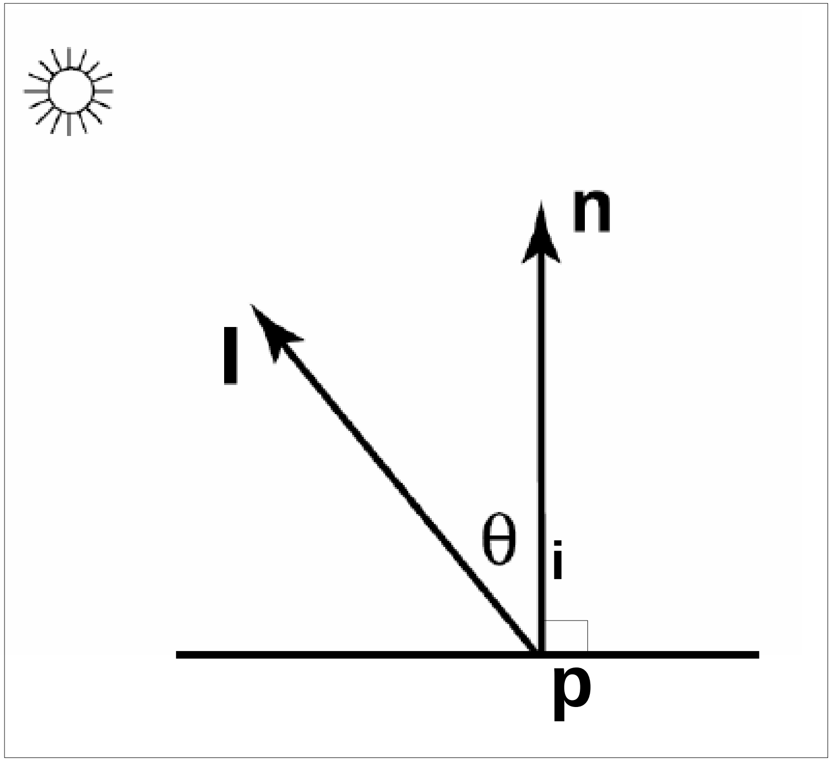
how we calculate the illumination on a surface ?

$$E = E_L \cos\theta$$

E_L irradiance perpendicular to **l**

$\cos\theta = \mathbf{l} \cdot \mathbf{n}$ (dot product)

$$E = E_L(\mathbf{l} \cdot \mathbf{n})$$



U. PORTO FC

how can we create a light model?

measuring **irradiance** at a **plane perpendicular** to **l** tells us how **bright** the light is in general

how we calculate the illumination on a surface ?

