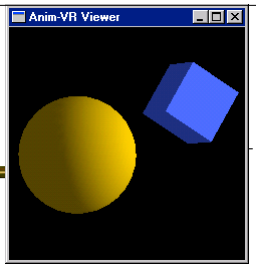
Global Illumination

Recall: Global vs. Local Illumination

- Global
  - Model interactions between light and objects
  - Impossible in practice!
    - Still active research problem
  - Basic idea: "Follow the light"
- Local
  - Consider each object independently

Much Different Approach Than OpenGL



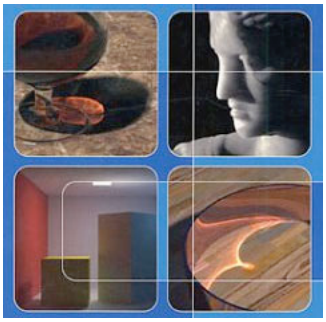
- Up to now:
  - Scan convert objects, computing color as we go
  - Can't easily do:
    - Shadows
    - Reflection
    - Accurate transparency
- Instead, let's think about light and the viewer's eye

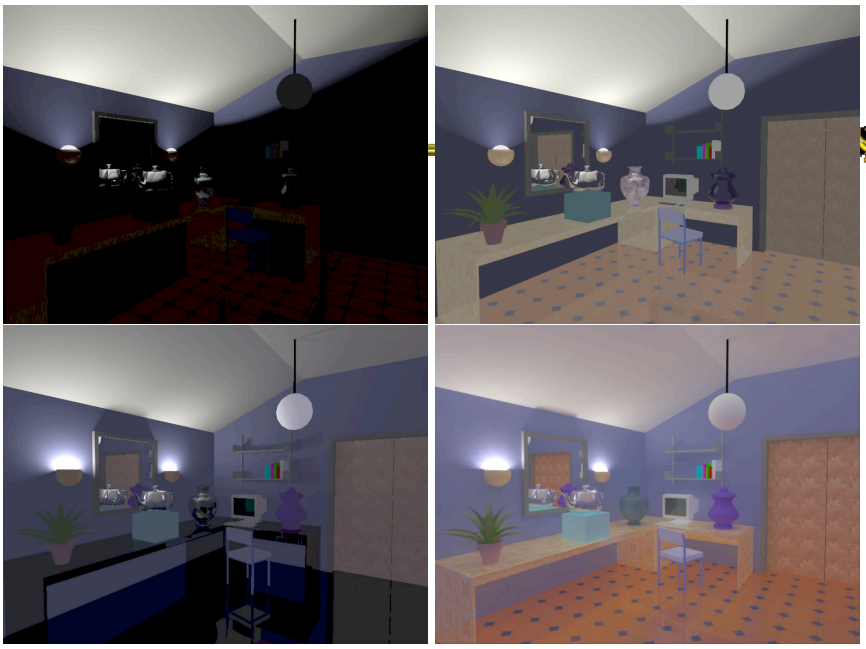
Global Illumination Models

- Framework to evaluate/compare algorithms
- The Rendering Equation
  - Kajiya's general formulation of global illumination
- Radiance Equation
  - Alternative version of the Rendering Equation
- Path notation
  - Classification of algorithms based on the kinds of surface-to-surface interactions they support
  - Non-mathematical

Global Illumination Algorithms

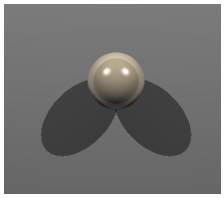
- Two well known methods
  - Ray tracing (10.3.1, Ch 12)
  - Radiosity (10.3.2, Ch 11)
- Newer methods
  - Fix flaws in RT and Rad (sec 10.5-10.11)
  - Other more recent methods
    - E.g., Photon mapping (Jenson)



