Global Illumination of Point Models

Rhushabh Goradia: Fifth Annual Progress Seminar
http://www.cse.iitb.ac.in/~rhushabh

Guide: Prof. Sharat Chandran
Date: 30/09/09
Problem Statement

➢ To compute a global illumination (GI) solution for complex scenes represented as point models
Problem Statement

➢ To compute a **global illumination (GI)** solution for complex scenes represented as point models

- Diffuse Effects
- Specular Effects
- Color Bleeding
- Soft Shadows
- Reflections and Refractions
- Caustics

(GAS 08, ICVGIP)
Problem Statement

➢ To compute a global illumination (GI) solution for complex scenes represented as point models

- Point Models: Discrete representation of a continuous surface
- No connectivity information between points
- Each point has certain attributes, e.g. Co-ordinates, normal, color
Motivation

- Virtual walkthroughs -- The Digital heritage project (Microsoft Research, India)

- Preserving monuments and Renovation

- E-Museums!
Challenges

Problem Statement: To compute a global illumination solution for complex scenes represented as point models.

- Hard to segment entities: Inhibiting Surface Re-construction
- GI Algorithm should handle all sort of surfaces: Diffuse and Specular
- Expensive!
- Fast Ray-Tracer
Challenges

- **Problem Statement:** To compute a global illumination solution for **complex scenes** represented as point models

- Hard to segment entities: Inhibiting Surface Re-construction

- GI Algorithm should handle all sort of surfaces: Diffuse and Specular

- Expensive!

- **Pioneers to give a complete GI package!**
Global Illumination of Point Models

- Diffuse Effects
- Specular Effects
Plan

- Introduction – *Problem Definition*
- Diffuse Effects – *Overview*
- Specular Effects – *Details*
- Results
- Wrap-up
Plan

➤ Introduction – *Problem Definition*

➤ Diffuse Effects – *Overview*

➤ Specular Effects – *Details*

➤ Results

➤ Wrap-up
Diffuse Interactions

- A Diffuse scene
Diffuse Interactions

- Source emits light in all directions
Diffuse Interactions

- Follow one particular ray
- LD path
Diffuse Interactions

- Diffuse BRDF
- High Absorption
Diffuse Interactions

- LDD/LD\(^+\) path
Global Illumination of Point Models

- Diffuse Effects
- Specular Effects

Radiosity based GI Algorithm
Global Illumination of Point Models

Diffuse Effects

Specular Effects

Radiosity based GI Algorithm

A N-body problem
Global Illumination of Point Models

Diffuse Effects

Radiosity based GI Algorithm

Specular Effects

A N-body problem
Global Illumination of Point Models

Diffuse Effects

- Radiosity based GI Algorithm

Specular Effects

- A N-body problem
Global Illumination of Point Models

- Diffuse Effects
- Specular Effects

- Radiosity based GI Algorithm
- A $N$-body problem
- Size of Point Models: Hundreds of Thousands
  Not Practical to Implement with such high time complexity
Fast Multipole Method (FMM) reduces the $O(N^2)$ time complexity to $O(N)$

Follows Factorization and a Hierarchical Structure
Fast Multipole Method (FMM) reduces the $O(N^2)$ time complexity to $O(N)$

Follows Factorization and a Hierarchical Structure

Work done in my 2nd year of PhD
Visibility calculation between point pairs is essential to give correct GI results as a point receives energy from other points only if it is visible – $O(N^3)$
Visibility Map (V-Map) gives a view-independent, hierarchical visibility solution for any given scene.

- Work done in my 3rd year of PhD
- [GKSD 07], Visibility Map for Global Illumination in Point Clouds, GRAPHITE 2007
Global Illumination of Point Models

- Diffuse Effects
  - V-Map
  - FMM

- Specular Effects

... but still not fast enough!

Around 20x speed-up

GPU
Global Illumination of Point Models

Diffuse Effects
- V-Map
- FMM

Specular Effects

... but still not fast enough!

