Practical Implementation of Light Scattering Effects Using Epipolar Sampling and 1D Min/Max Binary Trees

Egor Yusov
Graphics Software Engineer – Intel Corporation
Demo
Agenda

• Introduction
• Light Scattering Theory
• Solution to Inscattering Integral
• Epipolar Sampling
• 1D Min/Max Binary Trees
• Implementation
• Performance
Introduction

• Light scattering greatly enhances the realism
• It is hard to compute
• Achieving high-performance is challenging

\[ \int_{S_c}^{S_0} e^{-T(P\rightarrow C)} \beta(P)p(\theta) \frac{Ie^{-T(L\rightarrow P)}}{||L - P||^2} V(P)ds \]

\[ T(A \rightarrow B) = \int_A^B \beta(P)ds \quad p_R(\theta) = \frac{3}{16\pi} (1 + \cos^2 \theta) \]

\[ p_M(\theta) = \frac{1}{4\pi} \frac{3(1 - g^2)}{2 + g^2} \frac{1 + \cos^2 \theta}{(1 + g^2 - 2g \cos \theta)^{3/2}} \]
Previous Work

- Hoffman & Preetham, 2002 – Alanalytical model for outdoor light scattering
- Sun et al, 2005 – Semi-analytical solution to airlight integral
- Tevs et al, 2008 – Using maximum mipmaps for dynamic height field rendering
- Chen et al, 2011 – Using 1d min-max mipmaps for volumetric shadows
- Engelhardt & Dachsbacher, 2010 – Epipolar Sampling
Inscattering Integral Derivation

Aerial Perspective

\[ L_{\text{Cam}} = L_{\text{Obj}} \cdot e^{-T(0\rightarrow C)} + L_{\text{In}} \]

For homogeneous medium

\[ T(A \rightarrow B) = \int_{A}^{B} \beta(P)ds = \beta \|A - B\| \]
Inscattering Integral Derivation

Radiant Intensity of a Point Light

- $I$ – radiant intensity
- $\hat{v} = \frac{0-C}{\|0-C\|}$ – view direction

$$L_{In} = I$$
Inscattering Integral Derivation

Attenuation and optical thickness

- Amount of energy reaching P is
  - Inversely proportional to distance squared
  - Attenuated by the extinction in the media

\[ L_{In} = \frac{I \, e^{-T(L\rightarrow P)}}{||L - P||^2} \]
Inscattering Integral Derivation

Visibility factor

- $V(P) = 1$ if point is lit and 0 if shadowed
  - Filtered in practice

\[
L_{In} = V(P) \frac{I e^{-T(L\rightarrow P)}}{\|L - P\|^2}
\]
Inscattering Integral Derivation

Scattering coefficient

- Total scattered light (in any direction) is proportional to
  - Section length $ds$
  - Scattering coefficient $\beta(P)$

$$L_{In} = \beta(P) V(P) \frac{I e^{-T(L\rightarrow P)}}{\|L - P\|^2} ds$$
Phase function

- Phase function gives amount of light scattered towards the camera

\[ L_{In} = \int_{V_{P}} p(\theta) \beta(P) V(P) \frac{I e^{-T(L\rightarrow P)}}{\|L - P\|^2} \, ds \]
Inscattering Integral Derivation

Attenuation of inscattered light

- Differential inscattered light is attenuated in the media
- Total inscattering is given by integrating differential amounts

\[
L_{In} = \int_{C}^{0} e^{-T(P\rightarrow C)} p(\theta) \beta(P) V(P) \frac{l e^{-T(L\rightarrow P)}}{||L - P||^2} \, ds
\]
Inscattering Integral Derivation

Directional light

- Phase function does not vary across the ray
- Light intensity is constant

\[ L_{ln} = \rho(\theta) \int_C e^{-T_{(P\rightarrow C)}} \beta(P)V(P)E_{Sun} \, ds \]
Scattering Properties of the Media

Rayleigh scattering

- Caused by molecules
- Wavelength-dependent
  \[ \beta_R.\text{rgb} = (5.8, 13.5, 33.1) \times 10^{-6} \]
- Almost isotropic:
  \[ p_R(\theta) = \frac{3}{16\pi} (1 + \cos^2 \theta) \]
Scattering Properties of the Media

Mie Scattering

- Caused by aerosols
- Wavelength-independent
  \[ \beta_M \cdot \text{rgb} = 2.0 \times 10^{-5} \]
- Anisotropic

\[
p_M(\theta) = \frac{1}{4\pi} \frac{3(1 - g^2)}{2(2 + g^2)} \frac{1 + \cos^2\theta}{(1 + g^2 - 2g\cos\theta)^{3/2}}
\]
Scattering Properties of the Media

Scattering due to both types of particles

- \( T(\mathbf{A} \rightarrow \mathbf{B}) = \beta_\Sigma \| \mathbf{A} - \mathbf{B} \| \)
  - \( \beta_\Sigma = \beta_R + \beta_M \)
- \( \beta p(\theta) \rightarrow \beta_R \cdot p_R(\theta) + \beta_M \cdot p_M (\theta) \equiv P_\Sigma (\theta) \)
Scattering Properties of the Media

Isotropic phase function

\[ p_M(\theta) = \frac{1}{4\pi} \]
Scattering Properties of the Media

Anisotropic phase function

\[
p_M(\theta) = \frac{1}{4\pi} \frac{3(1 - g^2)}{2(2 + g^2)} \times \frac{1 + \cos^2\theta}{(1 + g^2 - 2g\cos\theta)^{3/2}}.
\]
Solution to Inscattering Integral

