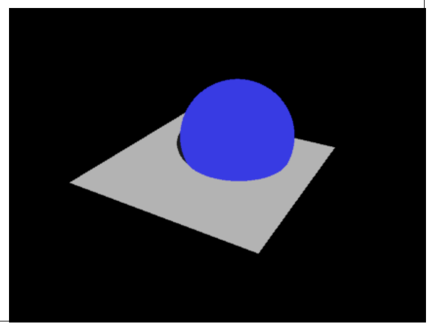
Illumination Modelling

Local shading analysis: interaction between one light source, the viewer and a single point on the object surface.

E = ambient + specular + diffuse.

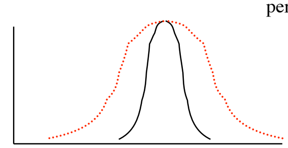
ambient = $I_a K_a$



Specular term

how do we model specular term?

θ = angle between $\overline{V}$ & $\overline{R}$ → perfect mirror dir'n



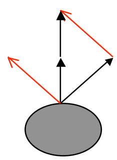
empirical approximation : $\cos^n\theta$

perfect polished mirror refl. $n \to \infty$ (delta function)

Specular term is $K_s \cos^n\theta$

$$R = 2\bar{N}(\hat{N}\cdot\hat{L}) - \hat{L}$$

$$R\cdot V = (2N(N\bullet L)) - L) \bullet V$$

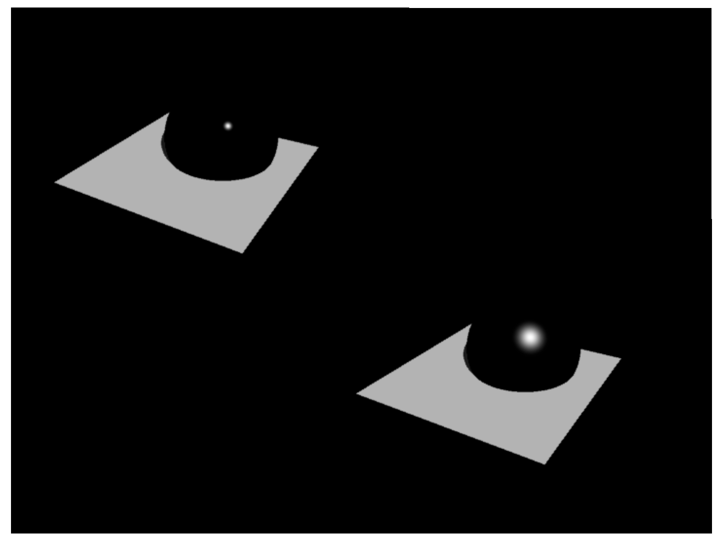


•Problem: computational cost.
•Solution: an approximation known as the *halfway vector* H.

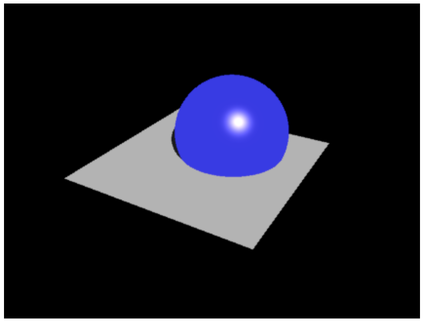
•Instead of asking if V is close to R, ask if N is (almost) halfway between V & L (if so, then R is along V).

