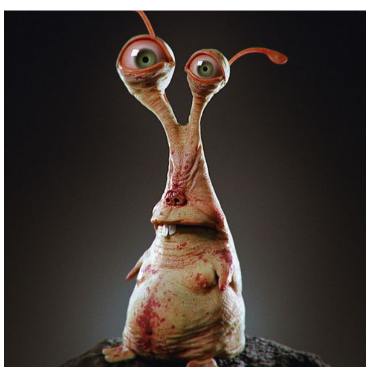
Shading

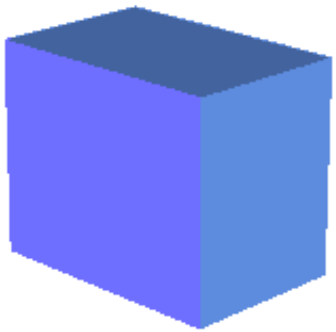
3D Computer Graphics (Lab 6)



Introduction

- Rendering is the process of generating a 2D image from a 3D object.
- Which technique did we implement to render a 3D object? *ray tracing*

Rendering without a shading model



Rendering with a shading model

- A shading model attempts to model how light from light sources interacts with 3D objects in a scene.
  - It is computationally unfeasible to simulate all of the physics behind the interaction between light and 3D objects.
  - A number of approximate models have been invented which do a good job and produce various levels of realism.

Initial assumptions

1) There is only one light source.

2) The light source is a point.

3) The light source has an intensity I_s which takes on values between 0 and 1

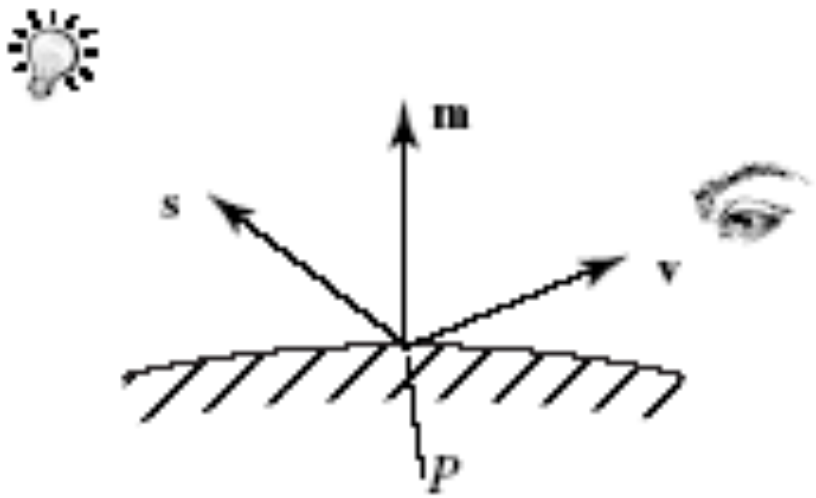
If $I_s = 0$ then there is no light

If $I_s = 1$ then there is white light

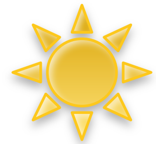
Interaction of light and a 3D object

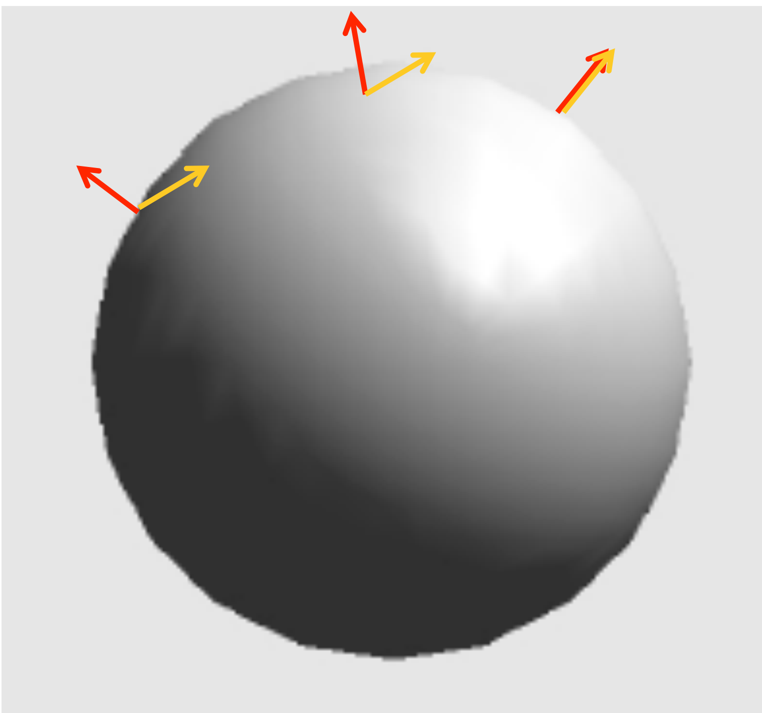
We consider three vectors to compute the interaction of light at a point P on the surface of a 3D object:

