7.1 Ray Tracing

Reading

Recommended:

- Foley, et al., Sections 15.10, 16.1, 16.12

Optional:

- A. Glassner, editor. *An Introduction to Ray Tracing*. Academic Press, 1989. Chapter 1.
- T. Whitted. An improved illumination model for shaded display. *Communications of the ACM* 23(6), 343-349, 1980.
- J. Painter and K. Sloan. Antialiased ray tracing by adaptive progressive refinement. *Computer Graphics* (SIGGRAPH 89) 23(3): 281–288.
- Michael Oren and Shree K. Nayar. Generalization of Lambert's reflectance model. *Computer Graphics Proceedings, Annual Conference Series* (SIGGRAPH 94), 239–246.

Introduction

Now we move on to (what at least used to be) one of my favorite subjects: "photorealistic rendering."

We'll take this in two parts:

1. Ray tracing
   - an earlier and more intuitive method
   - corresponds to Monte Carlo integration
2. Radiosity
   - adapated from methods in heat transfer
   - corresponds to a finite-element solution

Overview of lecture

1. Local illumination: the Phong model
2. Basic ray-tracing algorithm
3. Ray–surface intersection calculations
4. Acceleration techniques
   - Hierarchical bounding volumes
   - Space-subdivision methods
   - Progressive methods
5. Antialiasing
   - Supersampling
   - Adaptive sampling
   - Distributed ray tracing

The Phong illumination model

Introduced by Phong Bui-Tuong in 1975:

- Not physically-based
- Gives a first-order approximation to physical light reflection
- Very fast
- Widely used

The problem

Given:

- A point P on a surface
- Visible through pixel p
- A unit vector $\mathbf{V}$ from P to the viewer
- A unit vector $\mathbf{L}$ from P to a point light source

Find: The intensity and color of light radiating from P to the viewer.

Solved by developing a model of light interaction with materials.

This model is known as:

- A "shading model"
- A "lighting model"
- A "light reflection model"
- A "local illumination model"
- A "reflectance model"

Ambient reflection

The simplest shading rule is to just give each polygon a single color (16.1):

$$I = k_i$$

where:

- I is the resulting intensity
- k_i is the intrinsic shade associated with the object

Alternate characterization

The same shading rule has an alternate characterization:

Suppose that the only light is *diffuse* and *nondirectional* – for example:

- Object suspended in a room coated with latex paint
- Light source is not directly visible to the object

Alternate characterization, cont.

In this case, light reflects from the surface according to its "reflectivity" k_a.

Gives the new ambient shading equation (16.2):

$$I = k_a I_a$$

where:

- k_a is the "ambient reflection coefficient"
  - Really, the "reflectance of ambient light"
  - This reflectance is assumed to be "diffuse" — equal in all directions
- I_a is the "ambient intensity"
  - Really, the "intensity of the ambient light"

Wavelength dependance

Really, k_a and I_a are functions over all wavelengths λ.

Ideally, we should do the calculation on these functions:

$$I(\lambda) = k_a(\lambda)\, I_a(\lambda)$$

and then find good RGB values to respresent $I(\lambda)$.

"Traditionally," though, k_a and I_a are represented in RGB, and the computation is performed on each color component separately:

$$I_{\text{red}} = k_{a,\text{red}}\, I_{a,\text{red}}$$

$$I_{\text{green}} = k_{a,\text{green}}\, I_{a,\text{green}}$$

$$I_{\text{blue}} = k_{a,\text{blue}}\, I_{a,\text{blue}}$$

Diffuse reflection

Review of ambient shading model:

- Objects have different colors
- We can control the overall light intensity
  - What happens when we turn off the lights?
  - What happens as the light intensity increases?
  - What happens if we change the color of the lights?

Problem: So far, objects are all uniformly lit.

- Not the way things usually appear
- Due to the existence of "directional light sources"

"Diffuse" or "Lambertian" reflection will allow reflected intensity to vary with the direction of the light.

