Interactive Computer Graphics

Lecture 7: Shading

The Need for shading

In the last lecture we added realism to graphics scenes by drawing them as solids.

Object faces were either filled with a plain colour or with a mapped texture.

In real scenes, the reflected light intensity is also important.

The Physics of shading

If we look at a point on an object we perceive a colour and a shading intensity that depends on the various characteristics of the object and the light sources that illuminate it.

For the time being we will consider only the brightness at each point. We will extend the treatment to colour in the next lecture.

Object Properties

Looking at a point on an object we see the reflection of the light that falls on it. This reflection is governed by:

1. The position of the object relative to the light sources
2. The surface normal vector
3. The albedo of the surface (ability to adsorb light energy)
4. The reflectivity of the surface

Light Source Properties

The important properties of the light source are

1. Intensity of the emitted light
2. The distance to the point on the surface

(When considering colour the first is an intensity distribution as we shall see later)

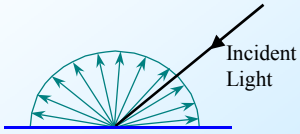
Surface Characteristics

Surfaces can have reflective characteristics ranging from:

1. Dull, (matt) surfaces
2. Shiny (specular) surfaces
3. Reflective (mirror) surfaces.

We will consider matt surfaces first.

Matt surfaces



Perfectly Matt surface
The reflected intensity is the same in all directions

Lambert's Cosine Law

$I_{reflected} = I_i \ k_d \ Cos(\theta)$

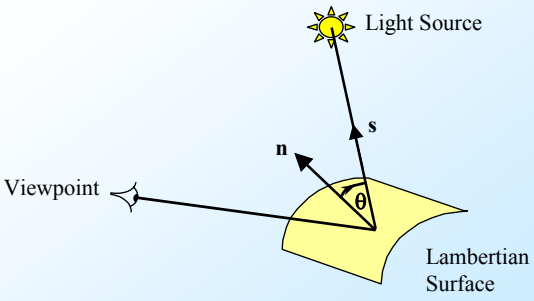
where

Ii is the intensity of the incident light

Kd is a constant (surface albedo)

and θ is the angle between the surface normal and the direction of the light source

Lambert's Cosine Law



Colour

Strictly speaking the surface properties will vary with the wavelength of the light (λ) so we should write:

$I_{reflected}(\lambda) = I_i(\lambda) \ k_d(\lambda) \ Cos(\theta)$

However we will consider just the intensity component for the moment.

Vector Notation

If we specify our graphics scene in 3D space it is convenient to use vectors for the light source direction and the surface normal.

since $\mathbf{n}.\mathbf{s} = |\mathbf{n}||\mathbf{s}|Cos(\theta)$
we can write $Cos(\theta) = \mathbf{n}.\mathbf{s} \ / \ |\mathbf{n}||\mathbf{s}|$

and so Lambert's law is
$I_{reflected} = I_i \ k_d \ \mathbf{n}.\mathbf{s} \ / \ |\mathbf{n}||\mathbf{s}|$

Ambient Light

Pure diffuse shading is impossible to achieve in a scene of several objects, since light that is reflected is itself a light source.

Thus in addition to the direct light from the light sources there is also a background 'ambient' light.

Radiosity

To model the ambient light accurately is very difficult, attempts at doing so are called ‘radiosity’ modelling

As a first approximation we can treat the ambient light as a constant for the whole scene

Reflectance equation with ambient light

$I_{reflected} = I_a \ k_a + I_i \ k_d \ \mathbf{n.s} / |\mathbf{n}||\mathbf{s}|$

and since in general we know nothing about the intensity of the ambient light we can simply write

$I_{reflected} = k_a + I_i \ k_d \ \mathbf{n.s} / |\mathbf{n}||\mathbf{s}|$

and treat k_a as an heuristic constant

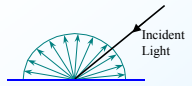
Specularity

In practice no surface is purely matt.

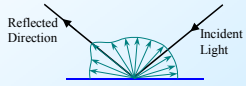
There is usually more energy emitted in the reflected direction than any other.