- m the normal vector to the surface at P
- s the vector from P to the light source
- v the vector from P to the eye (camera)



Lighting calculations





vector s pointing to the light source

normal vector m to the surface

The larger the angle α between s and m at a point on the surface, the less light this point receives from the light source.

The amount of light illuminating the surface is given by

$$I_s \cos\alpha = I_s \frac{s.m}{|s||m|}$$

Lambert's law

- If α=0, the amount of light illuminating P = I_s
- If α increases, cos α decreases and hence the amount of light illuminating the surface decreases as well.

Initial assumptions

1) There is only one light source.

2) The light source is a point.

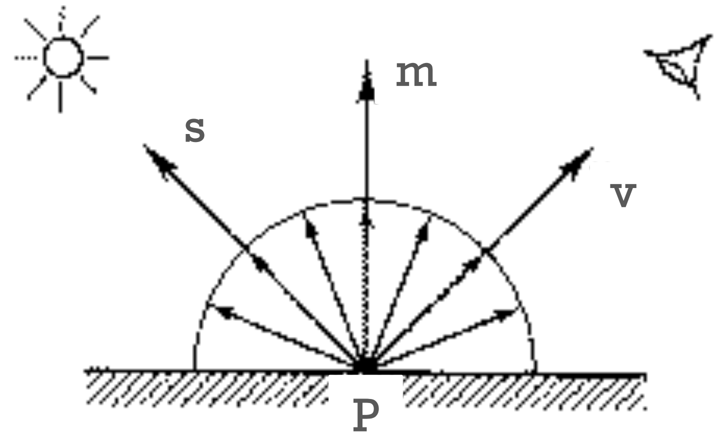
3) The light source has an intensity I_s which takes on values between 0 and 1

   If $I_s = 0$ then there is no light
   If $I_s = 1$ then there is white light

4) There is only 1 way in which light interacts with a 3D object: diffuse reflection

Diffuse reflection

Diffuse reflection means that light is reflected uniformly in all directions.



How much light is reflected?

This depends on the material the object is made of.

We define the diffuse reflection coefficient ρ_d as a number between 0 and 1 which specifies how much of the incoming light is diffusely reflected.

The amount of light diffusely reflected in any direction:

if $s.m > 0$ $\quad I = \rho_d I_s \cos\alpha = \rho_d I_s \dfrac{s.m}{|s||m|}$

else $\quad I = 0$

Diffuse reflection

and hence, in the direction to the eye

The amount of light diffusely reflected in any direction:

if $s.m > 0$ $\quad I = \rho_d I_s \dfrac{s.m}{|s||m|}$

else $\quad I = 0$

Example

Spheres with diffuse reflection.

