Advanced Rendering

Computer Graphics and Visualization

Pedher Johansson
Department of Computer Science
Fall 2016

The Radiosity Model

- At each surface in a model the amount of energy that is given off (Radiosity) is comprised of
  - the energy that the surface emits internally, plus
  - the amount of energy that is reflected off the surface
- The amount of incident light hitting the surface can be found by summing for all other surfaces the amount of energy that they contribute to this surface.

Form Factor (f_{ij})

- the fraction of energy that leaves surface i and lands on surface j
- Dependent of distance and angles.

Notation

n patches numbered 1 to n

b_i radiosity of patch i

a_i area patch i

ρ_i = reflectivity of patch i

f_{ij} form factor = fraction of energy leaving patch j that reaches patch i

- $b_i a_i$ = total intensity leaving patch i
- $e_i a_i$ = emitted intensity from patch i

Radiosity Equation

- Energy balance

$$b_i a_i = e_i a_i + \rho_i \sum f_{ji} b_j a_j$$

- Reciprocity

$$f_{ij} a_i = f_{ji} a_j$$

- Radiosity equation

$$b_i = e_i + \rho_i \sum f_{ij} b_j$$

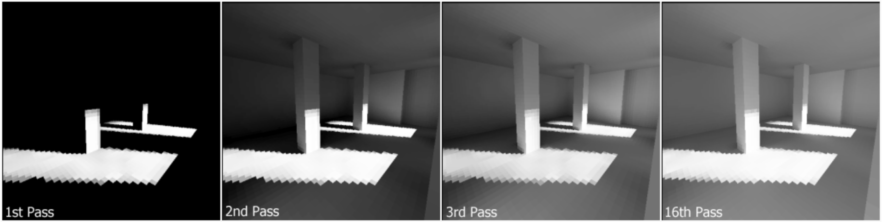
Matrix Form

$$
\begin{aligned}
\mathbf{b} &= [b_i] \\
\mathbf{e} &= [e_i] \\
\mathbf{R} &= [r_{ij}]\ r_{ij} = \rho_i \text{ for } i \neq j,\ r_{ii} = 0 \\
\mathbf{F} &= [f_{ij}]
\end{aligned}
$$

