Global Illumination

Introduction to Computer Graphics
Torsten Möller
Reading

• Chapter 20+24, Shirley+Marschner
• Chapter 29, Hughes, van Dam, et al.
• [Chapter 5.11, 11.1-11.5, Angel]
Limitation of local illumination

- A concrete example:

(a) Global illumination (GIM): can model shadows, multiple inter-object reflections (we used an ambient term in LIM), etc.

(b) Local illumination (LIM): all spheres look the same if no light decay; each rendered independently
Illumination in OpenGL

- Use the Phong local illumination model and consider only direct illumination by light sources
- Self-emission possible
- There is also a global ambient light source
- Flat and smooth (default) shading models are supported
- Geometric primitives go through the graphics pipeline independently
- Visibility is handled using the depth buffer
- User has to simulate global illumination with own implementation
Recall: LIM summary

• The color of a point $p$ due to light source $L$:
  
  $$I = \text{global\_ambient} + \text{self\_emission} + \text{ambient} + \text{attenuation} \times [\text{diffuse} + \text{specular}]$$

• Global ambient component independent of any light

• Emission term is unaffected by any light source and it does not affect any other surface (only used to color the emitter)

• To deal with colors, get $I_R$, $I_G$, $I_B$

• For multiple light sources: sum the intensity $I$’s and normalize
Global illumination

• Try to figure out how to illuminate the whole scene dependently

• Local illumination model is still needed in the process

• Incorporate all lights in the scene, especially reflected light off objects in the scene, not just those directly from light sources
Global illumination model (GIM)

• Brute-force solution:
  – for all light sources
    • for all light rays in all directions
      – trace ray until it hits screen or “leaves the room”
  – computationally intractable

• Two established algorithms:
  – Ray tracing: discretize the image plane
  – Radiosity: discretize the environment
  – Both are simplified treatment of the rendering equation
Images from global illumination

Ray Tracing

Radiosity
GI - models

- Two approaches to its understanding
  - mathematical
  - classification according to types of interaction in the scene
Physically Based Illumination

• Everything so far has been pretty heuristic
• cannot model:
  – wavelength dependent phenomena
  – anisotropic behaviours
  – many other physical phenomena (real physics)
• ongoing research - main early contributions
  – Hanrahan, Krüger (1993)
Some terminology

- **Flux**: the rate at which light energy is emitted; this is measured in watts (W)
- **Solid angle**: angle at the apex of a cone; measured by the area on a sphere centered at the apex that the cone covers
- **Steradian (sr)**: is a unit; 1 sr is the solid angle of a cone that intercepts an area equal to the square of the sphere's radius. E.g., there are $2\pi$ steradians in a hemisphere.
More confusing terminologies

- **radiant intensity**: flux radiated into a unit solid angle, i.e., 1sr, measured in W/sr = intensity from a point source

- **radiance** \( I \): radiant intensity per unit of projected surface area, measured in W/(sr \cdot m^2) = intensity from a surface

- **irradiance** (flux density) \( E \): incident flux per (un-projected) unit surface area, in W/m^2
  
  \[
  E_i = I_i \cos \alpha_i d\omega_i
  \]

\[
E_i = I_i (N \cdot L) d\omega_i = I_i \cos \theta_i d\omega_i
\]
Bidirectional reflectivity (BR)

- **bidirectional reflectivity** $\rho$ : ratio of reflected radiance (intensity) to the incident irradiance (flux density) responsible for it

\[
\rho = \frac{I_r}{E_i}
\]

\[
I_r = \rho I_i (N \cdot L) d\omega_i
\]
One resulting illumination equation

• Typical model: BR is part diffuse and part specular
  \[ \rho = k_d \rho_d + k_s \rho_s \]
  \[ k_d + k_s = 1 \]

• Taking into account light from other surfaces, i.e., add an ambient term, we get an illumination equation:
  \[ I_r = \rho_a I_a + \sum_{1 \leq j \leq n} I_{ij} (N \cdot L_j) (k_d \rho_d + k_s \rho_s) d\omega_i \]

• There are many ways of improving upon this – see [F]
A Typical BRDF
GIM: rendering equation

- Follow the physical law of energy conservation (light is a form of energy)
  - Need to assume a closed environment
  - Imagine an infinite number of rays bouncing around with energy absorption, emission, and reflection

- We reach a steady state, an equilibrium, which we compute, from the rendering equation

- Ray tracing is seen as one way of simplifying the rendering equation
Kajiya’s rendering equation (1986)

- Consider two points $x$ and $x'$ in the scene. We have

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

- Interpretation:
  - Light leaving $x'$ arriving at $x$ has two components:
    - self emission from $x'$, and
    - reflection at $x'$ from other points $x''$
  - Occlusion between $x$ and $x'$ is encoded in $g(x', x)$
Terms in the rendering equation

\[ I(x', x) = g(x', x) \left[ \epsilon(x', x) + \int_{S} \rho(x'', x', x) I(x'', x') dx'' \right] \]

- **I (x’, x):** total light intensity leaving x’ and arriving at x
- **Visibility factor:**
  - \( g(x', x) = 0 \) if x is blocked from x’
  - \( g(x', x) = 1/d^2 \) or other attenuation factor otherwise, where d is the distance between x and x’
- **\( \epsilon(x', x) \):** self emission from x’ in the direction of x
Terms in the rendering equation