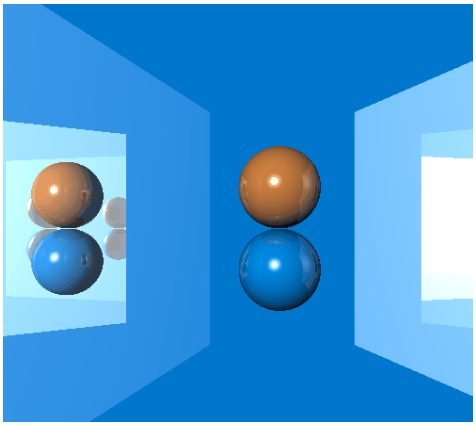


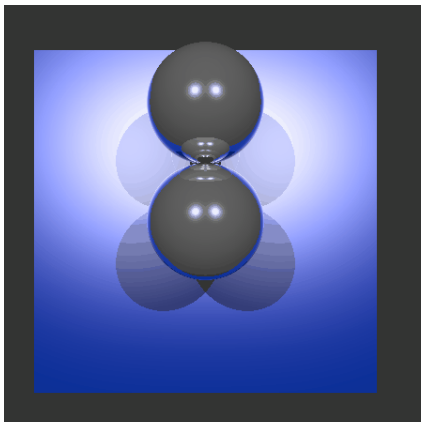
Whitted Ray Tracing

- View dependent approach
- "Trace" rays around scene
  - Can't follow photons from light source to eye
  - Rather, trace from eye into scene
    - Models direct interactions
    - Complex interactions approximated by "ambient" term
  - Basis for modern offline algorithms
    - Films such as Toy Story



Follow Light Through Scene





Forward Ray Tracing

- Lights emit photons, so follow the photons
  - Trace photon paths (rays)
  - Bounce off objects
    - Reflect and refract, attenuate
  - When a ray enter's eye
    - Calculate intersection with viewplane
    - Accumulate color in the pixel
- Expensive, but accurate!
  - Needed to model caustics (see two-pass r. t., sec 10.7)

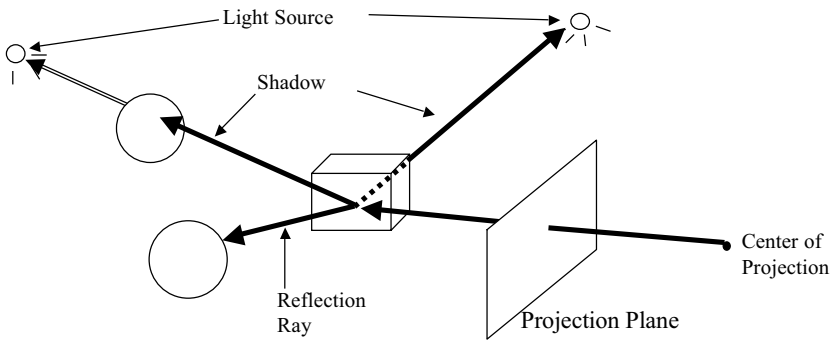
Backward Ray Tracing

- Observation:
  - Only care about light that enters eye
- Trace those rays into the scene directly
  - For each pixel
    - Compute ray from eye though pixel
    - Intersect with objects
    - Compute color using closest object intersection
- "Whitted Ray Tracing" = backward ray tracing

Coloring Pixels

- Use illumination model to determine color at intersection of ray and object
  - Use the same model we've been using
  - Easily compute specular now, using light rays!
- Can do proper transparency
  - Compute color of farther pixel
  - Combine it with transparent object

Reflection, Refraction and Shadows



Diffuse light, refraction, shadows?

- Local diffuse computation, no reflected rays
  - Why not?
- Only sharp shadows
  - Lights are all point lights
- "Ideal" refraction
  - No material defects, unreal looking
    - Helped by distributed ray tracing (sec 10.6)

Radiosity

- Attempt to model complex interactions using heat transfer equations
  - Basic techniques limited to perfect diffusers
    - Diffuse-diffuse interaction
  - No sharp highlights or reflections
- Computes a single radiosity (illumination) for each patch in the scene
- View independent: render patches using a subsequent pass

Model Scene as Diffuse Patchs

- Start with patches that emit light
- Shoot light from each into scene
  - Consider interaction between a light patch and all receiving patches
  - Light is deposited in each of these
- Sort patches according to energy
  - Shoot light from highest stored patch
- Iterate

Issues

- Light may arrive back: iterative process
  - Guaranteed to converge (light lost each time)
- Must consider "shadowing" at each step
- Very sensitive to patch definitions
  - Aliasing if patch shouldn't have constant illum.
  - Can't know ideal patch structure ahead of time
    - This is a key problem! Requires progressive refinement