$$H = \frac{L+V}{(L+V)}$$

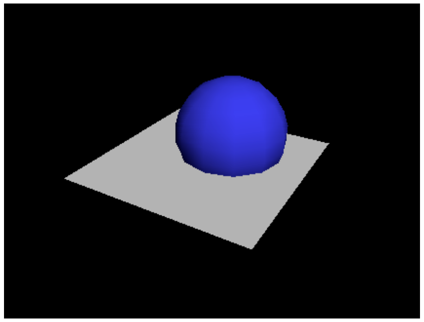
$$N\cdot H \approx \cos\theta$$



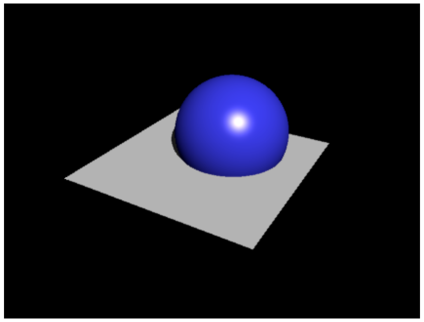
Ambient + Specular



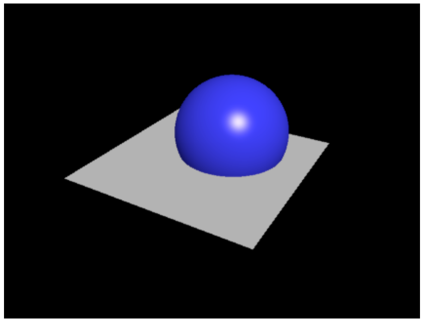
Diffuse Lighting



Specular + Diffuse



Ambient+Specular+Diffuse



Ray Tracing / Casting

Generate the image by tracing rays coming out of the observer's eye (COP) thru each pixel up to first object intersection.

→Leads IMMEDIATELY to approach to visible surface determination.

Just draw (color) intersections.

→Method for shading. Compute $\hat{N}$ & apply I.M. just at intersection pixels.

→Shadows:

Pixels (pts on objs) not seen by any (some) sources.

For each intersection pt $\bar{p}$, fire a ray at (each) source. It obstructed, then it is in shadow.

→leads to "hard" shadows (no pnumbra)

Ray Tracing / Casting (cont'd)

Thus, for each light source pt is seen (illuminated) or in shadow. In shadow, it is effectively zero.

Define S = 0 or 1

$$I = I_a K_a + S I \left[K_d (\bar{N} \cdot \bar{L}) + K_s (\bar{R} \cdot \bar{V})^n \right]$$

(with arrows labelling K_a: $O_{d□}$; S: $□_i$; I: i; K_d: $O_{d□}$; K_s: $O_{s□}$)

Other specialized shadow algs exist.

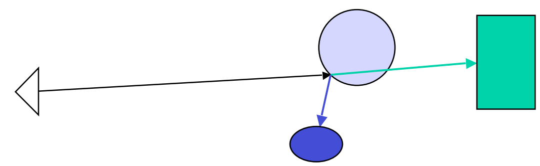
RAY TRACING

Each ray intersects some object

$$w\frac{1}{d^2}$$

$$I = K_a I_a + S I_p f_{att} (K_a (\bar{N} □ \bar{L}) + (K_s (\bar{N} □ H)^2))$$

RAY TRACING



Primary ray from eye leads to secondary rays due to reflection (specular) & refraction (transparency).

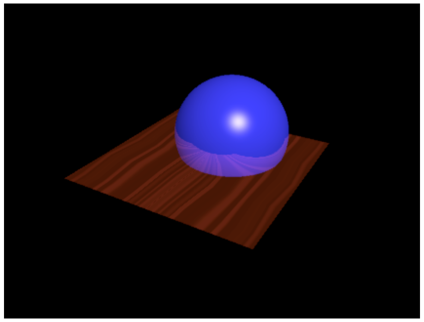
$$I = K_a I_a + \sum_{i □ \text{sources}} S_i f_i I_{pi} (K_d (\bar{N} \cdot \bar{L}) + K_s (\bar{N} \cdot \bar{H})^n)$$

$$+ \; K_s I_r \; + \; K_t I_t$$

specularly reflected daughter ray

transmitted daughter ray

… + transparency



Computing Rays

Consider: COP $= \bar{p}_o(X_o, Y_o, Z_o)$

pixel on window $\bar{p}_1$

rays are $\bar{p}(t) = \bar{p}_o + t(\bar{p}_1 - \bar{p}_o) = \bar{p}_o + t\bar{d}$

Example problem: Find intersection of ray $\bar{r}(\lambda) = \bar{a} + \lambda\bar{b}$

with sphere $f(\bar{x}) = \bar{x}\cdot\bar{x} - 1 = 0$

(detail expl)

This also answers the side question:
why so many spheres in ray-traced images?