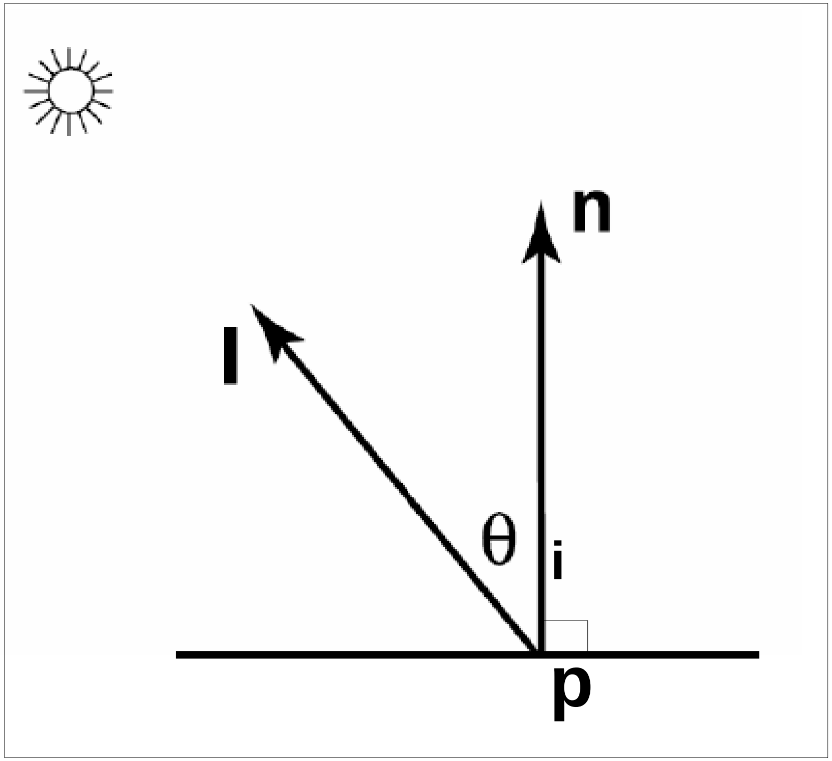
$$E = E_L \cos\theta$$

E_L irradiance perpendicular to **l**

$\cos\theta = \mathbf{l} \cdot \mathbf{n}$ (dot product)

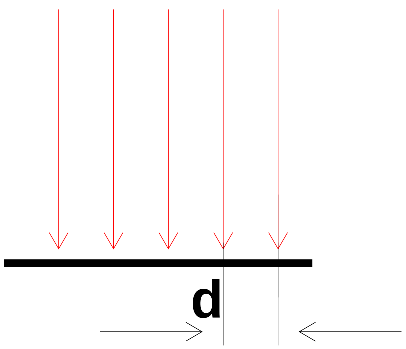
$E_k = E_L(\mathbf{l}\cdot\mathbf{n})$ $E = \sum E_k$

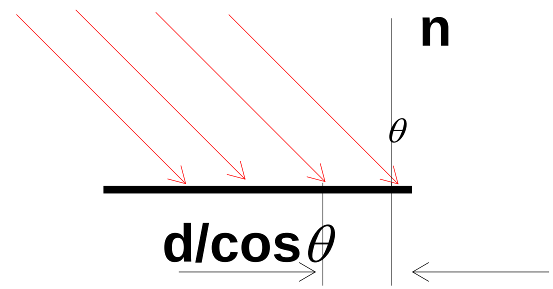
k = 1… n, where n are the lights in the scene



how can we create a light model?

meassures the density of the rays





Irradiance is **proportional** to the **density** of the rays

Inversely proportional to the distance **d** between the rays

Since **irradiance** is inversely proportional to the distance **d** it is **proportional** to **cos** θ

Lambert Shading Model

$$\mathbf{c} \propto \cos(\theta)$$
$$\boldsymbol{c} \propto \boldsymbol{n} \cdot \boldsymbol{l}$$