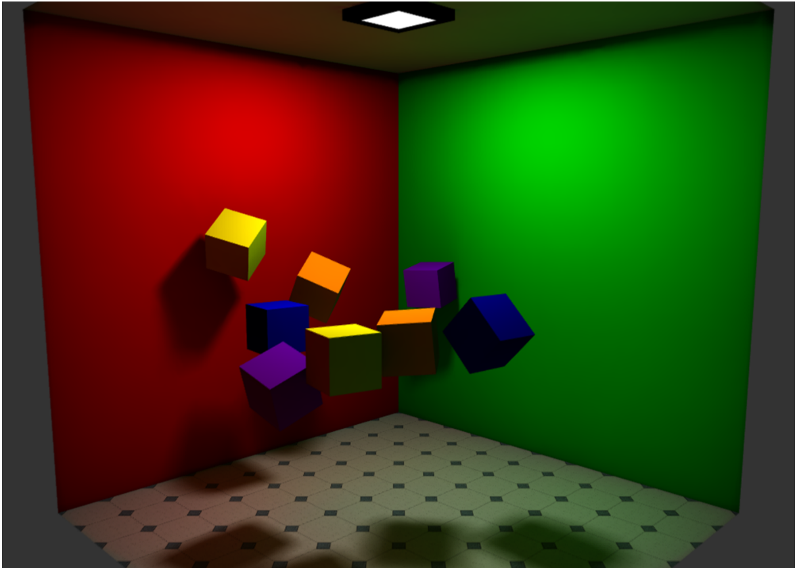
$$b = [I - RF]^{-1} e$$

Solving the Equation

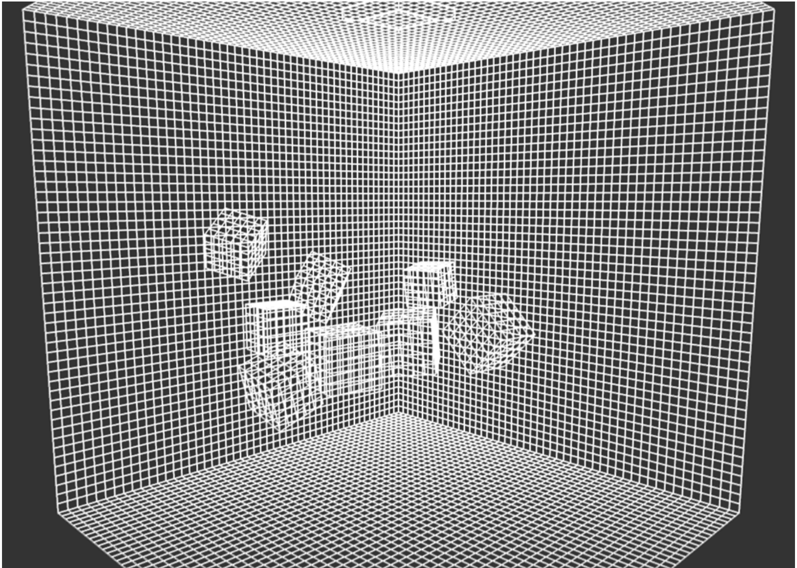
- Factorization not useful since n usually is very large
- Since F is sparse iterative methods usually requires only $O(n)$ operations per iteration.



Rendered Image

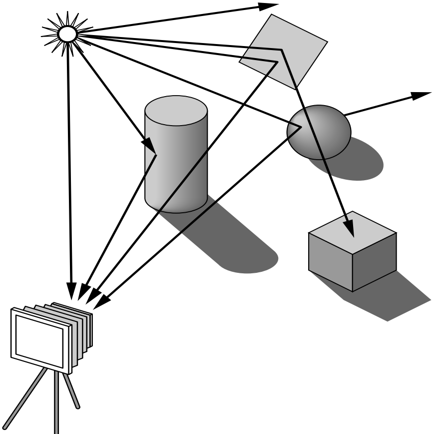


Pathches



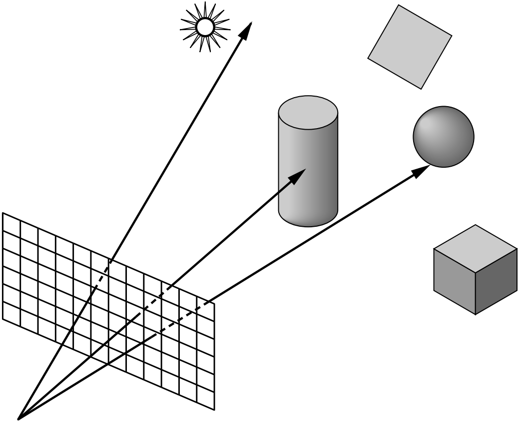
Ray Tracing

- Follow rays of light from a point source
- Most rays do not affect what we see
- Scattering produces many (infinite) additional rays

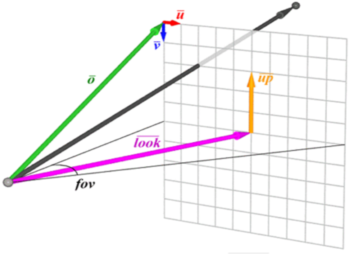


Ray Casting

- Only rays that reach the eye matter
- Need at least one ray per pixel



Calculating the Ray



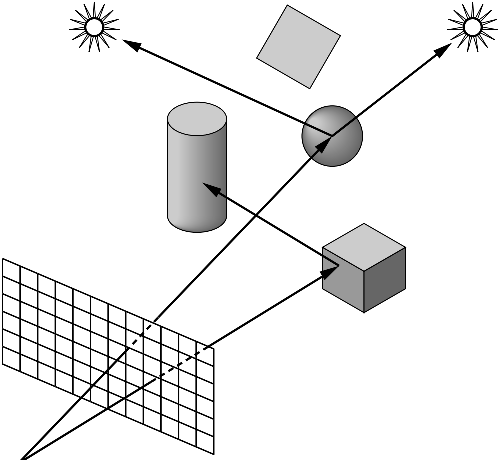
$$\hat{u} = \hat{\text{look}} \times \hat{\text{up}}$$

$$\hat{v} = \hat{\text{look}} \times \hat{u}$$

$$\hat{o} = \hat{\text{look}} \frac{\text{W}}{2\tan(\frac{\text{fov}}{2})} - \frac{W}{2}\hat{u} - \frac{H}{2}\hat{v}$$

Shadow Rays

- Even if a point is visible, it will not be lit unless we can see a light source from that point



Recursion

- Follow reflection and refraction rays to other objects
- Follow shadow, reflection, refraction rays off reflected surfaces

Process is recursive

Diffuse Surfaces

- Theoretically the scattering at each point of intersection generates an infinite number of new rays that should be traced
- In practice, we only trace the transmitted and reflected rays but use the Phong model to compute shade at point of intersection

When to Stop?