Around 20x speed-up

GPU

➢ Running times reduced from hours to minutes
Global Illumination of Point Models

Diffuse Effects
- V-Map
- FMM

Specular Effects

➢ Work done in my 4th year of PhD
➢ [GAS 08], GPU-based Hierarchical Computations for View Independent Visibility, ICVGIP 2008

Around 20x speed-up
Plan

- Introduction – *Problem Definition*
- Diffuse Effects – *Overview*
- Specular Effects – *Details*
- Results
- Wrap-up
Plan

➢ Introduction – *Problem Definition*

➢ Diffuse Effects – *Overview*

➢ Specular Effects – *Details*
  ➢ *How it affects diffuse interactions?*
  ➢ *Details on generating specular effects*

➢ Results

➢ Wrap-up
Diffuse Interactions

- LDD/LD⁺ path
Diffuse-Specular Scene
Diffuse-Specular Scene

- All possible paths taken up by light
  - LDD/LD^+ 
  - LSD/LS^+D 
  - LDS/LD^+S^+D 
  - LSS
All possible paths taken up by light

- LDD/LD^+
- LSD/LS^+D
- LDS/LD^+S^+D
- LSS
Diffuse-Specular Scene

- All possible paths taken up by light
  - LDD/LD^+
  - LSD/LS^+D
  - LDS/LD^+S^+D
  - LSS

- Light received through LSD path must be distributed during diffuse inter-reflections
Diffuse-Specular Scene

- All possible paths taken up by light:
  - \( \text{LDD/}LD^+ \)
  - \( \text{LSD/}LS^+D \)
  - \( \text{LDS/LD}^+S^+D \)
  - \( \text{LSS} \)

- Light received through LSD path must be distributed during diffuse inter-reflections

- Pre-process and Store
Diffuse-Specular Scene

- All possible paths taken up by light
  - LDD/LD⁺
  - LSD/LS⁺D
  - LDS/LD⁺S⁺D
  - LSS

- Light received through LSD path must be distributed during diffuse inter-reflections

- Pre-process and Store

- LS⁺D -- **Caustics!**
All possible paths taken up by light:
- LDD/LD+
- LSD/LS^D
- LDS/LD^S^D
- LSS

LSS similar to LS^D

Pre-processed Caustics!
Diffuse-Specular Scene

- All possible paths taken up by light
  - $LDD/\text{LD}^+$
  - $LSD/\text{LS}^+\text{D}$
  - $\text{LDS/}\text{LD}^+\text{S}^+\text{D}$
  - $LSS$

- **Time-consuming**

- Energy transfer too low!

- We can do it, but temporarily ignore these paths for the purpose of efficiency.
Diffuse-Specular Scene

- **Diffuse case**

- **Specular case**
Diffuse-Specular Scene

- **Diffuse case**

- **Specular case**

- During FMM transfers, surfaces $A$ and $B$ will still be invisible to $L$!
Fusing Together: **Diffuse and Specular Effects**

- $LS^+D$ handled by caustics
- FMM takes care of $LD^+$ path
- FMM ignores contributions from specular splats ($LS^+D$)
- FMM does not transfer energy to specular splats ($LD^*S^+$)
Fusing Together: Diffuse and Specular Effects

- $LS^+D$ handled by caustics
- FMM takes care of $LD^+$ path
- FMM ignores contributions from specular splats ($LS^+D$)
- FMM does not transfer energy to specular splats ($LD^*S^+$)

- *What about view-dependence?* -- $L(S|D)^*E$
View-Independence v/s View-Dependence

\[ L(S|D)^*E \]
Plan

- Introduction – *Problem Definition*

- Diffuse Effects – *Overview*

- Specular Effects – *Details*
  - *How it effects diffuse interactions?*
  - *Details on generating specular effects*

- Results

- Wrap-up
Global Illumination of Point Models

Diffuse Effects

- V-Map
- FMM

Specular Effects

- Work done in my 5th year of PhD

Around 20x speed-up

GPU
Global Illumination of Point Models

Diffuse Effects
- V-Map
- FMM

Specular Effects
- Caustics
- Reflections
- Refractions

GPU
Global Illumination of Point Models

Diffuse Effects
- V-Map
- FMM

Specular Effects
- Caustics
- Reflections
- Refractions

GPU

View-Dependent
Global Illumination of Point Models

Diffuse Effects
- V-Map
- FMM

Specular Effects
- Caustics
- Reflections
- Refractions
  - Coherent Rays
  - Super-Sampling

Texture Cache memory
- Fast Octree Traversal for Ray-Tracing
- Ray-Splat Intersections
- Normal Field on Splats
- Photon Generation
- Kd-Trees
Global Illumination of Point Models

- Diffuse Effects
  - V-Map
  - FMM
- Specular Effects
  - Caustics
  - Reflections
  - Refractions

- Fast Octree Traversal for Ray-Tracing
- Ray-Splat Intersections
- Normal Field on Splats
- Coherent Rays Super-Sampling
- Texture Cache memory
Paths of Photons: Caustics
Paths of Photons: Caustic Example
Caustics on Point Models: Phases