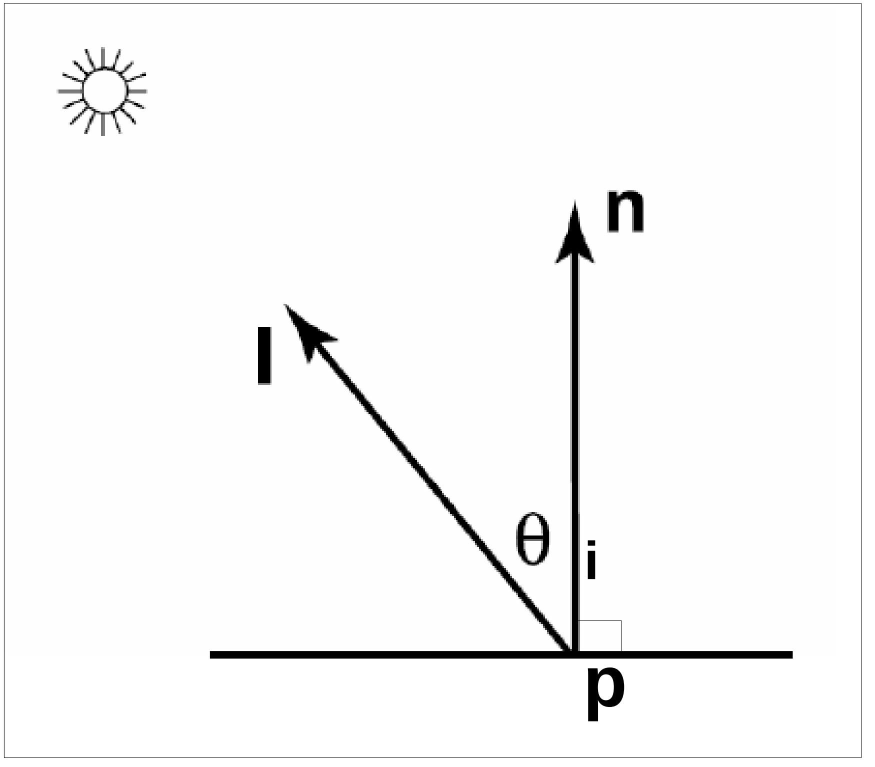
c: color vector

n: surface normal

l: direction to light

θ: light/normal angle

$$\cos(\theta) = \frac{n \cdot l}{|n| \cdot |l|}$$



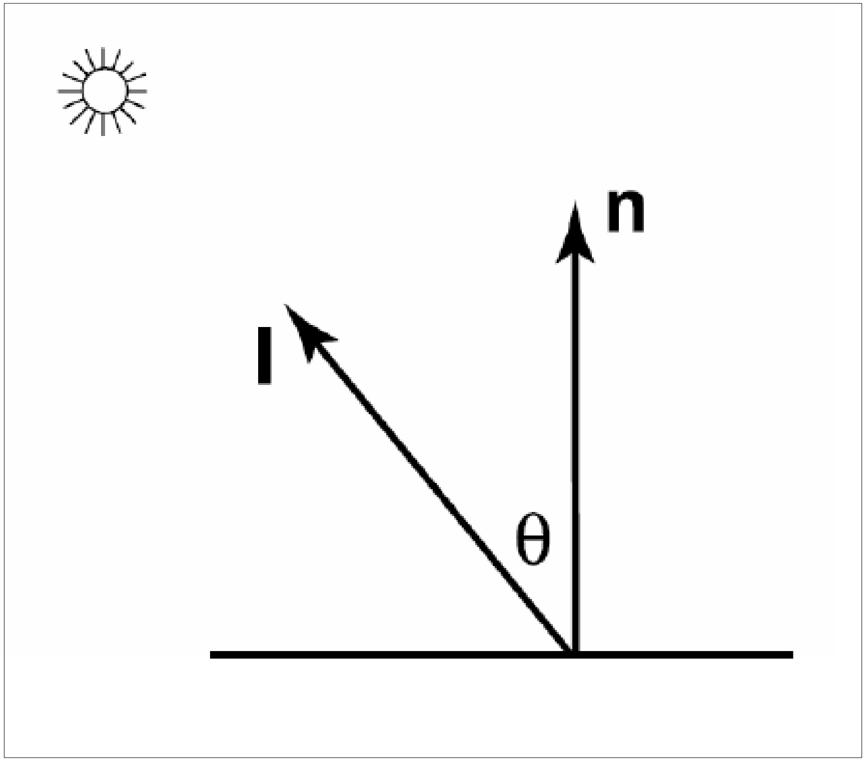
Lambert's law

$$I_{diffuse} = k_d \; I_{light} \cos(\theta) = k_d \; I_{light} \, \boldsymbol{n} \cdot \boldsymbol{l}$$

I_{light}: light source intensity

k : surface reflectance coefficient in [0,1]

θ: light/normal angle



Lambert's law

$$I_{diffuse} = k_d \; I_{light} \cos(\theta)$$
$$= k_d \; I_{light} \; \boldsymbol{n \cdot l}$$