Diffuse reflectors

Diffuse reflection occurs from dull, matte surfaces, like latex paint or chalk.

These "diffuse" or "Lambertian reflectors" reradiate light equally in all directions.

Picture a rough surface with lots of tiny "microfacets":

The microfacets have the effect of distributing light rays in all directions.

Note:

- Light may actually penetrate the surface, bounce around, and then reflect back out.
- Accounts for colorization of diffusely reflected light by plastics

Diffuse reflectors, cont.

While the reflected intensity from a diffuse surface does not depend on the direction of the viewer, the incoming energy does depend on the direction of the light source:

An illuminated area (Foley, 16.2)

Diffuse reflectors, cont.

The incoming energy is proportional to $\cos\theta$, giving the diffuse reflection equations (16.3), (16.4):

$$\begin{aligned} I &= k_d I_\ell \cos\theta \\ &= k_d I_\ell (\mathbf{N} \cdot \mathbf{L})_+ \end{aligned}$$

where:

- k_d is the "diffuse reflection coefficient"
- I_ℓ is the intensity of the light source
- $\mathbf{N}$ is the normal to the surface
- $\mathbf{L}$ is the direction of the light source
- $(x)_+$ means $\max\{0, x\}$

Intensity drop-off with distance

The laws of physics state that the intensity of a point light source must drop off with its distance squared.

We can incorporate this effect by multiplying I_ℓ by $1/d^2$.

Sometimes, this distance-squared drop-off is considered "too harsh." Foley et al. suggest using (16.8)

$$f(d) = \min\left\{1, \frac{1}{a_0 + a_1 d + a_2 d^2}\right\}$$

for user-supplied constants a_0, a_1, a_2.

Specular reflection

"Specular reflection" accounts for the highlight that you see on objects.

- Particularly important for smooth, shiny surfaces, such as:
  - · Metals
  - · Polished rocks
  - · Plastics
  - · Safeway apples

Properties:

- Specular reflection depends on the viewing direction $\mathbf{V}$
- Often determined soley by the color of the light
  - · Corresponds to absence of internal reflections

"Derivation"

(fig 14-12)

For a perfect mirror reflector, light is reflected about $\mathbf{N}$. So

$$I = \begin{cases} I_\ell & \text{if } \mathbf{V} = \mathbf{R} \\ 0 & \text{otherwise} \end{cases}$$

For a near-perfect reflector, you might expect the highlight to fall off quickly with increasing angle ϕ.

Known as:

- "Specular" reflection (Phong)
- "Rough specular" reflection
- "Directional diffuse" reflection
- "Glossy" reflection

"Derivation," cont.

One way to get this effect is to take $(\mathbf{R} \cdot \mathbf{V})$, raised to a power n_s.

As n_s gets larger,

- The drop-off becomes {more, less} gradual
- Gives a {larger, smaller} highlight
- Simulates a {more, less} glossy surface

Thus, the specular component is given by:

$$I = k_s \, I_\ell \, (\mathbf{V} \cdot \mathbf{R})_+^{n_s}$$

The Phong shading model

Since light is additive, we can handle multiple lights by taking the sum over every light.

The Phong shading model can be summarized by putting all the terms we have seen together (16.15):

$$I = k_a \, I_a + \sum_i f(d_i) \, I_{\ell i} \, [k_d \, (\mathbf{N} \cdot \mathbf{L}_i)_+ + k_s \, (\mathbf{V} \cdot \mathbf{R}_i)_+^{n_s}]$$

Note: $k_a, k_d, k_s, I_a, I_{\ell i}, I$ are all RGB vectors.