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

• \( x'' \): any other point in the scene

• \( \rho(x'', x', x) \): bidirectional reflectance distribution function or BRDF at \( x' \)
  – Ratio between reflected and incident light intensity at \( x' \) with respect to \( x \) (out-going direction) and \( x'' \) (in-coming direction)
  – Specifies how much of \( I(x'', x') \) from \( x'' \) gets reflected in the direction of \( x \)
The rendering equation

\[ I(x', x) = g(x', x) \left[ \varepsilon(x', x) + \int_{S} \rho(x'', x', x) I(x'', x') dx'' \right] \]

- Integral is taken over all surfaces in the scene
- To solve for the unknown I’s, we require:
  - A model of the emitted light \( \varepsilon \) (property of surface)
  - A model for the BRDF \( \rho \) (property of surface)
  - Visibility information \( g \) (determined by geometry)
\[ L_o(x, \omega_o) = L_e(x, \omega_o) + \int_{\Omega} f_r(x, \omega_i, \omega_o) L_i(x, \omega_i) \langle N(x), \omega_i \rangle \, d\omega_i \]
Solving the rendering equation

• Equation is complex
  – no analytic solution in general
  – practical algorithms perform lots of simplification

• If we can solve for the I’s, then we know how much light intensity is received (from all other points) at each point in the scene, i.e., a color can be assigned – this is view independent

• In practice, points become small patches that discretize the scene (patch flat shaded) – a coarse approximation
Simplifying the rendering equation

• Ray tracing – *view dependent*
  – Consider $I(x', x)$ only for $x$ (a pixel) on the image plane
  – Consider $I(x', x)$ only for $x'$ along ray from the eye to pixel $x$
  – Consider only perfect reflections and refractions, thus a very small subset of rays

• Radiosity – *view independent*
  – Discretize the scene into small patches to be flat shaded
  – Each patch is assumed to be perfectly diffuse
  – Project patches onto image plane using computed colors – this is ok for diffuse surfaces (look the same)
Comment on light source

• In ray tracing, we assumed point light sources
  – There is only one shadow ray to consider per light source

• In radiosity, light sources are modeled using patches, so we can have area light sources – more realistic
GI - path notation

• At a point incoming light may be scattered or reflected diffusely or specularly and may have come from a multitude of interactions itself.

• For pairs of surfaces we have:
  – diffuse to diffuse transfer
  – specular to diffuse transfer
  – diffuse to specular transfer
  – specular to specular transfer
GI - path notation

- Radiosity: diffuse to diffuse
- (Whitted) ray-tracing: specular to specular
- String notation (Heckbert 90):
  - L - light source
  - E - eye point
  - Path specify transfer mechanism
  - 4 possibilities: DD, DS, SD, SS
GI - path notation
What Light Paths Do You See?
GI - path notation

- Complete algorithm: $L(D|S)E$
- Local reflection model: $LD|S$
- Hidden surface removal: $LD|SE$ - string of length 1
- Shadow algorithm typically need at least string of length 2
GI - Solutions

• Brute-force solution:
  – for all light sources
    • for all light rays in all directions
      – trace ray until it hits screen or “leaves the room”
    – computationally intractable
  – approximation of brute-force:
    – constrain light-object interaction
    – consider small subset of the rays
GI -(Whitted) ray tracing

- Traces light rays in reversed direction
- hence is view-dependent
- hybrid: includes a local illumination
- for each hit point we include the contribution of the direct light before we continue with the reflected (transmitted) ray
GI -(Whitted) ray tracing

- $\vec{R}_i$: Reflected ray
- $\vec{N}_i$: Surface normal
- $\vec{L}_i$: Shadow ray
- $\vec{T}_i$: Transmitted ray

Viewpoint

Point light source
GI -(Whitted) ray tracing

b) Ray tree for paths

EYE

Recursion Terminates

Eye

mirror sphere

Light ray (no contribution)

no refraction

opaque sphere

c) Contributions from global and local components

LSSE +

LDSE +

no local contribution

© Machiraju/Zhang/Möller
GI -(Whitted) ray tracing

• Includes direct diffuse reflection (LD), but not diffuse-diffuse (DD)
• restricted to specular reflection
• path characterization:
  – LS*E or LDS*E
• rendering equation:
  – integral over sphere simplifies to two directions - light direction and perfect reflected (refracted) ray
GI - Radiosity

• Implements diffuse-diffuse
• no rays - “patches” interact
  – scene needs to be divided into “patches”
• view-independent
  – one pass computes light distribution in the whole scene
  – second pass renders one particular viewpoint
GI - Radiosity

– Start with light source
– distribute light to all patches in scene
– select the patch with the highest contribution and do the same
– repeat

• the interaction of light rays between two patches is averaged and expressed as a “form factor”
• called progressive refinement
GI - Radiosity

- Consider visibility between patches
- in the end - each patch has constant radiosity
- path: LD*E
- rendering equation:

\[ B(x') = \varepsilon(x') + \int_{S} B(x) F(x, x') \, dx \]
GI - Radiosity

• Ray-traced with main light turned off
• typical for indoor scenes
• no diffuse interactions
  - problem!
GI - Radiosity

• Same scene as before - main light turned off
• computed using a radiosity method
• scene accounts for diffuse interactions