How would you change
this equation to support
more than one light**?**

Lambert's law

$$I_{diffuse} = k_d \ I_{light} \cos(\theta)$$
$$= k_d \ I_{light} \, \boldsymbol{n \cdot l}$$

How would you change this equation to support more than one light**?**

$$I_{diffuse} = \sum \ k_d \ I_{light} \, \boldsymbol{n \cdot l}$$

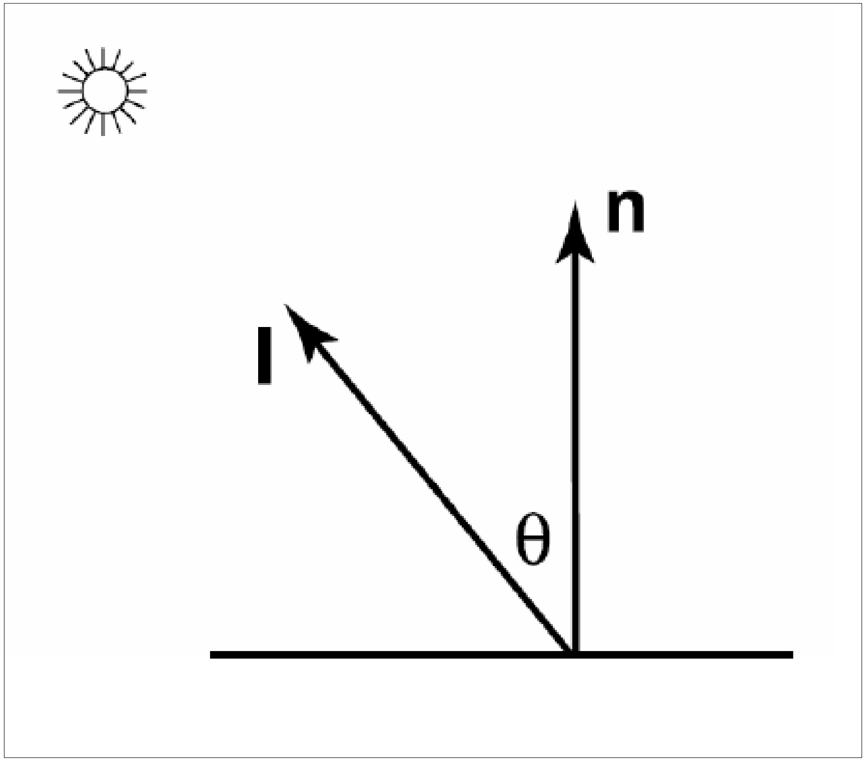
How to calculate all these normals?