- Generate caustic photons at the light source
- Trace photons ($LS^+D$) in the scene. Deposit them on diffuse surfaces after at least one hit from a specular surface.
  - Traverse photons through octree
  - Use Ray-Splat intersection
- Form a $kd$-tree on the caustic map for fast photon search during ray-tracing
- Render using ray-tracing (View-dependent)
Global Illumination of Point Models

Diffuse Effects
- V-Map
- FMM
- Texture Cache memory

Specular Effects
- Caustics
- Reflections
- Refractions
- Photon Generation
  - Kd-Trees
- Fast Octree Traversal for Ray-Tracing
  - Ray-Splat Intersections
  - Normal Field on Splats
- Coherent Rays
  - Super-Sampling

GPU
Caustics: Photon Generation

- The photons emitted have distribution corresponding to the distribution of emissive power of the light source
The photons emitted have distribution corresponding to the distribution of emissive power of the light source.

\[ P_{\text{photon}} = \frac{P_{\text{light}}}{n_e} \]

- \( P_{\text{photon}} \) = Power per photon
- \( P_{\text{light}} \) = Emissive power of light
- \( n_e \) = Number of Photons
The photons emitted have distribution corresponding to the distribution of emissive power of the light source

\[ P_{\text{photon}} = \frac{P_{\text{light}}}{n_e} \]

- The number of photons \( n_e \) is pre-defined (e.g. 4,00,000)
- The emissive power of light source \( P_{\text{light}} \) is also taken as input
Caustics: Usage of V-Map

- Caustics paths: Only $LS^+D$
- Send rays only to visible specular leaf nodes (leaves which contain specular points)
- Use visible links of light source (V-Map)
- Saves time as we do not search for any caustic generators

Remember – Scene is divided into an octree!
Global Illumination of Point Models

Diffuse Effects
- V-Map
- FMM

Specular Effects
- Caustics
- Reflections
- Refractions
  - Photon Generation Kd-Trees
  - Fast Octree Traversal for Ray-Tracing
    - Ray-Splat Intersections
    - Normal Field on Splats
  - Coherent Rays Super-Sampling

GPU
- Texture Cache memory
Global Illumination of Point Models

Diffuse Effects
- V-Map
- FMM

Specular Effects
- Caustics
- Reflections
- Refractions

Photon Generation
- Kd-Trees

GPU
- Texture Cache memory
- Fast Octree Traversal for Ray-Tracing
- Ray-Splat Intersections
- Normal Field on Splats
- Coherent Rays
- Super-Sampling
Caustics: Number of Photons To Shoot Per Leaf

- Distance from centroid of the disk to center of the splat
- Radius of the splat

If “D” is the distance from centroid of the disk to center of a splat plus the splat’s radius, the radius “r” of the disk is set to the “maximum D”
Caustics: Number of Photons To Shoot Per Leaf

>We find the *solid angle* subtended by this average disk $D$ of *leaf* $A$ from the light source.

---

- $l$ -- Distance from centroid of the disk to center of the splat
- $l$ -- Radius of the splat

If “$D$” is the distance from centroid of the disk to center of a splat plus the splat’s radius, the radius “$r$” of the disk is set to the “maximum $D$.”
Caustics: Number of Photons To Shoot Per Leaf

We find the **solid angle** subtended by this average disk $D$ of *leaf A* from the light source

$$S.A. = \frac{(Area \times \cos \theta)}{d^2}$$
Caustics: Number of Photons To Shoot Per Leaf

We find the *solid angle* subtended by this average disk $D$ of *leaf* $A$ from the light source

$$S.A. = \frac{(Area \times \cos \theta)}{d^2}$$

$$S.A.(A) = \frac{(\pi r^2 \times DP)}{d^2}$$

where $DP = dot(Dist, n)$
Caustics: Number of Photons To Shoot Per Leaf

We find the solid angle subtended by this average disk $D$ of leaf $A$ from the light source.

$$S.A.(A) = \frac{\pi r^2 \times DP}{d^2}$$

$$N_{\text{photons}}(A) = n_e \times S.A.(A) / 4\pi$$
Caustics: Number of Photons Per Splat

\[ P.P.S(A) = \frac{N_{\text{photons}}(A)}{N.P(A)} \]

where,

\[ P.P.S(A) = \text{Photons per splat in } A \]
\[ N_{\text{photons}}(A) = n_e * S.A.(A) / 4\pi \]
\[ N.P.(A) = \text{Number of points in } A \]

- The remaining \( N_{\text{photons}}(A) \% N.P.(A) \) number of photons are distributed randomly to splats in A
**Caustics: Number of Photons Per Splat**

\[
P.P.S(A) = \frac{N_{\text{photons}}(A)}{N.P.(A)}
\]

where,

\[
P.P.S(A) = \text{Photons per splat in } A
\]

\[
N_{\text{photons}}(A) = n_e \times S.A.(A) / 4\pi
\]

\[
N.P.(A) = \text{Number of points in } A
\]

- The remaining \(N_{\text{photons}}(A) \% N.P.(A)\) number of photons are distributed randomly to splats in \(A\).

