The Rendering Equation

Logistics

- Checkpoint 3
  - Grading complete
  - Please update when ready
- Checkpoint 4
  - Due Monday (hey, that's today!)
- Checkpoint 5
  - Given today
- RenderMan
  - Due May 14
  - Mac problems…see me after class.

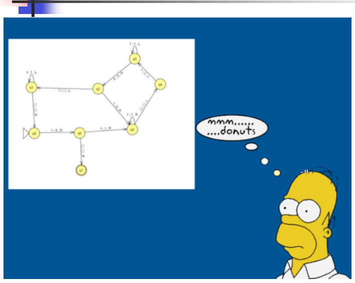
Projects

- Approx 26-28 projects
- Listing of projects now on Web
- Presentation schedule
  - Presentations (15 min max)
  - Last 4 classes (week 9 + week 10 + finals week)
  - Sign up
    - Email me with 1st , 2nd , 3rd choices
    - First come first served.
- Mid-quarter report due Wednesday
  - Drop in dropbox.

Finals date has been set

- Saturday, May 19th
- 8:00am -- 10am
- Room 70-1620
- Project presentations.
- Conflicts? Let me know.

Most important question…



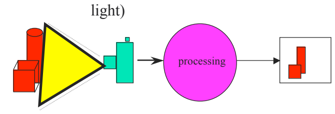
DUNKIN' DONUTS

OR

Krispy Kreme

Computer Graphics as Virtual Photography

Photography:	real scene	camera (captures light)	photo processing	Photographic print



Computer Graphics:	3D models	camera model (focuses simulated lighting)	tone reproduction	synthetic image

Today's Class

- Rendering
  - The Rendering Equation
  - Intro to Recursive Ray Tracing
- Checkpoint 5 -- Reflection

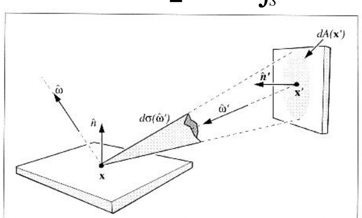
The Rendering Equation

- Kajiya: 1986
- "Unified context for viewing rendering algorithms as more or less accurate approximations to the solution of a single equation"
- Expresses the quantity of light transferred from one point **x'** to another **x**, summed over all points.

The Rendering Equation

- General form

$$I(x,x') = g(x,x')\left[\varepsilon(x,x') + \int_S \rho(x,x',x'')I(x',x'')dx''\right]$$



Ashdown

The Rendering Equation

- Transport Intensity

$$\boxed{I(x,x')} = g(x,x')\left[\varepsilon(x,x') + \int_S \rho(x,x',x'')I(x',x'')dx''\right]$$

I(x, x') = Transport energy or intensity of light passing from point x' to point x (unoccluded two point transport)

The Rendering Equation

- Geometry term

$$I(x,x') = \boxed{g(x,x')}\left[\varepsilon(x,x') + \int_S \rho(x,x',x'')I(x',x'')dx''\right]$$

$g(x, x')$ = geometry term

= 0, if x is not visible from x'

= $1/d^2$ otherwise

The Rendering Equation

- Emittance

$$I(x,x') = g(x,x')\left[\boxed{\varepsilon(x,x')} + \int_S \rho(x,x',x'')I(x',x'')dx''\right]$$

$\varepsilon(x, x')$ = light energy emitted from point x' towards x.

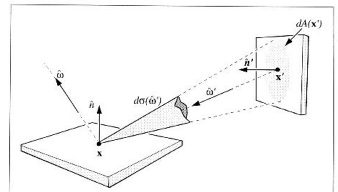
The Rendering Equation

- Scattering

$$I(x,x') = g(x,x')\left[\varepsilon(x,x') + \boxed{\int_S \rho(x,x',x'')I(x',x'')dx''}\right]$$

$\rho(x, x', x'')$ = light energy reflected off point x' towards point x from light coming from x''

The Rendering Equation



$\rho(x, x', x'')$ = light energy reflected from point x' towards point x from light coming from x'', i.e. BRDF

The Rendering Equation

- Incoming = direct + indirect (scattered)

$$I(x,x') = g(x,x')\left[\varepsilon(x,x') + \int_S \rho(x,x',x'')I(x',x'')dx''\right]$$

$$I = \underbrace{g\varepsilon}_{\text{direct}} + \underbrace{gR(I)}_{\text{indirect}}$$

The Rendering Equation

- In short…
  - The transport of light from point **x'** to point **x** is equal to the *sum* of
    - the light emitted from **x'** in the direction of **x** and
    - the total light *scattered* from x' towards x due to light from all *other surfaces* in the scene.