how to calculate the **normal**

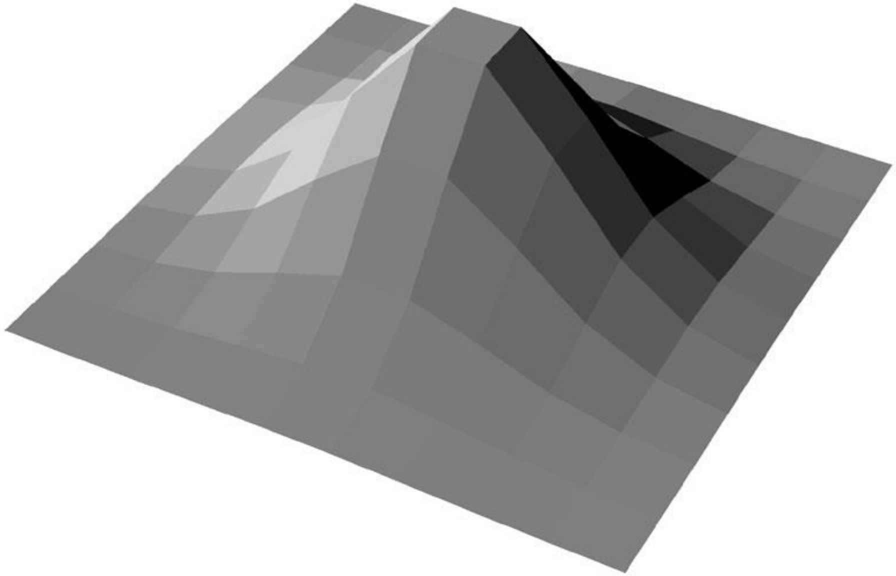
$$a.x + b.y + c.z + d = 0$$

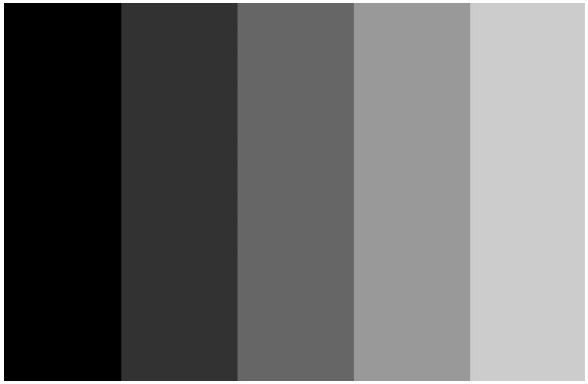
$$\mathbf{n} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

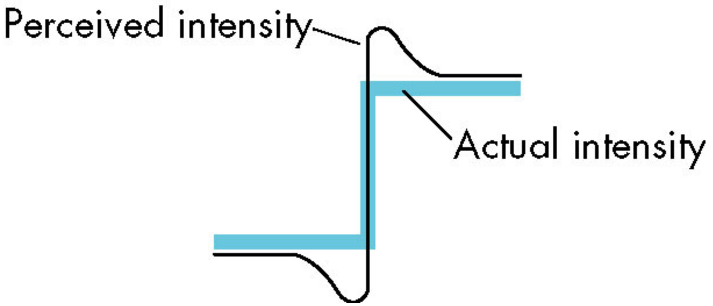
$$\mathbf{n} = (p_2 - p_1) \times (p_1 - p_0)$$



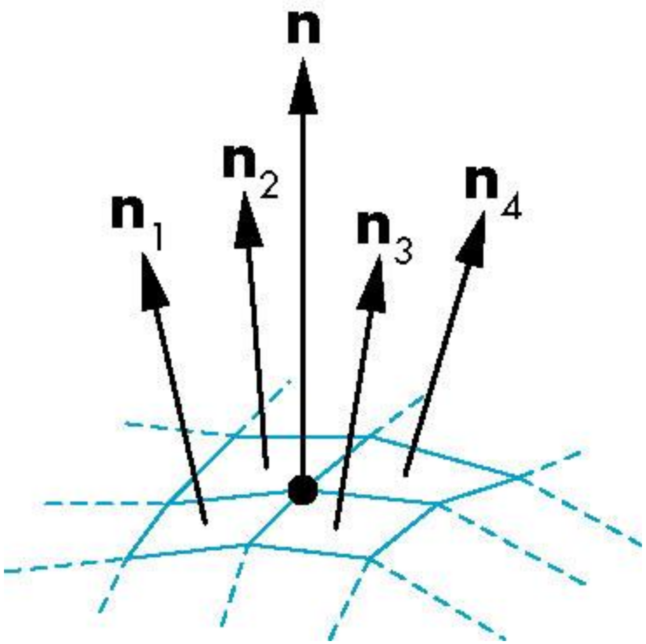
flat shading (ambient)