This effect is termed specular reflection

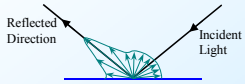
Surface Characteristics



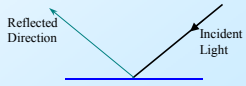
Perfectly Matt surface
The reflected intensity is the same in all directions



Slightly specular (shiny) surface
Slightly higher intensity in the reflected direction



Highly specular (shiny) surface
High intensity in the reflected direction

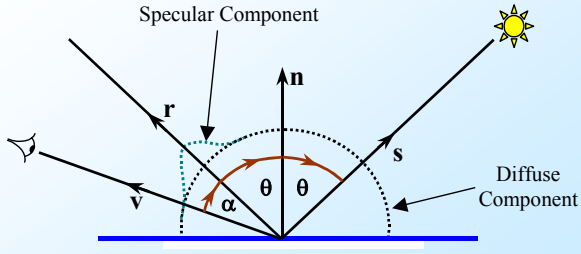


Perfect Mirror
All light re-admitted in the reflected direction

Modelling specular reflection

In practice we can model this effect by adding a specular component to the diffuse component.

The specular component will depend on the angle between the reflected direction and the viewpoint.



Bui-Tuong Phong formula

This empirical formula has gained universal acceptance.

$$I_{reflected} = Ii\ f(\theta)\ \cos^{t}(\alpha)$$

where Ii is the incident light intensity,
$f(\theta)$ is a function that depends on the angle between the incident light ray and the surface normal
t is the shininess

Simplifications and Complications

Usually $f(\theta)$ is taken to be a constant giving a complete reflectance equation as:

$$I_{reflected} = k_a + I_i\ k_d\ Cos(\theta) + I_i\ K_s\ \cos^{t}(\alpha)$$

Some systems specify separate intensities for each component from each light source.

$$I_{reflected} = I_a\ k_a + I_i\ k_d\ Cos(\theta) + I_i\ K_s\ \cos^{t}(\alpha)$$

Vector formulation

$$I_{reflected} = k_a + I_i\ k_d\ \mathbf{n.s} / |\mathbf{n}||\mathbf{s}| + I_i\ K_s\ (\mathbf{r.v} / |\mathbf{r}||\mathbf{v}|)^{t}$$

In practice we can make life easier by normalising all the vectors so that they are specified as unit vectors.

$$\boldsymbol{n}=\mathbf{n}/|\mathbf{n}| \quad \boldsymbol{s}=\mathbf{s}/|\mathbf{s}| \quad \boldsymbol{r}=\mathbf{r}/|\mathbf{r}| \quad \boldsymbol{v}=\mathbf{v}/|\mathbf{v}|$$

$$I_{reflected} = k_a + I_i\ k_d\ \boldsymbol{n.s} + I_i\ K_s\ (\boldsymbol{r.v})^{t}$$

Inverse Square Law

It is well known that light falls off according to an inverse square law. Thus, if we have light sources close to our polygons we should model this effect.

$$I = I_i/d^2$$

where
I is the incident light intensity at a point on an object
I_i is the light intensity at the source and
d is the distance from the light source to the object

Heuristic Law

Although physically correct the inverse square law does not produce the best results.

Instead packages tend to use:

$$I = I_i/(d + k)$$

where k is an heuristic constant

Distance to the viewpoint

One might be tempted to think that light intensity falls off with the distance to the viewpoint, but it doesn't!

Why not?

Using Shading

There are three levels at which shading can be applied in polygon based systems:

Flat Shading
Gouraud Shading
Phong Shading

They provide increasing realism at higher computational cost

Flat Shading

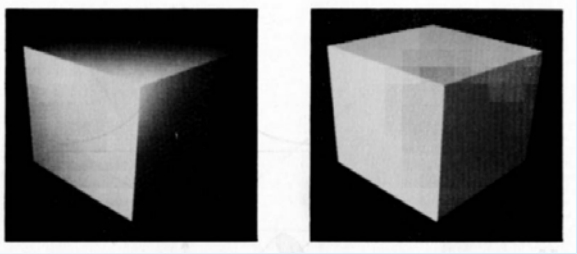
Each polygon is shaded uniformly over its surface.

The shade is computed by taking a point in the centre and the surface normal vector. (Equivalent to a light source at infinity)

Usually only diffuse and ambient components are used.