The Rendering Equation

- This can be expanded using the Neuman series for implementation purposes:

$$I = g\varepsilon + g\varepsilon(Rg) + g\varepsilon(Rg)^2 + g\varepsilon(Rg)^3 + \dots$$

or

$$I = \sum_{i=0}^{\infty} g\varepsilon(Rg)^i$$

The Rendering Equation

- Why is this important?

$$I = \underbrace{g\varepsilon}_{\text{direct}} + \underbrace{g\varepsilon(Rg)}_{\text{1st scattering}} + \underbrace{g\varepsilon(Rg)^2}_{\text{2nd scattering}} + \underbrace{g\varepsilon(Rg)^3}_{\text{3rd scattering}} + \dots$$

Rendering methods can be characterized by the number of scatterings considered

The Rendering Equation

- Local vs Global Illumination Models

$$\underbrace{I = g\varepsilon}_{\text{direct}} + \underbrace{g\varepsilon(Rg)}_{\text{1st scattering}} + \underbrace{g\varepsilon(Rg)^2}_{\text{2nd scattering}} + \underbrace{g\varepsilon(Rg)^3}_{\text{3rd scattering}} + \ldots$$

Local illumination - only considers direct component

Global illumination - also considers other scattered component

The Rendering Equation

- Local Illumination

$$I = \underbrace{g\varepsilon}_{\text{direct}}$$

Only object's first contact with light is considered.
Lighting "simulated" by illumination model used.

NOTE: Kajiya does not include ambient light!

The Rendering Equation

- Global Illumination

$$\underbrace{I = g\varepsilon}_{\text{direct}} + \underbrace{g\varepsilon(Rg)}_{\text{1st scattering}} + \underbrace{g\varepsilon(Rg)^2}_{\text{2nd scattering}} + \underbrace{g\varepsilon(Rg)^3}_{\text{3rd scattering}} + \ldots$$

Considers multiple scatterings
- Ray-Tracing
- Radiosity
- Kajiya's method

The Rendering Equation

- Notes:
  - Uses geometric optics, based on light as rays
  - Phase, diffraction, and transmission through participatory media not considered (i.e., only homogenous refraction considered)
  - Dependence on wavelength is implied
  - Not expressed using physical units
    - Question: Isn't I (x, x') simply radiance at a point?
    - Yes! (with the exception of some geometric terms…)
    - More when we talk about radiosity

The Rendering Equation

- Summary
  - Equation for describing rendering algorithms.
  - Describes light arriving at a point from another point (and indirectly all other points)
  - Considers direct light and recursive scattering

Solving the Rendering Equation

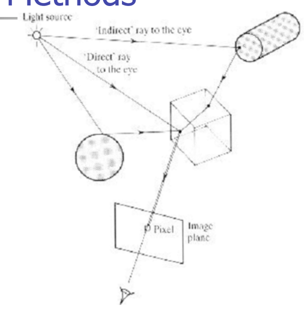
- Approximations to the solution of the rendering equation
  - Recursive Ray Tracing
  - Radiosity
  - Kajia's Method (path tracing)

Rendering Methods

- Ray Tracing
  - Light rays are traced from the eye, through a viewing plane, into scene
  - When rays strike an object, further rays are spawned representing reflection and refraction.
  - These newly spawned rays can strike objects and spawn more rays, etc.

Rendering Methods

- Ray Tracing



Watt/Watt

Rendering Methods

- Ray tracing

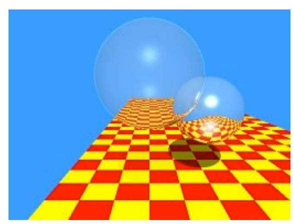
$$I = g^{\varepsilon} + g^{\varepsilon}(Rg) + g^{\varepsilon}(Rg)^2 + g^{\varepsilon}(Rg)^3 + \ldots$$

direct, 1st scattering, 2nd scattering, 3rd scattering

Number of scatterings depend on recursion depth allowed by ray tracer

Rendering Method

- Ray Tracing – Examples



- More on this in 2nd half

Rendering Methods

- Radiosity
  - The problem with ray tracing
    - Great for specular type reflections
    - Awful for diffuse reflections.
  - Radiosity
    - From the guys who brought you Cook-Torrance illumination model!
    - Not quite as elegant as ray tracing
    - More physically based

