Advanced Graphics

Approximate Global Illumination

Part 2
Precomputed radiance transfer

**Enlighten**

Instant radiosity

Light propagation volumes

Voxel cone tracing
Overview

Real-time (commercial) adaptation of radiosity
   – Several compromises to increase speed

Only used for indirect light
   – Direct lighting is done normally

Computes indirect illumination on the CPU
   – While the GPU is doing other things
Overview

Steps

– Sample light input from the GPU
– Project on low detail environment mesh
– Compute radiosity
– Transfer radiosity textures to the GPU
– Sample radiosity on high detail mesh
Result

Direct light
(computed normally)

Final image
Sample Light

Point sample to reduce bandwidth
  – Use previous frame
  – Also includes indirect light
Radiosity Mesh

Low detail mesh with large patches
Compute Radiosity

Probably using non-diffuse hierarchical radiosity
  – No details published
  – Links can be precalculated

Only does one bounce
  – Next bounce is automatically done next frame
Transfer to GPU

Uploads a new radiosity lightmap
  – GPU does not wait for the CPU
  – GPU just uses newest lightmap available

Lightmap is very small

Example lightmap (not used for the other images)
Sample Radiosity

Sample radiosity on the high detail mesh
  – Correct for different normal
  – Use directional information
Combine

Combine with the direct lighting and textures
Dynamic Objects

What about moving/animating objects?
  – Are not included in the precalculation step

Place *light probes* in the scene
  – Captures the indirect illumination in space
    • Not on a surface
  – Dynamic objects sample from near light probes
    • Passive role in the global illumination
Advantages

Amortize computation cost over multiple frames
  – Not necessary to calculate it every frame
    • (Possible in most techniques)

Simulates global illumination with many bounces
  – But there is an update delay

Supports all types of light sources
Disadvantages

Computation on the CPU
  – Introduces overhead and latency
  – Technique can be adapted for compute shaders

Dynamic objects don’t reflect light
  – (Can be done for a few large rigid objects)

Input lighting is sampled from the view
  – Totally incorrect in some situations
Usage

Commercial product by Geomerics
  – Not all details are public

Used in Battlefield 3, Medal of Honor Warfighter and more
Video
Precomputed radiance transfer
Enlighten
Instant radiosity
Light propagation volumes
Voxel cone tracing
Instant Radiosity

Totally unrelated to the radiosity techniques!
  – Horribly confusing name
  – More similarity with photon mapping

In theory not an approximation
  – The real-time extensions are approximations
Two phases

– Shoot and bounce photons through the scene
  • Stochastically, just like photon mapping
  • Create a Virtual Point Light (VPL) at each intersection

– Gather light from the VPL’s
  • First, render shadow maps for each VPL
  • Render the scene, lit with all VPL’s
Shooting Phase

Shooting photons requires ray tracing
– Not fast enough
– Use rasterization!

Use a *Reflective Shadow Map* (RSM) to create the VPL’s
Reflective Shadow Maps

Render from the light source perspective
- Position (depth)
- Normal
- Outgoing color (light * albedo, without shading)

Very similar to the G-buffer
Reflective Shadow Maps

Pixels in the RSM are VPL’s

- RSM provides the information needed for the VPL
- Combine pixels to reduce the number of VPL’s
- Creates around 400 VPL’s

Much faster than ray tracing

- But only does one bounce
Create a shadow map for each VPL
  – Very expensive!

Sacrifice shadow map accuracy for speed
  – Hardly noticeable

Use Imperfect Shadow Maps (ISM)
Imperfect Shadow Maps

Scene is represented as points
  – Render them into tiny shadow maps

All shadow maps are located in one texture
  – Render all shadow maps simultaneously
Imperfect Shadow Maps

Fill gaps by averaging nearby depth values

Use valid depth values to create a mipmap pyramid

Fill unknown values using the mipmap pyramid
Multiple Bounces

Can be extended with multiple bounces

Create additional VPL’s from the ISM’s
  – Using Imperfect Reflective Shadow Maps
Disadvantages

- Sample many shadow maps during rendering
- Accuracy problems
Usage

Not used in games

– As far as I know
Video
Precomputed radiance transfer

Enlighten

Instant radiosity

Light propagation volumes

Voxel cone tracing
Light Propagation Volumes

Move light through a 3D voxel grid

– Propagating cell by cell
– Use spherical harmonics for directional storage
Overview

Steps

– Render RSM’s
  • Low resolution (e.g. 128 x 128)
– Inject light into voxel grid
– Propagate light through voxel grid
– While rendering, sample from voxel grid
Light Injection

Each RSM pixel is a VPL

– Convert light distribution to spherical harmonics

Render to the 3D voxel grid

– Render a point to select the cell
– VPL position within the voxel cell is ignored
Propagate Light