- Some light will be absorbed at each intersection
- Ignore rays that go off to infinity
- Count steps

Acceleration

Intersection tests can be as much as 95% of processing time of a ray-tracer

Using trivial rejects

- Bounding hiearchy (bounding volume)
- First intersect the bounding volume, if the ray intersects the BV, then proceed and test the children of the BV.

Antialiasing

- Sampling will always result in aliasing effects
- We could cast several rays per pixel, and jitter the pixels using a noise function and calculate the medium intensity

Photon Mapping

- A fast, global illumination algorithm based on Monte-Carlo method
- Casting photons from the light source, and saving the information of reflection when it hits a surface in the ”photon map”, then render the results

Photon Tracing

- The process of emitting discrete photons from the light sources and tracing them through the scene
- The goal is to populate the photon maps that are used in the rendering pass to calculate the reflected radiance at surfaces

Photon Emission

- A photon's life begins at the light source.
- For each light source in the scene we create a set of photons and divide the overall power of the light source amongst them.
- Brighter lights emit more photons

Photon Scattering

- Emitted photons from light sources are scattered through a scene and are eventually absorbed or lost
- When a photon hits a surface we can decide how much of its energy is absorbed, reflected and refracted based on the surface's material properties
- Use a russian roulette technique to decide whether the photon is reflected or not based on the probability.

Photon Map

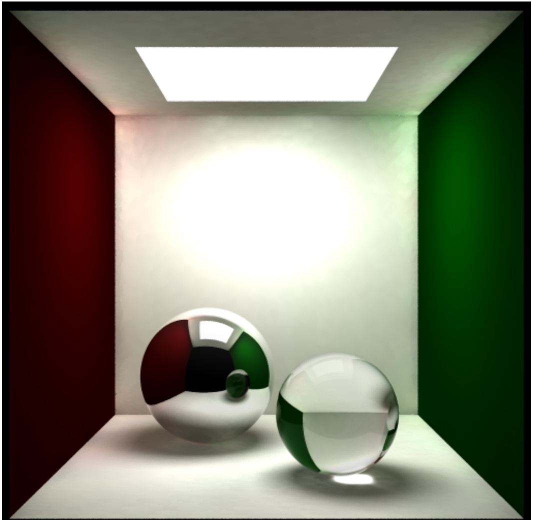
- When a photon makes a diffuse bounce, the ray intersection is stored in memory
  - 3D coordinate on the surface
  - Color intensity
  - Incident direction
  - The data structure of all the photons is called Photon Map

Second Pass – Rendering

- Finally, a traditional ray casting procedure is performed by shooting rays from the camera
- At the location the ray hits the scene, a sphere is created and enlarged until it includes N photons

Samples

- 343 samples per pixel and 10 million photons



Precision

- The precision of the final results depends on
  - the number of photons emitted
  - the number of photons counted for calculating the radiance

Particle Systems

- Most important of procedural methods
- Used to model
  - Natural phenomena (Clouds, Terrain, Plants)
  - Crowd Scenes
  - Real physical processes

Newtonian Particle

- Particle system is a set of particles
- Each particle is an ideal point mass
- Six degrees of freedom
  - Position
  - Velocity
- Each particle obeys Newtons' law

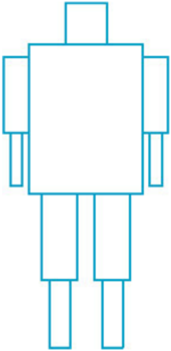
$$f = ma$$

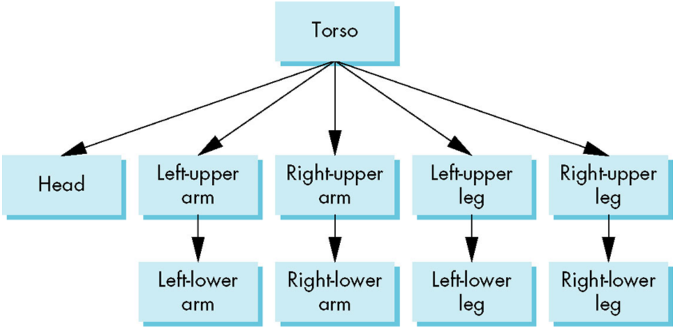
Solution of ODEs

Forces

- Independent Particles $O(n)$
  - Gravity
  - Wind forces
  - calulation
- Coupled Particles $O(n)$
  - Meshes
  - Spring-Mass Systems
- Coupled Particles $O(n^2)$
  - Attractive and repulsive forces