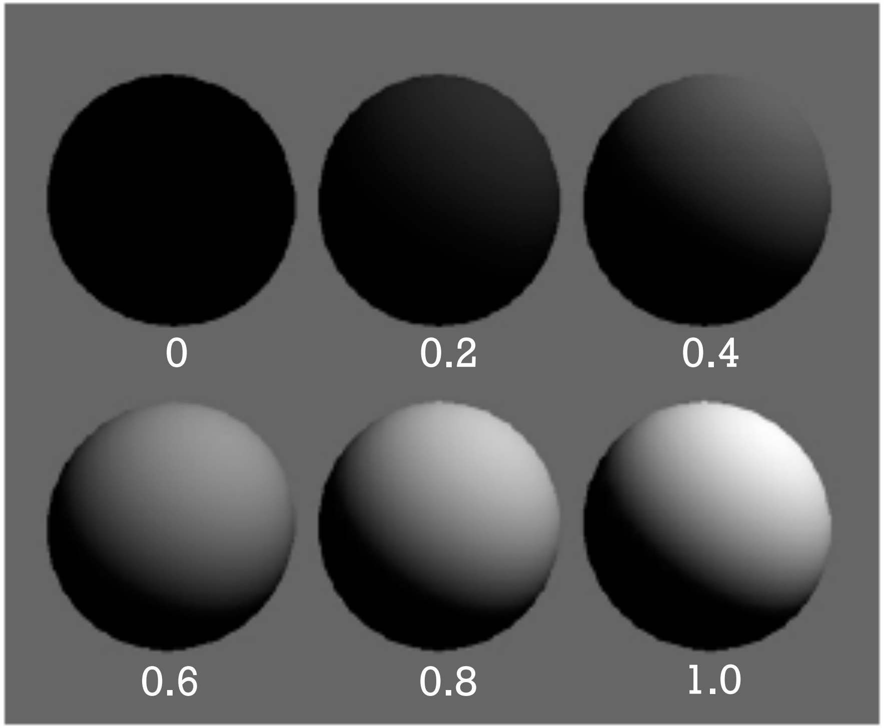
$I_s = 1$

ρ_d ranges from 0 to 1

(In practice, ρ_d depends on the material the object is made of.)

Shortcoming of this shading model?

Shadows are seen to be unrealistically deep and harsh.



Shading model

The amount of light that reaches the eye from the point P:

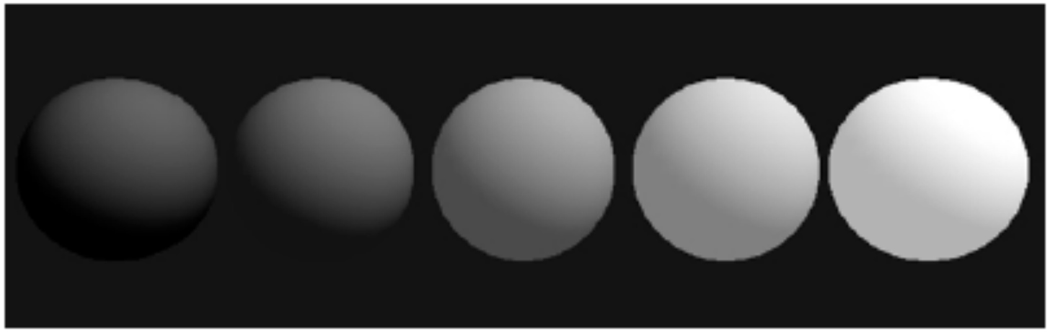
if $s.m > 0$ $\quad I = \boxed{\rho_d I_s \frac{s.m}{|s||m|}} \boxed{+\rho_a I_s}$

else $\quad I = \boxed{0} \quad \boxed{+\rho_a I_s}$

diffuse ambient

The ambient reflection coefficient ρ_a is a number between 0 and 1 which specifies how much of the intensity of the light source is always added.

Example ρ_a 0 0.1 0.3 0.5 0.7



Modest ambient light softens shadows.

Too much ambient light washes out shadows.

Adding colour

- It is straightforward to extend the shading model to the case of coloured light reflecting from coloured surfaces.

Simply compute the red, green and blue components of reflected light:

if $s.m > 0$

$$I_r = \rho_{dr} I_{sr} \frac{s.m}{|s||m|} + \rho_{ar} I_{sr}$$

$$I_g = \rho_{dg} I_{sg} \frac{s.m}{|s||m|} + \rho_{ag} I_{sg}$$

$$I_b = \rho_{db} I_{sb} \frac{s.m}{|s||m|} + \rho_{ab} I_{sb}$$

else

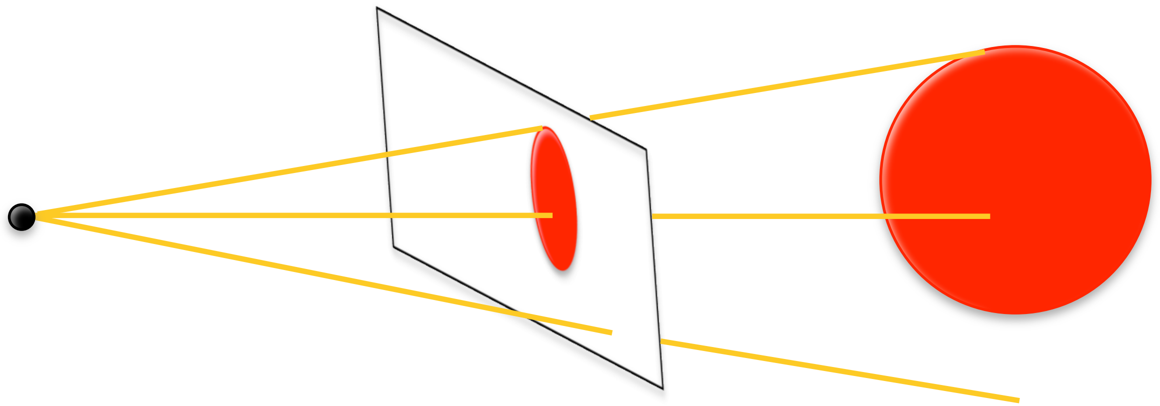
$$I_r = \rho_{ar} I_{sr}$$

$$I_g = \rho_{ag} I_{sg}$$

$$I_b = \rho_{ab} I_{sb}$$

Remember …

Ray tracing



For each pixel

1. Create a ray from the eye of the camera through this pixel.
2. Compute the intersection of this ray with the 3D object.

The colour of the intersection point determines the colour of the pixel.

How did we determine the colour of the intersection point?

By simply looking at the colour stored in the Material object of the hit shape.

This colour will now be computed by the shading model.

Final shading model

if $s.m > 0$

$$\boxed{I_r} = \rho_{dr} I_{sr} \frac{s.m}{|s||m|} + \rho_{ar} I_{sr}$$

$$\boxed{I_g} = \rho_{dg} I_{sg} \frac{s.m}{|s||m|} + \rho_{ag} I_{sg}$$

$$\boxed{I_b} = \rho_{db} I_{sb} \frac{s.m}{|s||m|} + \rho_{ab} I_{sb}$$

else

$$I_r = \rho_{ar} I_{sr}$$

$$I_g = \rho_{ag} I_{sg}$$

$$I_b = \rho_{ab} I_{sb}$$

The colour of the hitPoint

What information does the shading model need to compute the colour of the intersection point?

Final shading model

if $s.m > 0$

$$I_r = \rho_{dr} I_{sr} \boxed{\frac{s.m}{|s||m|}} + \rho_{ar} I_{sr}$$

$$I_g = \rho_{dg} I_{sg} \boxed{\frac{s.m}{|s||m|}} + \rho_{ag} I_{sg}$$

$$I_b = \rho_{db} I_{sb} \boxed{\frac{s.m}{|s||m|}} + \rho_{ab} I_{sb}$$

else

$$I_r = \rho_{ar} I_{sr}$$

$$I_g = \rho_{ag} I_{sg}$$

$$I_b = \rho_{ab} I_{sb}$$

s can be computed based on the coordinates of the hitPoint and the coordinates of the light source.

m is the hitNormal stored in the Intersection object

Final shading model

if $s.m > 0$

$$I_r = \rho_{dr} \boxed{I_{sr}} \frac{s.m}{|s||m|} + \rho_{ar} \boxed{I_{sr}}$$

$$I_g = \rho_{dg} \boxed{I_{sg}} \frac{s.m}{|s||m|} + \rho_{ag} \boxed{I_{sg}}$$

$$I_b = \rho_{db} \boxed{I_{sb}} \frac{s.m}{|s||m|} + \rho_{ab} \boxed{I_{sb}}$$

else

$$I_r = \rho_{ar} I_{sr}$$

$$I_g = \rho_{ag} I_{sg}$$

$$I_b = \rho_{ab} I_{sb}$$

Colour of the light source

Final shading model

if $s.m > 0$

$$I_r = \boxed{\rho_{dr}} I_{sr} \frac{s.m}{|s||m|} + \boxed{\rho_{ar}} I_{sr}$$

$$I_g = \boxed{\rho_{dg}} I_{sg} \frac{s.m}{|s||m|} + \boxed{\rho_{ag}} I_{sg}$$

$$I_b = \boxed{\rho_{db}} I_{sb} \frac{s.m}{|s||m|} + \boxed{\rho_{ab}} I_{sb}$$

else

$$I_r = \rho_{ar} I_{sr}$$

$$I_g = \rho_{ag} I_{sg}$$

$$I_b = \rho_{ab} I_{sb}$$

Diffuse reflection coefficients

Ambient reflection coefficients

These numbers specify material properties of the 3D object to be rendered.

Final shading model

if $s.m > 0$

$$I_r = \rho_{dr} I_{sr} \frac{s.m}{|s||m|} + \rho_{ar} I_{sr}$$

$$I_g = \rho_{dg} I_{sg} \frac{s.m}{|s||m|} + \rho_{ag} I_{sg}$$

$$I_b = \rho_{db} I_{sb} \frac{s.m}{|s||m|} + \rho_{ab} I_{sb}$$

else

$$I_r = \rho_{ar} I_{sr}$$

$$I_g = \rho_{ag} I_{sg}$$

$$I_b = \rho_{ab} I_{sb}$$

Note: this shading model computes the colour of the hitPoint in the presence of one light source.