- Use **random sampling** on each splat of \(A\) to get photon hit locations, the number of samples on each splat equal to \(P.P.S\) of that splat.

"Leaf A"
Ray-Splat Intersections: Normal Field

- Photons sent to specular leaves from light source
Generate new reflective or refractive rays
Ray-Splat Intersections: Normal Field

- We need normal at the intersection point on the splat
We need normal at the intersection point on the splat.

- [LMR 07], Splat Based Ray-Tracing of Point Clouds, Journal of WSCG, 2007

Pre-process and Store
We need normal at the intersection point on the splat.

Average disk of a node with center “c”
Tracing Caustic Photons
Ray-Splat Intersections: Minimum Delta (▵)

Ray R  Reflected Ray R′

Ray R

Reflected Ray R′

LS

LS

L

LSD

LSS

S_j

S_{j+1}

i

i+1
Ray-Splat Intersections: **Minimum Delta** ($\Delta$)

- We set $\Delta$ to 0.05
We perform a standard Ray-Disk Intersection to find the intersecting splat.
Store the view-independent caustic map as a *kd-tree*.

Storing Caustic Photons: KD-Tree
Storing Caustic Photons: KD-Tree

- Store the view-independent caustic map as a *kd-tree*

- Searching for nearest photons in the *kd-tree* is a run time step, while storing them is pre-computation

- We use the Approximate Near Neighbor Search (ANN) library for searching and constructing the *kd-tree*
Global Illumination of Point Models

- Diffuse Effects
  - V-Map
  - FMM

- Specular Effects
  - Caustics
  - Reflections
  - Refractions
  - Photon Generation
    - Kd-Trees

- GPU
  - Texture Cache Memory
  - Fast Octree Traversal for Ray-Tracing
  - Ray-Splat Intersections
  - Normal Field on Splats
  - Coherent Rays
  - Super-Sampling
Global Illumination of Point Models

Diffuse Effects
- V-Map
- FMM

Specular Effects
- Caustics
- Reflections
- Refractions
  - Super-Sampling
  - Coherent Rays

Photon Generation
- Kd-Trees

FAST OCTREE TRAVERSAL FOR RAY-TRACING
- Ray-Splat Intersections
- Normal Field on Splats

Texture Cache memory

GPU
Tracing Caustic Photons

Fifth Annual Progress Seminar, 2009

Rhushabh Goradia, Sharat Chandran
Tracing Photons: Similar To Ray-trace Rendering

- **Diffuse Splat**
- **Specular Reflective Splat**
- **Specular Refractive Splat**
- **Eye/Camera**
- **View Plane**
Global Illumination of Point Models

Diffuse Effects
- V-Map
- FMM

Specular Effects
- Caustics
- Reflections
- Refractions
- Photon Generation
- Kd-Trees
- Coherent Rays
- Super-Sampling
- Ray-Splat Intersections
- Normal Field on Splats
- Fast Octree Traversal for Ray-Tracing

GPU
- Texture Cache memory
Ray-trace Rendering: Octree Traversal

Ray NOT hitting the Octree
Ray hitting an Empty node
Ray hitting a filled leaf
No ray-splat intersections

Δ -- Delta increment to the ray

\( \hat{i} \) — Find leaf containing intersection point “\( i \)”. If empty (node highlighted in red), proceed to the next node along the ray direction (node highlighted in blue), else, do ray-splat intersections (node highlighted in green)
Ray-trace Rendering: Octree Traversal

- Is ray hitting Octree? -- *Ray-Box Test*

- Is ray hitting Octree?
  - Ray NOT hitting the Octree
  - Ray NOT hitting an Empty node
  - Ray hitting a filled leaf
  - Ray hitting an Empty node

- No ray-splat intersections

- **Δ** -- Delta increment to the ray

Δ − Find leaf containing intersection point “i”. If empty (node highlighted in red), proceed to the next node along the ray direction (node highlighted in blue), else, do ray-splat intersections (node highlighted in green)
Ray-trace Rendering: Octree Traversal