Humanoid Figure

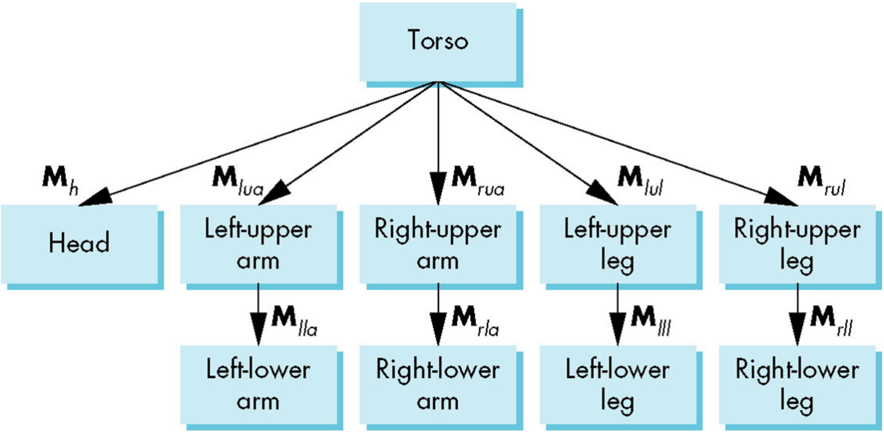




Example of Rotations

- The position of the figure is determined by 11 joint angles (two for the head and one for each other part)
- Display of the tree requires a graph traversal
  - Visit each node once
  - Display function at each node that describes the part associated with the node, applying the correct transformation matrix for position and orientation

Tree with Matrices



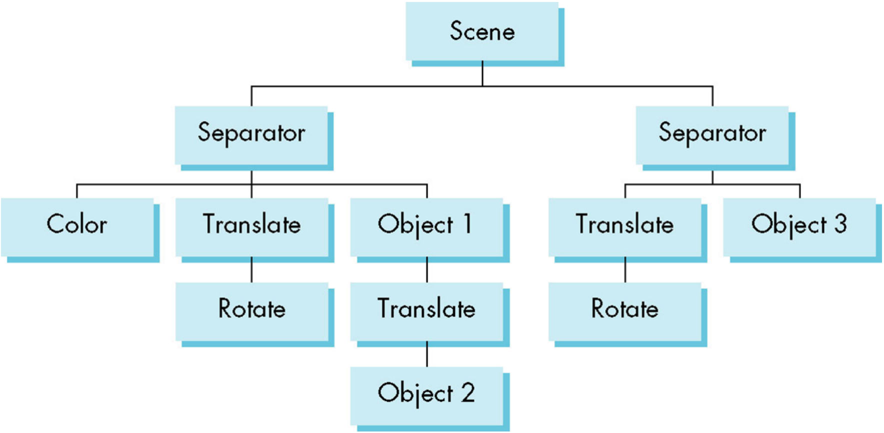
Stack-based Traversal

- Set model-view matrix to M and draw torso
- Set model-view matrix to MM_h and draw head
- For left-upper arm need MM_{lua} and so on
- Rather than recomputing MM_{lua} from scratch or using an inverse matrix, we can use the matrix stack to store M and other matrices as we traverse the tree

Scene Descriptions

- If we recall figure model, we saw that
  - We could describe model either by tree or by equivalent code
  - We could write a generic traversal to display
- If we can represent all the elements of a scene (cameras, lights,materials, geometry), we should be able to show them in a tree
  - Render scene by traversing this tree

Scene Graphs



Scene Descriptions

- There are a few standard API's available
  - VRML (Virtual Reality Modeling language)
  - Java3D
  - Open Scene Graph
- No one is dominant
- A simple API is shown in E. Angel, 10.9