gouraud (smooth) shading

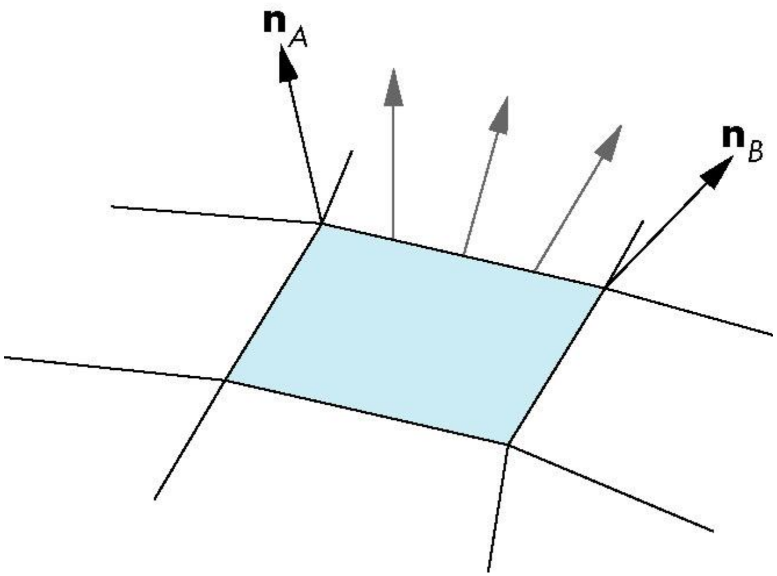


$$\mathbf{n} = \frac{\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4}{\left|\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4\right|}$$

– In OpenGL: `glShadeModel(GL_SMOOTH)`

phong (smooth) shading

1. calculate the normals on the side of the polygons by interpolating the vertex normals

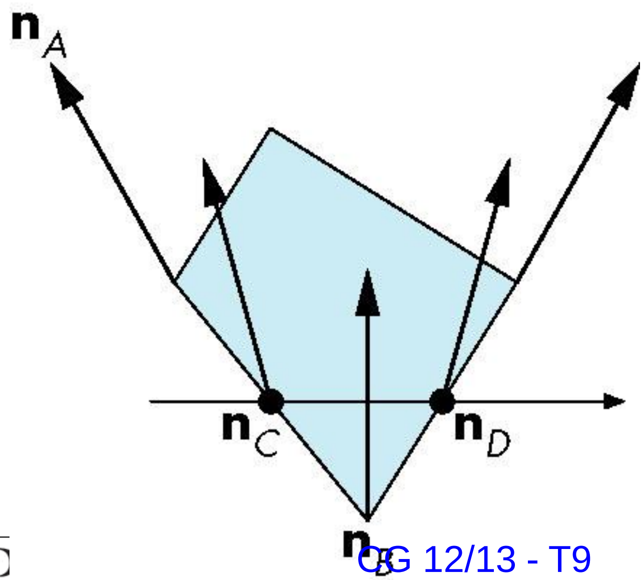


$$\mathbf{n}(\alpha) = (1-\alpha)\mathbf{n}_A + \alpha\mathbf{n}_B$$

phong (smooth) shading

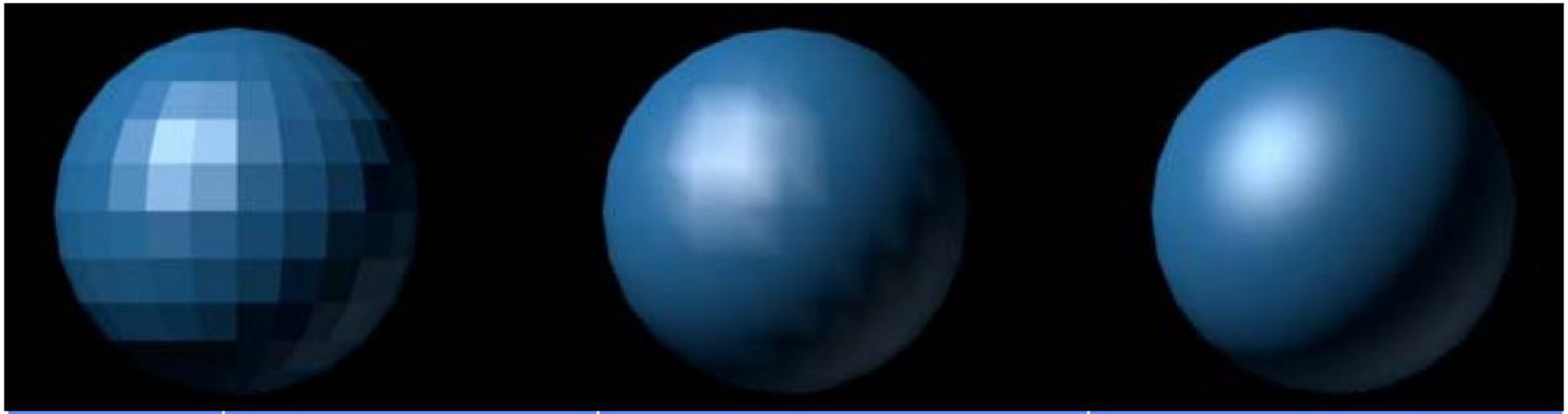
1. calculate the normals on the side of the polygons by interpolating the vertex normals