- If Yes, get Intersection point

- Ray NOT hitting the Octree
- Ray NOT hitting an Empty node
- Ray NOT hitting a filled leaf
- No ray-splat intersections

- Δ -- Delta increment to the ray
- i — Find leaf containing intersection point “i”. If empty (node highlighted in red), proceed to the next node along the ray direction (node highlighted in blue), else, do ray-splat intersections (node highlighted in green)
Ray-trace Rendering: Octree Traversal

- Increment by ▲

- Ray NOT hitting the Octree
- Ray NOT hitting an Empty node
- Ray hitting a filled leaf
- No ray-splat intersections

View Plane

Δ -- Delta increment to the ray

\[ i \] — Find leaf containing intersection point “i”. If empty (node highlighted in red), proceed to the next node along the ray direction (node highlighted in blue), else, do ray-splat intersections (node highlighted in green).
Ray-trace Rendering: Octree Traversal

- Find leaf containing point \( i \)

\( i \rightarrow \) Find leaf containing intersection point “\( i \)”. If empty (node highlighted in red), proceed to the next node along the ray direction (node highlighted in blue), else, do ray-splat intersections (node highlighted in green)

\( \triangle \) -- Delta increment to the ray

Eye/Camera

View Plane

Ray NOT hitting the Octree

Ray hitting an Empty node

Ray hitting a filled leaf

No ray-splat intersections
Ray-trace Rendering: Octree Traversal

- Find the exit plane

---

- Δ -- Delta increment to the ray

\[ \Delta \rightarrow \] Find leaf containing intersection point "\( i \)". If empty (node highlighted in red), proceed to the next node along the ray direction (node highlighted in blue), else, do ray-splat intersections (node highlighted in green).
Ray-trace Rendering: Octree Traversal

1. Find the new intersection point

- Ray NOT hitting the Octree
- Ray NOT hitting an Empty node
- Ray hitting a filled leaf
- No ray-splat intersections

Triangles (Δ) -- Delta increment to the ray

Δ — Find leaf containing intersection point “i”. If empty (node highlighted in red), proceed to the next node along the ray direction (node highlighted in blue), else, do ray-splat intersections (node highlighted in green).
Ruby trace Rendering: Octree Traversal

- Increment by ▲

Ray NOT hitting the Octree
Ray hitting an Empty node
Ray hitting a filled leaf

Δ -- Delta increment to the ray

i — Find leaf containing intersection point “i”. If empty (node highlighted in red), proceed to the next node along the ray direction (node highlighted in blue), else, do ray-splat intersections (node highlighted in green)
Ray-trace Rendering: **Octree Traversal**

- **Eye/Camera**
- **View Plane**

- Ray NOT hitting the Octree
- Ray hitting an Empty node
- Ray hitting a filled leaf
- No ray-splat intersections

\[ \Delta \text{ -- Delta increment to the ray} \]

\[ \hat{i} \text{ -- Find leaf containing intersection point \"i\". If empty (node highlighted in red), proceed to the next node along the ray direction (node highlighted in blue), else, do ray-splat intersections (node highlighted in green)} \]
**Ray-trace Rendering: Octree Traversal**

- Do Ray-Splat Intersections

---

- Ray NOT hitting the Octree
- Ray hitting an Empty node
- Ray hitting a filled leaf

---

- **Δ** -- Delta increment to the ray
- `i` — Find leaf containing intersection point “`i`”. If empty (node highlighted in red), proceed to the next node along the ray direction (node highlighted in blue), else, do ray-splat intersections (node highlighted in green)
Ray-trace Rendering: Octree Traversal

- Find the exit plane

Δ -- Delta increment to the ray

ı − Find leaf containing intersection point "ı". If empty (node highlighted in red), proceed to the next node along the ray direction (node highlighted in blue), else, do ray-splat intersections (node highlighted in green)
Ray-trace Rendering: Octree Traversal

- Find the new intersection point

- Delta increment to the ray

\[ i \rightarrow \text{Find leaf containing intersection point "}i\text{"}. \text{If empty (node highlighted in red), proceed to the next node along the ray direction (node highlighted in blue), else, do ray-splat intersections (node highlighted in green).} \]
Ray-trace Rendering: Octree Traversal

- **Increment by ▲**

- **Ray NOT hitting the Octree**
- **Ray hitting an Empty node**
- **Ray hitting a filled leaf**

- **No ray-splat intersections**

View Plane

- ▲ -- Delta increment to the ray
- \( \vec{i} \) — Find leaf containing intersection point “\( i \)”. If empty (node highlighted in red), proceed to the next node along the ray direction (node highlighted in blue), else, do ray-splat intersections (node highlighted in green)
Ray-trace Rendering: Octree Traversal