What if there are multiple light sources?

Compute the colour of the hitPoint for each light source and simply add these colours together.

Shading polygonal meshes

Final shading model

if $s.m > 0$

$$I_r = \rho_{dr} I_{sr} \boxed{\frac{s.m}{|s||m|}} + \rho_{ar} I_{sr}$$

$$I_g = \rho_{dg} I_{sg} \boxed{\frac{s.m}{|s||m|}} + \rho_{ag} I_{sg}$$

$$I_b = \rho_{db} I_{sb} \boxed{\frac{s.m}{|s||m|}} + \rho_{ab} I_{sb}$$

else

$$I_r = \rho_{ar} I_{sr}$$

$$I_g = \rho_{ag} I_{sg}$$

$$I_b = \rho_{ab} I_{sb}$$

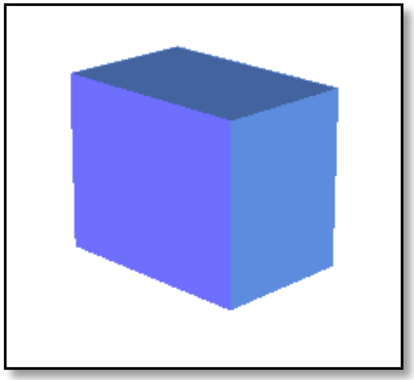
m is the hitNormal stored in the Intersection object

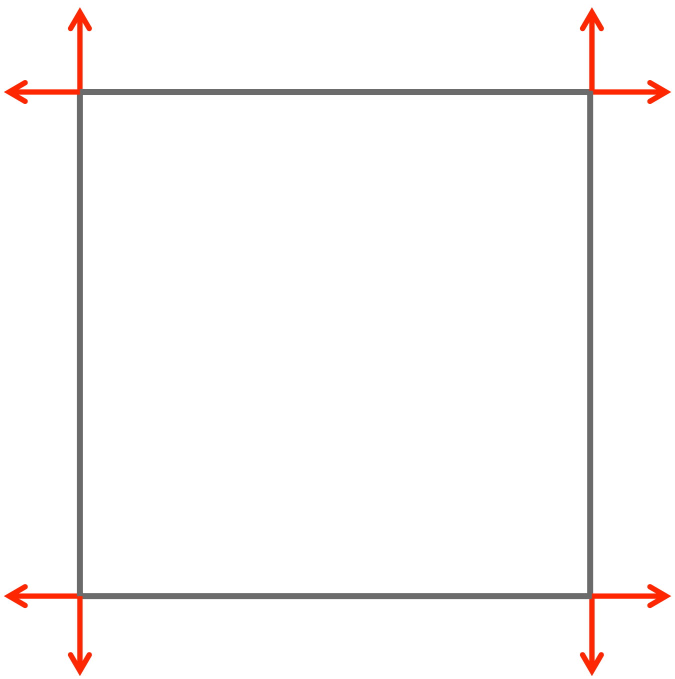
How did we determine the hitNormal when computing the hitPoints between a ray and a polygonal mesh?

We simply took the normal vector associated with one vertex of the face hit by the ray.

This means that all the points in a face get the same normal vector!