Choosing the parameters

Experiment with different parameters settings. To get you started, here are a few suggestions:

- Try n_s in the range [0, 100]
- Try $k_a + k_d + k_s \leq 1$
- Use a small k_a (~ 0.1)

	n_s	k_d	k_s
Metal	large	small, color of metal	large, color of metal
Plastic	medium	medium, color of plastic	medium, white
Lunar dust	0	large, white	0

Ray-tracing algorithm

Tracing a ray (Glassner, fig. 11)

Algorithm:

1. For each pixel, trace a "primary ray" to the first visible surface (same as "ray casting").
2. For each intersection, trace "secondary rays":
   - "Reflected ray" in direction $\mathbf{R}$.
   - "Refracted ray" in direction $\mathbf{T}$.
   - "Shadow rays" in directions $\mathbf{S}_i$ to light sources.

Shading

Let $I(p, r)$ be the intensity seen from point p along direction r:

$$I(p, r) \;=\; I_{\text{direct}} \;+\; I_{\text{reflected}} \;+\; I_{\text{transmitted}}$$

where

- I_{direct} is computed from the Phong model
- $I_{\text{reflected}} \;=\; k_{\text{reflected}}\, I(P, \mathbf{R})$
- $I_{\text{transmitted}} \;=\; k_{\text{transmitted}}\, I(P, \mathbf{T})$

Reflected and transmitted directions

Law of reflection:

$$\theta_i \;=\; \theta_r$$

Snell's law of refraction:

$$\eta_i \sin\theta_i \;=\; \eta_t \sin\theta_t$$

where η_i, η_t are "indices of refraction."

The ray-tracing tree

As the recursively traced rays ricochet through the scene, they are added to a "ray-tracing tree":

The ray-tracing tree (Glassner, fig. 12)

Ray-tracing pseudocode

function $RayTrace(p, r)$:
  $(P, Obj) \leftarrow RayCast(p, r)$
  $I \leftarrow Obj.k_a * I_a$
  for each light source ℓ **do**:
    $(P', LightObj) \leftarrow RayCast(P, Dir(P, \ell))$
    if $LightObj = \ell$ **then**:
      $I \leftarrow I + \langle \text{diffuse term} \rangle + \langle \text{spec term} \rangle$
    end if
  end for
  $I \leftarrow I + Obj.k_{\text{reflected}} * RayTrace(P, \mathbf{R})$
  $I \leftarrow I + Obj.k_{\text{transmitted}} * RayTrace(P, \mathbf{T})$
  return I
end function

Q: How do you bottom out?

Intersecting rays with spheres

Given:

- A ray $\mathbf{P}(s)$ through initial position $\mathbf{P}_0$ in direction $\mathbf{u}$ (a unit vector):

$$\mathbf{P}(s) \;=\; \mathbf{P}_0 \;+\; s\,\mathbf{u}$$

- A sphere S centered at $\mathbf{P}_c$ with radius r

Find: The intersection of $\mathbf{P}(s)$ with S.

Solution:

Acceleration techniques

Vanilla ray tracing is really slow!

In practice, some acceleration technique is almost always used:

1. Hierarchical bounding volumes

Acceleration techniques, cont.

Spatial subdivision (Foley, 15.62)

2. Space subdivision methods

Idea:

- Partition objects spatially.
- Trace ray through voxel array.

Partition can:

- Be uniform or adaptive
- Use octtrees or k-d trees

Acceleration techniques, cont.

3. Adaptive progressive refinement

Idea [Painter and Sloan]:

- Interpolate between ray traced samples.
- Only send rays where variance is high.

Antialiasing

One straightforward way to do antialiasing is to supersample, either uniformly or adaptively:

Supersampling by tracing rays through corners (Foley, 15.66)

Distributed ray tracing

Idea:

- Use non-uniform (jittered) samples.
- Replaces aliasing artifacts with noise.
- Can distribute rays in many different dimensions.

Q: What if rays are distributed over:

- Reflection/refraction directions?
- A light source area?
- A finite lens aperture?
- Time?

Summary

What to take home from this lecture:

1. An understanding of the Phong model
2. Enough to implement basic recursive ray tracing.
3. How reflection and transmission directions are computed.
4. How ray–object intersection tests are performed.
5. Basic acceleration strategies.
6. Concept of distributed ray tracing.