- Is it diffuse or specular?
- Search kd-Tree for caustic photons

---

Δ — Delta increment to the ray

\( i \) — Find leaf containing intersection point “\( i \)”. If empty (node highlighted in red), proceed to the next node along the ray direction (node highlighted in blue), else, do ray-splat intersections (node highlighted in green).
Ray-trace Rendering: Octree Traversal

- If specular leaf, generate secondary ray
- Reduce Intensity
- No Shadow Rays
We now how we traverse the ray through the leaves of the octree

How do we get to the leaf containing the intersection point?
Ray-trace Rendering: Octree Traversal

- We now how we traverse the ray through the leaves of the octree.
- *How do we get to the leaf containing the intersection point?*
Ray-trace Rendering: Octree Traversal

- We now how we traverse the ray through the leaves of the octree

- How do we get to the leaf containing the intersection point?
We now how we traverse the ray through the leaves of the octree

How do we get to the leaf containing the intersection point?
- We now how we traverse the ray through the leaves of the octree

- How do we get to the leaf containing the intersection point?
Ray-trace Rendering: Octree Traversal

- We now how we traverse the ray through the leaves of the octree
- *How do we get to the leaf containing the intersection point?*

- No condition checks per level
- Use SFC ordering
Ray-trace Rendering: Octree Traversal

➢ We now how we traverse the ray through the leaves of the octree

➢ How do we get to the leaf containing the intersection point?

➢ ..... On GPU !
Memory Map on CUDA-GPU

- Device
  - Multiprocessor N
  - Multiprocessor 2
  - Multiprocessor 1
    - Shared Memory
    - Registers
      - Processor 1
      - Processor 2
      - Processor M
    - Instruction Unit
    - Constant Cache
    - Texture Cache
  - Device Memory
  - Texture memory
Memory Map on CUDA-GPU

- Store Octree as a texture on GPU
- Improve texture cache-hits
Memory Map on CUDA-GPU

- Store Octree as a texture on GPU
- Improve texture cache-hits
- Store Octree as an array on CPU
Ray-trace Rendering: Octree Traversal

- Store Empty nodes as well!
- Octree becomes a $N^3$ tree ($N=2$)
Ray-trace Rendering: Octree Traversal

Octree Node Pool in Texture memory

Node Texel – 32 bits

- 30 bits used for storing address of the first child (if Internal Node) or address to the point data in data pool (if node is a leaf)
- 1 bit to check if the current node is empty
- 1 bit to check if the current node is a leaf

Octree Traversal:

- All 8 children of I₁ grouped together
- All 8 children of I₂ grouped together

Octree Node Pool:

- E – Empty Node
- L – Leaf Node
- I – Internal Node

Node Texel:

- 8 bits for R
- 8 bits for G
- 8 bits for B
- 8 bits for A
Ray-trace Rendering: Octree Traversal

Octree Node Pool in Texture memory

All 8 children of $I_1$ grouped together

All 8 children of $I_2$ grouped together

Pointer to $I_1$’s First Child

Pointer to $I_2$’s First Child

Node Texel – 32 bits

30 bits used for storing address of the first child (if Internal Node) or address to the point data in data pool (if node is a leaf)

1 bit to check if the current node is a leaf

1 bit to check if the current node is empty

Node Texel – 32 bits

8 bits 8 bits 8 bits 8 bits

No parent pointer required

Point data stored as another 1D texture
Ray-trace Rendering: Octree Traversal

Child node (0,0)  
Child node (1,0)  
p = (0.6, 0.4)

Child node (0,1)  
Child node (1,1)

\[ n = c + \text{int}(p \times N) \]
\[ = c + \text{int}((0.6 \times 2, 0.4 \times 2)) \]
\[ = c + \text{int}((1.2, 0.8)) \]
\[ = c + (1, 0) \]

Octree Node Pool in Texture memory

All 4 children of \( I_1 \) grouped together in \( N^2 \) Tree

Point to \( I_1 \)'s First Child

\[ c + (1, 0) = c + (y \times 2 + x \times 1) \]
\[ = c + (1 \times 2 + 0 \times 1) \]
\[ = c + 2 \]
\[ = \text{Third Child} \]

\( E \) – Empty Node  \( L \) – Leaf Node  \( I \) – Internal Node
Ray-trace Rendering: Octree Traversal

\[ n = c + \text{int}(p \times N) \]