Propagate light to the 6 direct neighbors
– Not to the diagonal neighbors
Propagate Light

How do incorporate occlusion?
- Detect then the light hits an object

Use a separate voxel representation of the scene
- Called *Geometry Volume* (GV)
- Each cell has an occlusion value
- (Partially) blocks light from passing through
Propagate Light

No propagation (only injection)
Propagate Light

After 4 iterations
Propagate Light

After 6 iterations
Propagate Light

After 8 iterations
Sampling Light

Sample from the voxel grid (LPV) at the desired location
  – Interpolate spherical harmonics coefficients
Geometry Volume

Geometry volume should be dynamic
  – Allows for dynamic objects
  – Recreate GV every time

Use available RSM’s
  – From the light sources
  – From the camera (deferred shading G-buffer)
Geometry Volume

Inject occlusion into the geometry volume
  – For each RSM pixel
  – Increase occlusion in the respective cell

Use spherical harmonics to store directional occluding power
  – More occlusion in the direction of the normal
The geometry volume is an approximation
– Results in accuracy issues
Cascades

Propagation is inefficient over large distances
– And the voxel grid requires a lot of memory

Use cascaded light propagation volumes
– Exactly like cascaded shadow maps
– Common setup: 3 cascades with each $32 \times 32 \times 32$ voxels
Volumetric Lighting

Easy to do volumetric lighting

– Lighting information in the volume is known
Limitations

Directional inaccuracy
  – Diagonal propagation is not accurate
  – Light is smeared out as a result

Occlusion inaccuracy
  – Due to the coarse Geometry Volume

One indirect bounce
  – Can be extended to include multiple bounces
Usage

Popular global illumination technique
– Used in many titles

Debuted in Crysis 2
Comparison with photon mapping

Light Propagation Volumes

Photon Mapping
Video
Precomputed radiance transfer

Enlighten

Instant radiosity

Light propagation volumes

**Voxel cone tracing**
Sparse Voxel Octree

Scene is represented as *Sparse Voxel Octree* (SVO)

- Stores occlusion values for geometry

Light is inserted into the SVO

- But no propagation

- Instead uses *Voxel Cone Tracing* to gather light
Sparse Voxel Octree

Octree structure to store voxels
  – Combine empty blocks of voxels
  – Commonly used for ray tracing through voxel grids
Sparse Voxel Octree

Features

– Allows for hardware texture interpolation when sampling a tree node
– Parent nodes contain the average of the child nodes
– Additional links to spatial neighbors
Sparse Voxel Octree
Overview

Steps

– Inject light in the SVO
– Calculate average light in the SVO parent nodes
– Gather light using Voxel Cone Tracing
Inject Light

Render a RSM for each light source

For each pixel (VPL)
  – Find the leaf node in the SVO
  – Accumulate the light
    • Store (simplified) directional distribution
Pre-integration

Propagate light through parents
– Each parent is the average of its children

Functions as a pre-integration step
– Similar to mipmapping
– Optimizes gathering over large areas
Gather Light

Gather light from all directions
  – Similar to the final gather step in photon mapping

Trace multiple cones
  – Using Voxel Cone Tracing
  – Commonly 3 or 5 cones
Voxel Cone Tracing

Ray march through the SVO
  – Taking samples at higher levels in the tree

Use quadrilinear interpolation
  – Interpolate between two detail levels
Voxel Cone Tracing

Accumulate occlusion along the ray

Large distance between large samples
  – Less samples when the cone is wider
  – Very wide cones are cheap!
    • But the accuracy suffers
Specular

Use a narrow cone for (glossy) specular lighting
Specular

Approximate Global Illumination
Dynamic Objects

Dynamic objects are supported
– Create their SVO representation each frame
– No other changes necessary
Limitations

Interpolation in the SVO includes too much area
  – This problem for shadow maps was solved using summed-area tables

Single bounce indirect light only
  – Relies on final gather

Cannot amortize gathering over multiple frames
  – All voxel cone tracing is done each frame
Usage

New technique
  – Published late 2011

Implemented in Unreal Engine 4
  – No UE4 games released yet
Comparison

Better quality than light propagation volumes
– But still a coarse approximation

Light propagation volumes
Voxel cone tracing
Reference
CONCLUSION

Precomputed radiance transfer
Enlighten
Instant radiosity
Light propagation volumes
Voxel cone tracing
Observations

Very inaccurate

– Only low frequency indirect lighting
  • No sharp changes
– Direct lighting is done separately

Independence of geometric complexity

– Voxelization
– Point-based representation (ISM)
– Low poly mesh (Enlighten)
Is accurate real-time global illumination possible?

Path tracing has shown impressive results

The guest lecture will be given by Jacco Bikker
– One of the leading researchers in real-time path tracing
Real-time path tracing

(Path tracing noise is very difficult for video compression)