Polygonal mesh

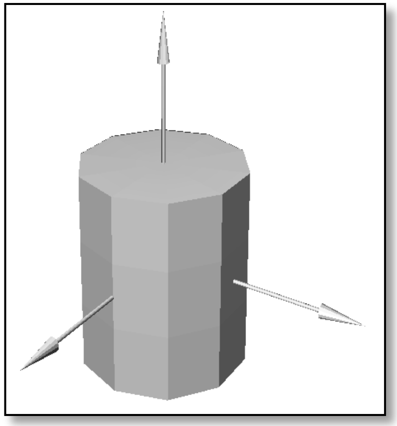




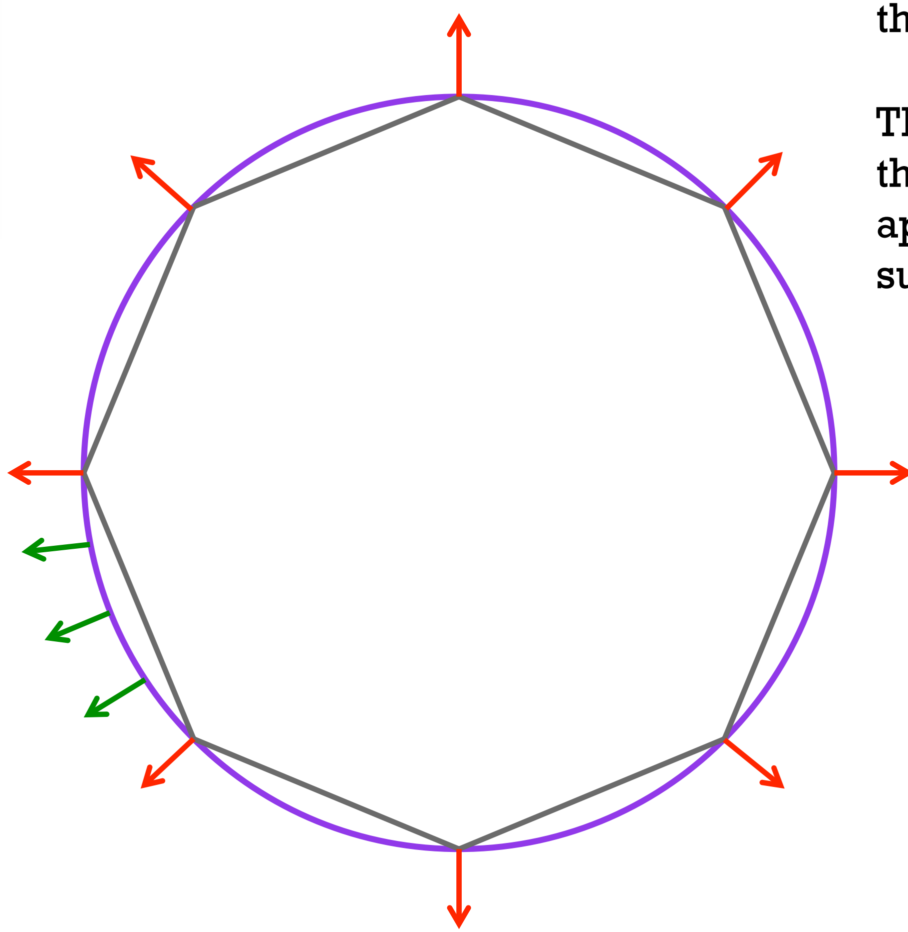
If all the points in a face get the same normal vector, the faces are clearly visible in the final image.

This is fine if the polygonal mesh is identical to the true surface of the 3D object.

Polygonal mesh



Top view of cylinder



If all the points in a face get the same normal vector, the faces are clearly visible in the final image.

This is not what you want if the polygonal mesh is an approximation of the true surface of the 3D object.

Solution?

Phong shading
= interpolate the normal vectors

true surface

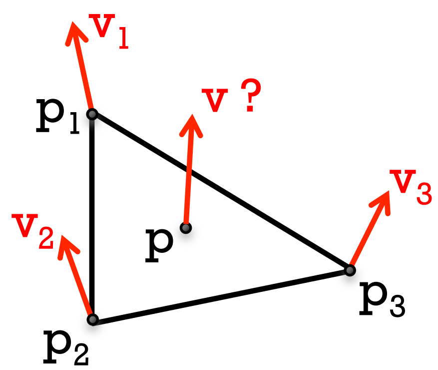
polygonal mesh

normal vector

Normal vector interpolation

A triangular mesh is a polygonal mesh in which the faces are triangles.

One can interpolate the normal vectors for a triangle as follows:



x03 = p.x – p3.x
y03 = p.y – p3.y
z03 = p.z – p3.z
x13 = p1.x – p3.x
y13 = p1.y – p3.y
z13 = p1.z – p3.z
x23 = p2.x – p3.x
y23 = p2.y – p3.y
z23 = p2.z – p3.z

a = $x13^2 + y13^2 + z13^2$
b = x13.x23 + y13.y23 + z13.z23
c = $x23^2 + y23^2 + z23^2$
d = a.c-b^2

l1 = ((c.x13 – b.x23).x03 + (c.y13 – b.y23).y03 + (c.z13 – b.z23).z03)/ d;
l2 = ((-b.x13 + a.x23).x03 + (-b.y13 + a.y23).y03 + (-b.z13 + a.z23).z03)/ d;
l3 = 1 - l1 - l2;

v = l1.v1 + l2.v2 + l3.v3
v.normalize()

Questions?