- \[ n = c + \text{int}((0.6 \times 2, 0.4 \times 2)) \]
- \[ n = c + \text{int}((1.2, 0.8)) \]
- \[ n = c + (1, 0) \]

Octree Node Pool in Texture memory

All 4 children of \( I_1 \) grouped together in \( N^2 \) Tree

- \( c + (1, 0) = c + (y \times 2 + x \times 1) \)
- \[ = c + (1 \times 2 + 0 \times 1) \]
- \[ = c + 2 \]
- \[ = \text{Third Child} \]

E – Empty Node  L – Leaf Node  I – Internal Node
Ray-trace Rendering: Octree Traversal

- Update $p$

\[ p = p \times N - \text{int}(p \times N) \]
Ray-trace Rendering: Octree Traversal

Octree Node Pool in Texture memory

$p = p \times N + \text{int}(p \times N)$
$= (0.6 \times 2, 0.4 \times 2) + \text{int}((0.6 \times 2, 0.4 \times 2))$
$= (1.2, 0.8) + \text{int}((1.2, 0.8))$
$= (0.2, 0.8)$

Update $p$

$\mathbf{p} = p \times N - \text{int}(p \times N)$
Ray-trace Rendering: **Point Data Texture on GPU**

- Every point has following attributes:
  - Co-ordinates (3 floats)
  - Normal (3 floats)
  - Color (3 floats)
  - Radius (1 float)
  - Material ID (1 float)

- Total of 44 bytes per point
- 3 Texels, each of 16 bytes

![Point Data Texel – 128 bits](image)

**R** 32 bits  **G** 32 bits  **B** 32 bits  **A** 32 bits
Ray-trace Rendering: Point Data Texture on GPU

Octree Node Pool in Texture memory

Data Pool in Texture memory

Begin-End Texture Pool

Node Texel – 128 bits

Each location is 4 bytes
Ray-trace Rendering: **Multiple Leaves Per Splat**

![Diagram](image)

- LEAF $L_1$
- LEAF $L_2$
- Ray $R$
- $r_j$, $c_j$, $S_j$
Ray-trace Rendering: Multiple Leaves Per Splat

- Traverse top-down. Check if axis-aligned bounding box of splat intersects the node.
- If leaf, perform a parameterized intersection test, using splat-aligned square bounding box.
Ray-trace Rendering: **Multiple Leaves Per Splat**

- Adding splats to every intersecting leaf means adding **44 bytes** of data for every *Extra Splat (ES)*
- Store only the address (**4 bytes**) of the *ES*!
Ray-trace Rendering: Multiple Leaves Per Splat

- Adding splats to every intersecting leaf means adding **44 bytes** of data for every _Extra Splat (ES)_.

- Store only the address (**4 bytes**) of the _ES_!

---

**Octree Node Pool in Texture memory**

```
... LEAF L₁ ... LEAF L₂ ... 
```

**Begin-End Texture Pool**

```
... beginP for Leaf L₁ ... endP for Leaf L₁ ... 
... beginP for Leaf L₂ ... endP for Leaf L₂ ... 
```

**Data Pool in Texture memory**

```
... beginP for Leaf L₁ ... endP for Leaf L₁ ... 
... beginP for Leaf L₂ ... endP for Leaf L₂ ... 
```

Each point data is **44 bytes**.
Ray-trace Rendering: Multiple Leaves Per Splat

- Make some space for *Extra Splats (ES)*

Octree Node Pool in Texture memory

```
... LEAF L₁ ... LEAF L₂ ...
```

Begin-End Texture Pool

```
... beginP for Leaf L₁ E E₁ E₂ E₃ E₄ endP for Leaf L₁/ beginP for Leaf L₂ ... endP for Leaf L₂
```

Data Pool in Texture memory

```
E₃ ... E₁ ... 
```

- Each location is 4 bytes
- Each point data is 44 bytes
Ray-trace Rendering: Multiple Leaves Per Splat

- Make some space for *Extra Splats (ES)*

Octree Node Pool in Texture memory

- Data Pool in Texture memory

Pre-process step

Each point data is 44 bytes

Each location is 4 bytes
Global Illumination of Point Models

Diffuse Effects
- V-Map
- FMM

Specular Effects
- Caustics
- Reflections
- Refractions

Texture Cache memory
- Fast Octree Traversal for Ray-Tracing
- Ray-Splat Intersections
- Normal Field on Splats
- Coherent Rays Super-Sampling