Computing Rays (cont'd)

Sol'n: Find λ s.t. $f(\bar{r}(\lambda)) = 0$

$f(\bar{r}(\lambda)) = \bar{r}\cdot\bar{r} = 1$

$= (\bar{a} + \bar{b}\lambda)(\bar{a} + \bar{b}\lambda) - 1$

$= (\bar{b}\cdot\bar{b})\lambda^2 + 2(\bar{a}\cdot\bar{b})\lambda + (\bar{a}\cdot\bar{a}) - 1$

$= A\lambda^2 + 2B\lambda + C$ where $A = \bar{b}\cdot\bar{b}$ $B = (\bar{a}\cdot\bar{b})$ $C = \bar{a}\cdot\bar{a} - 1$

quadratic in l

$$\lambda = \frac{-2B \pm \sqrt{4B^2 - 4AC}}{2A} = \frac{-B}{A} \pm \frac{\sqrt{D}}{A} \qquad D = B^2 - AC$$

Real sol'n for $D \geq 0$

$D < 0$ ray misses sphere, $D = 0$ graze, $D > 0$ pierce

sol'n: both in front of eye both < 0, both behind

Computing Rays (cont'd)

With sphere, normal N for shading is simply:

centre: $\bar{C} = (a,b,c)$

$r(\lambda) == (x,y,z)$

$$\hat{N} = \frac{x - a, y - b, z - c)}{r}$$

Specific similar methods exist for many other types of surface.
Read Sec. 10.2
Assignment problem: do it for --- superquadrics or some such.

To make more efficient, use bounding volumes (boxes).

Alternative: spatial partioning (top-down bdding volumes).

only consider objs that
intersect relevant boxes

FOR AFFINE DEF OBJS, CAN INV X-FORM RAYS & USE METHOD SUITABLE TO UN-DEF OBJ.

17

Ray Tracing Refinements

Observe: computational burden of recursive RT is substantial.
For each ray: must do intersections with every object.
(And face culling doesn't work since some rays can come from behind.)

How many rays?

$$(\text{N pixels})\cdot\left[\begin{array}{l}(2^n-1)\ \text{nodes in ray tree}\\ +\ m(2^n-1)\ \text{shadow rays}\end{array}\right]$$

i.e. lots of incentive to make this efficient

18

Ray Tracing Refinements (cont'd)

➔ Light buffer: organize objects about each light source

➔ Tree depth control: avoid expanding ray tree for branches that have a small contribution.

➔ Area sampling: replace thin rays by cones (cone tracing).

19

Ray Tracing

not camera!

Follow rays from eye (through pixels) into scene.

BASIC STEP

Where they hit the 1st object:
compute reflectance, from light source(s).

RECURSION

Follow specular bounces & transparency rays to account for inter-object effects.
Leads to ray tree.

RT is core of global illumination.

Omits effect of other objects on diffuse reflectance term!

20

Visible Surface Determination (15.2)

15.1: Idea of coherence: if you have an answer someplace, it will probably apply nearby | | incremental algorithm

obj. coherence	- bounding box
	- flat shading
face edge coherence	- edges only change when they cross faces/edges
etc.	

Conclusion: computation precedes proj'n or depth info lost.
Typically after normalizing xform.

Face culling already discussed.
Bounding volumes & spatial partitioning.

Roberts Algorithm 15.3.1

Discuss Roberts thesis.

- Edge detection
- Labelling
- Back-projection

Alg: All edges on faces of convex polyhedron.

1) Face culling.

2) Test each remaining edge against each (convex) polyhedron.

a) Via bounding box.

b) Test line against relevant faces.

Appel’s Algorithm 15.3.2

Exploits object coherence - it’s a form of incremental algorithm.

Idea:

Define index of number of faces hiding a line - “quantitative invisibility”: QI
When moving behind a fron-facing polygon, increment QI, when coming out, decrement QI.

QI changes at either:

- open polygon edges.
- edges between fron & back facing polygons.