Shading from different light sources

Local Light Source Flat Shading



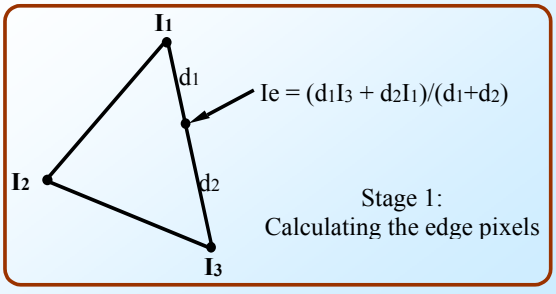
Interpolation Shading

A more accurate way to render a shaded polygon is to compute an independent shade value at each point.

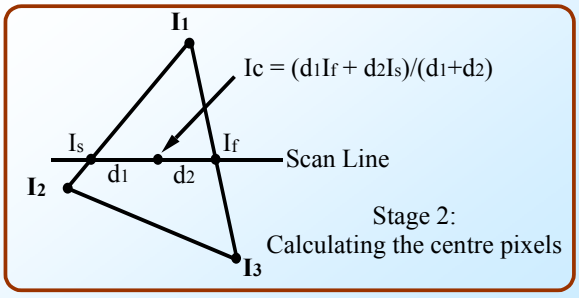
This is done quickly by interpolation.

1. Compute a shade value at each vertex
2. Interpolate to find the shade value at the boundary
3. Interpolate to find the shade values in the middle

Calculating the shades at the edges



Calculating the internal shades



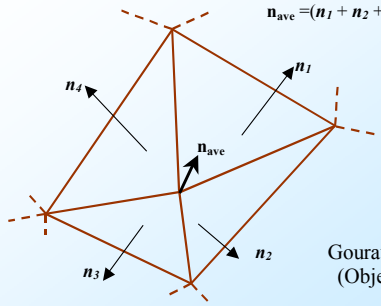
Interpolating over polygons: Gouraud Shading

In addition to interpolating shades over polygons, we can interpolate them over groups of polygons to create the impression of a smooth surface.

The idea is to create at each vertex an averaged intensity from all the polygons that meet at that vertex.

Computing an average normal vector at a vertex

$\mathbf{n}_{ave} = (\boldsymbol{n_1} + \boldsymbol{n_2} + \boldsymbol{n_3} + \boldsymbol{n_4})/4$



Gouraud Shading
(Object Space)

Gouraud Shading

The averaged normal is used to compute a shade value for the vertex.

This is then interpolated as before.

The boundaries of the polygons are thus smoothed out.

Phong Shading

One limitation of Gouraud shading is that we cannot determine specular components accurately, since we do not have the normal vector at each point on a polygon.

A solution is to interpolate the normal vector, not the shade value.

Interpolation of the 3D normals

We may express any point for this facet in parametric form:

$$\mathbf{P} = \mathbf{V1} + \mu 1(\mathbf{V2} - \mathbf{V1}) + \mu 2(\mathbf{V3} - \mathbf{V1})$$

The average normal vector at the same point may be calculated as the vector **a**:

$$\mathbf{a} = \boldsymbol{n1} + \mu 1(\boldsymbol{n2} - \boldsymbol{n1}) + \mu 2(\boldsymbol{n3} - \boldsymbol{n1})$$

and then

$$\mathbf{n}_{average} = \mathbf{a} / |\mathbf{a}|$$

2D or 3D

The interpolation calculations may be done in either 2D or 3D

For specular reflections the calculation of the reflected vector and viewpoint vector must be done in 3D.

Bump Mapping

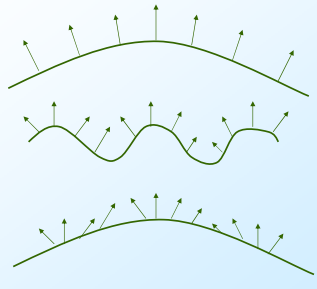
Technique developed by Blinn 1978 to enable a surface to appear rough (eg.wrinkled or dimpled) without the need to model this in three dimensions.

Achieved by peturbating the surface normals of a smooth surface to 'trick' the eye into thinking this surface is rough textured.

Large efficiency gain since model can be much simplified

Main limitation is that profiles and silhouettes of an object will not display the expected 3D surface characteristics

Bump Mapping



Bump Mapping

Bump Mapping

Bump Mapping