Photon Generation
- Kd-Trees
Global Illumination of Point Models

Diffuse Effects
- V-Map
- FMM

Specular Effects
- Caustics
- Reflections
- Refractions
- Photon Generation
  - Kd-Trees
- Fast Octree Traversal for Ray-Tracing
  - Ray-Splat Intersections
  - Normal Field on Splats
- Coherent Rays
- Super-Sampling

GPU
- Texture Cache memory
Ray-tracing: Coherent Rays & Super-Sampling

- Important to get *texture cache-hits*!
- Use Ray-Coherency and Super-Sampling *(8x4)*
- Gives well-behaved warp of 32 threads
- Also reduces *aliasing*
Fusing Together: Diffuse and Specular Effects

- FMM takes care of $LD^+$ path
- Ignores contributions from specular splats ($LS^+D$)
- Does not transfer energy to specular splats ($LD^*S^+$)
- $LS^+D$ and $LDS^+$ handled by caustics and ray-tracing
Fusing Together: Diffuse and Specular Effects

- FMM takes care of $LD^+$ path
- Ignores contributions from specular splats ($LS^+D$)
- Does not transfer energy to specular splats ($LD^*S^+$)
- $LS^+D$ and $LDS^+$ handled by caustics and ray-tracing

- This gives us the whole set-up of a complete global illumination package for point models!
Global Illumination of Point Models

Diffuse Effects
- V-Map
- FMM

Specular Effects
- Caustics
- Reflections
- Refractions
  - Photon Generation
  - Kd-Trees

Fast Octree Traversal for Ray-Tracing
Ray-Splat Intersections
Normal Field on Splats

Coherent Rays
Super-Sampling

Texture Cache memory

GPU
Plan

➤ Introduction – *Problem Definition*

➤ Diffuse Effects – *Overview*

➤ Specular Effects – *Details*

➤ *Results*

➤ *Wrap-Up*
Results

Work In Progress!
Plan

- Introduction – *Problem Definition*
- Diffuse Effects – *Overview*
- Specular Effects – *Details*
- Results
- Wrap-Up
Final Notes

- Lack of connectivity makes GI on point models, difficult

- Point-to-Point visibility arguably one of the more difficult problem in rendering

- V-Map helps in resolving visibility between clusters at group level

- FMM uses V-Map for diffuse irradiance transfers

- Parallel versions of V-Map construction and FMM implemented using CUDA on GPU

- Achieves upto 20x speed-ups

- Further, V-Maps were used for tracing initial paths of caustic photons, saving time!

- Factors like Number of photons to be sent, where to send were in sync with emissive power of light source and areas occupied by the splats as seen from the light source

- Normal field generated for each specular splat gave accurate secondary ray direction
Final Notes

- We used an **efficient yet naive octree traversal** algorithm (on GPU) to do photon tracing.
- **Texture cache** on GPU used for efficient octree storage and fast traversal.
- Same octree traversal algorithm used for ray-trace rendering as well.
- **Kd-tree** was used to organize the caustic photon map, enabling fast retrieval.
- Point data organized into texture accessible by leaves of the octree.
- Provision for **multiple leaves per splat** was added to avoid undesirable holes & artifacts.
- **Coherent rays and super-sampling** was performed to aid a well behaved warp on GPU and avoid aliasing artifacts.
- Diffuse and Specular effects are thus achieved in an **unified global illumination package** for point models.
Thank You

Fractal: Mandel Zoom - Satellite Antenna, Mandelbrot set
Photon Mapping

- A Two-pass GI method
Photon Mapping

- A Two-pass GI method
  - First pass builds the photon map (Follow Heckbert’s notation)
    - Emit photons from light sources into the scene
    - Store them in a photon map on hitting diffuse objects
Photon Mapping

- A Two-pass GI method
  - Second pass, the ray-trace rendering pass
    - Make $kNN$ queries on the photon map
    - Extract information about the radiance values
Paths of Photons

- Diffuse Interactions
- LD* path
- Solution: FMM
Paths of Photons

- Specular Interactions
- $LS^+D$ path
- Result: **Caustics**
Paths of Photons: Our Approach

- DO NOT CONSIDER RED PATHS ($LS^+D^+$)
- Considered during FMM
- ONLY $LS^+D$ for Caustics

![Diagram showing various photon paths](image)
View-Independence v/s View-Dependence

- LDDD
- LDD
- LSD
- LSSDD
- LDDD
- LD
- LS
- LS
- LSS
- LSD
- LSSD

- Eye/Camera View Plane
- Specular Reflective Splat
- Specular Refractive Splat
- Diffuse Splat

- EDD/ED⁺
- ESD/ES⁺D
- EDS/ED⁺S⁺D
- ESS