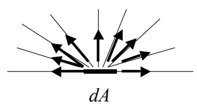
Rendering Methods

- Radiosity
  - Based on the theory of heat transfer
  - Calculate lighting in a steady state
  - Assumes that all surfaces are perfectly diffuse
  - Formulates a large systems of linear equations
  - Solution of these equations give you radiant exitance at each point

Rendering Methods

- Radiosity
  - Radiant exitance - radiant flux out

$$M = \frac{d\Phi}{dA}$$



Rendering Methods

- Radiosity
  - Image is created by using calculated radiant existance values in standard rendering process.
    - In essence, radiosity defines a “made to fit” texture mapping

Rendering Methods

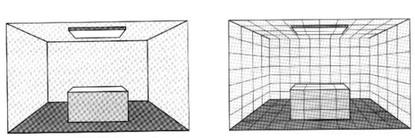
- Radiosity
  - View dependence vs view independence
    - Radiosity provides a view independent solution
    - Scene needs to be further rendered from a given view point.

Rendering Methods

- Radiosity
  - Not points -- But patches
    - Scene is subdivided into patches
    - Radiant exitance will be calculated for each patch

Rendering Methods

- Radiosity
  - Patches



[Ashdown94]

Radiosity - Basics

- Basic idea
  - Each patch will receive a certain amount of light from the environment
  - It will reflect fraction back into the environment
  - Keep track of amount of light reflected back
  - Continue till all light has been exhasted.

Radiosity - Basics

- Key idea
  - Since all objects are perfectly diffuse, we can determine where light is coming from (and going) by simply considering the geometry of the scene.
  - Calculation of radiant exitance per patch is can be detemined mathematically using a closed form solution.

Rendering Methods

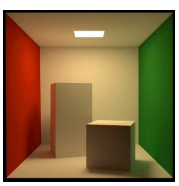
- Radiosity

$$I = \underbrace{g^{\varepsilon}}_{\text{direct}} + \underbrace{g^{\varepsilon}(Rg)}_{\text{1st scattering}} + \underbrace{g^{\varepsilon}(Rg)^2}_{\text{2nd scattering}} + \underbrace{g^{\varepsilon}(Rg)^3}_{\text{3rd scattering}} + \dots$$

Mathematical derivation assumes limit as number of scatterings goes to Infinity

Rendering Methods

- Radiosity – Examples

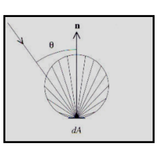


- Video

But does anyone use radiosity in practice

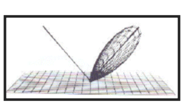
- Glad you asked!
  - Bunny (Blue Sky Studios)

Remember BRDFs?



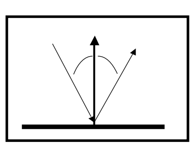
Perfectly diffuse

Radiosity



Specular & diffuse

Reality



Perfectly specular

Ray tracing

Rendering Methods

- Getting closer to reality
  - Ray tracing
    - Replace rays with cones/beams
    - Distributed ray tracing
      - Spawn more rays
      - Stochastic sampling
    - Supplement backward ray tracing with forward ray tracing (e.g. Photon Mapping)

Rendering Methods

- Getting closer to reality
  - Radiosity
    - Two path method – combine ray tracing and radiosity

Rendering Methods

- Kajiya's solution (Path tracing)
  - Much like ray tracing
  - Determine the path of light that eventually reaches the eye.
  - Only 1 ray spawned per intersection
    - Reflection in specular
    - Reflection in diffuse
    - Refraction
  - Which ray to spawn determined via stochastic sampling.

Kajiya's Method

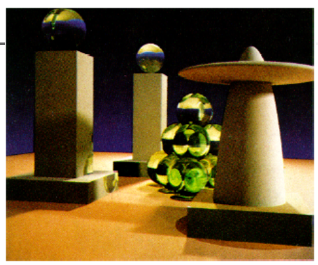


Figure 6. A sample image. All objects are neutral grey. Color on the objects is due to caustics from the green glass balls and color bleeding from the base polygon.

Summary

- Rendering equation
  - Mathematical expression of rendering process
- Solutions
  - Ray Tracing
  - Radiosity
  - Combination of both.
- Questions?

Questions?

- Next: Recursive ray tracing…in depth

- break