2. calculate the normals inside the polygon



$$\mathbf{n}(\alpha, \beta) = (1 - \beta)\mathbf{n}_C + \beta\mathbf{n}_D$$

Flat / Gouraud / Phong Shading

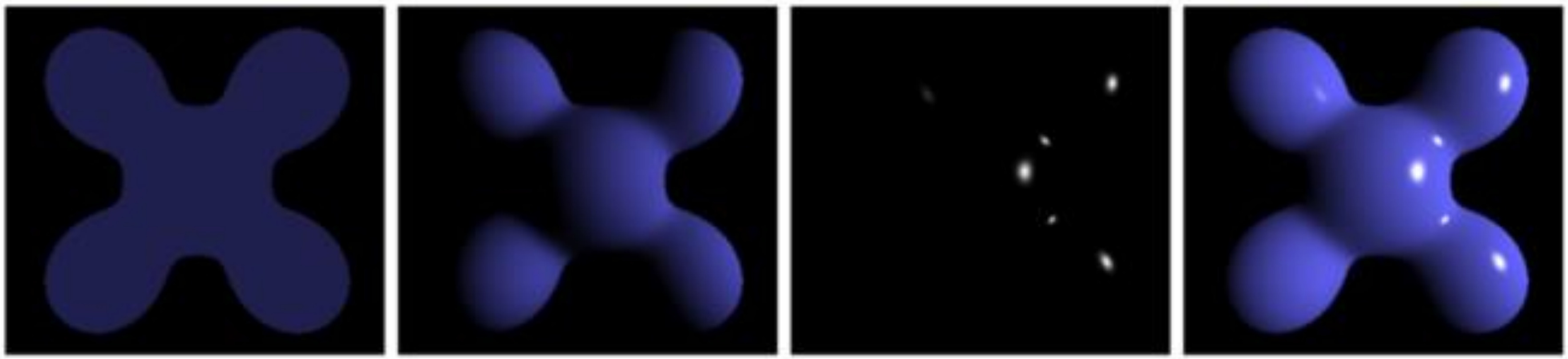


	Flat Shading	Gouraud Shading	Phong Shading*
Normal	Split; same for each triangle's three vertices	Each vertex has a normal which is used to compute a per vertex color	Interpolated to each fragment
Color	One color value computed for each triangle	Interpolated to each fragment	Each fragment has a normal which is used to compute a per fragment color

Slide by Ron Fedkiw, Stanford University

Illumination: components

phong



ambient + diffuse + specular = phong

reflection

other: blinn-phong, lambert, gouraud,...

In a nutshell

- Calculate each primary color separately
- Start with global ambient light
- Add reflections from each light source
- Clamp to [0, 1]

Summary

- Three main types of light:
  - Ambient, Diffuse, Specular
- Illumination on a surface depends on the irradiance angle with the normal
  - Lambert shading model
- How can we calculate these normals?
  - Flat shading, Gouraud shading, Phong shading