⇨ these are contour lines

Other edges do not change visibility.

Contour vs. line occlusion determined by looking at triangle formed by line & eye wrt contour line.

Cook & Torrance Model

Vs Phong: a physically-based model of specular reflection
-Based on incident energy instead of intensity
-Specular term is physically-based
-Color change of highlight based on physics

Key aspect is the role of the microfacet distribution in the specular reflection - re the physical micro-texture of the material.
Assume surface is made of small perfectly specular fragments. The ones aligned with the halfway vector reflect.

$$P_s = \frac{F_\square}{\square}\frac{DG}{(N \cdot V)(N \cdot L)}$$

M = 0 : mirror
small M : shiny
lg M : dull

D is fraction of aligned microfacets.
D can be based on Beckmann distribution.

$$\frac{1}{4m^2 \cos^4 \square} e^{\square[(\tan e / M]^2}$$

$\square =< NH$

$M \square$ *RMS microfacet slope*

Can combine several D’s for multi-scale roughness.

Cook & Torrance Model

G: Geometrical attenuation factor
Mutual shadowing, V-shaped grooves
(shadowed rays lead to diffuse energy)
also masking (interference)

$$\frac{2(\hat{N} \cdot \hat{H})(\hat{N} \cdot \hat{V})}{(\hat{V} \cdot \hat{H})}$$

$F_\square$:Fresnel term
describes reflection phenomenon on smooth (planar) side of each microfacet
describes color shift & intensity

Key diff. w. Phong is off-specular peak & larger specular term & sharp angles

Shading areas of an image

Constant (flat) shading

For each face, use N to compute IM.
Color the face a uniform color.

If we only know vertex locations, derive N for face.

Result: obvious artefacts due to number of faces used.

Interpolated Shading 16.2.4
a.k.a. Gouraud Shading

Compute normal at each vertex.
Compute vertex intensity I.
Interpolate intensity along each edge.

I_a at V_1 = spec + diffuse + ambient

I_a (along a)

$$= I - (I_1 - I_2)\, t \qquad t \in [0,1]$$

$$= I_1 - (I_1 - I_2)\, \frac{Y_1 - Y_s}{Y_1 - Y_2}$$

$$I_p = I_b(\hat{Y}) - (I_b(\hat{Y}) - I_a(\hat{Y}))\, \frac{X_b(\hat{Y}) - X_p}{X_b(\hat{Y}) - X_a(\hat{Y})}$$

Phong Shading

Interpolate normals instead of I.

Compute I at each point.

1) Use IM (illumination model) to compute I at vertices (Gouraud)
   Phong | | N is given at vertices
2) Compute N along edges by interpolation between vertex valves (do this for I/Gouraud)
3) Walk up each polygon face raster-line by raster-line. Along each line, compute N at each point by interpolation between end points. Compute I(N).

Phong

Now we have a different N at each point:
Thus, can have a highlight in the middle of a facet.

<u>SHADING</u>	<u>IM</u>
Flat shading	const
↓	↓
	ambient
	↓
Gouraud	Phong: am + dif + spec
↓	↓
Phong	Phys: Cooke & Torrance

29

Hidden surface removal summary

Z buffer:

No constraint on object types of compactness.
Z values ust be high-res: (16/32 bits pp).
Uses much space.
Precision problems.
Scan conversion can lead to

List priority:

Sort faces.
Painters algo.
→ typ fails
→ furthest z per face

30

Adding Surface Cues 16.3 READ IT

1) Surface detail polygons paint

2) Texture mapping

Like morphing but map (V,V) space of texture image onto 3D surface, then project.
Can use inverse mapping.
pixel| | surface| | texture map
combine appropriate texels from map
Can involve weighting

3) Bump mapping

→ perturb suface normal
Points P displaced:

$$Po\text{int}\, sPdisplaced : \overline{P}^1 = \overline{P} + B\frac{\overline{N}}{|\overline{N}|}$$

$$Note\,\overline{N} = \overline{P_s} \; \overline{P}_t \quad \mathrm{P_s}$$

Fails at silhouette edges
Don't actually move data points!