Directional light – fully analytical solution

\[ L_{In}^{Dir}(\theta, S) = \frac{E_{Sun} \cdot P_{\Sigma}(\theta)}{\beta_{\Sigma}} \left( 1 - e^{-\beta_{\Sigma} \cdot S} \right) \]

\[ S = ||O - C|| \]
Solution to Inscattering Integral

Point light source – integral reformulation

\[ L_{In}^{Pt} = L_{In}^{Pt}(h, S_C, S_O) = \]

\[ I \int_{S_C}^{S_O} P_{\Sigma}(\theta) \cdot e^{-\beta_{\Sigma}} ds \]

\[ \cos(\theta) = -\frac{s}{\sqrt{h^2 + s^2}} \]

\[ ||P - C|| + ||L - P|| \]

\[ ||L - P||^2 \]
Solution to Inscattering Integral

Point light source – look-up table

Precompute 2D LUT:

$$\mathcal{L}[h, S] = L_{In}^{Pt}(h, S, +\infty)$$
Solution to Inscattering Integral

Point light source – semi-analytical solution

\[ L_{Ln}^{Pt}(h, S_C, S_O) = \]
\[ \mathcal{L}[h, S_C] \]
\[ -e^{-L[h, S_O]} \]

\[ T(C \rightarrow O) \]

\[ L_0 \]

\[ h \]

\[ \mathcal{T}_C \rightarrow \mathcal{O} \]

\[ S_C \]

\[ S_O \]
Volumetric Shadows

\[ L_{In}^{Pt} = \sum_{i=0}^{n-1} \left[ e^{-}L[h, S_{i}^{Ent}] - e^{-}L[h, S_{i}^{Ext}] \right] \]

\[ T(S_{i}^{Ext} \rightarrow C) \]

\[ T(S_{i}^{Ent} \rightarrow C) \]
Volumetric Shadows

\[ L_{Ln}^{Dir} = F(\theta) \cdot \sum_{i=0}^{n-1} \left( e^{-\beta \Sigma \cdot S_{i}^{Ent}} - e^{-\beta \Sigma \cdot S_{i}^{Ext}} \right) \]

\[ F(\theta) = \frac{E_{Sun} \cdot P_{\Sigma}(\theta)}{\beta_{\Sigma}} \]
Volumetric Shadows

Starting point - tracing the view ray in shadow map space

For each pixel:
- Project the view ray onto the shadow map
- Set up $\text{Prev}L_{\text{In}.rgb} = \mathcal{L}[h, S_C]$, $L_{\text{In}.rgb} = 0$
- Step through each texel:
  - $\text{Curr}L_{\text{In}.rgb} = e^{-\beta_S \cdot (S-S_C)} \mathcal{L}[h, S]$
  - $L_{\text{In}.rgb} += (\text{Prev}L_{\text{In}.rgb} - \text{Curr}L_{\text{In}.rgb}) \cdot V(P)$
  - $\text{Prev}L_{\text{In}.rgb} = \text{Curr}L_{\text{In}.rgb}$
Volumetric Shadows

Perspective-correct interpolation

\[
\frac{1}{z} = \frac{1}{z_C} + \alpha \left( \frac{1}{z_O} - \frac{1}{z_C} \right)
\]

\[
\frac{S}{z} = \frac{S_C}{z_C} + \alpha \left( \frac{S_O}{z_O} - \frac{S_C}{z_C} \right)
\]
Volumetric Shadows

Starting point - tracing the view ray in shadow map space

- Need to test all samples along the ray
- Too slow (> 90 ms on GeForce 680 GTX)
- Optimizations are necessary!
Epipolar Sampling
Epipolar Sampling
Epipolar Sampling
Epipolar Sampling

Sample generation

Entry point

Exit point

Entry point

Exit point
Epipolar Sampling

Sample generation – improved sample placement

Too dense sampling

Much better sampling
Epipolar Sampling

Coordinate texture

Sample coordinate on each line
Epipolar Sampling

Sample classification

Initial ray marching samples are placed equidistantly.

Additional ray marching samples are placed at depth breaks.

The remaining samples are interpolation samples.
Epipolar Sampling

Interpolation source texture

| 0,0 | 0,3 | 0,3 | 3,3 | 4,4 | 4,8 | 4,8 | 4,8 | 8,8 |

0  1  2  3  4  5  6  7  8
Epipolar Sampling

Transformation from epipolar to rectangular coordinates

- Cast epipolar line through the pixel
- Perform bilateral filtering
  - One Gather() and two Sample() instructions are used
Epipolar Sampling

Fix-up pass

- Not all samples can be properly interpolated from epipolar coordinates
- These samples are marked in stencil
- Additional fix-up pass is performed without 1D min/max acceleration
Epipolar Sampling

91 ms -> 14 ms

We can do better!
1D min/max binary trees

1D height map & first level
1D min/max binary trees

Second level
1D min/max binary trees

Third level

Depth
1D min/max binary trees

Traversal optimization

- $$\max(d_A, d_B) < d_{\text{min}}$$: lit
- $$\min(d_C, d_D) > d_{\text{max}}$$: shadowed
- EF: neither lit nor shadowed
Slice Origin and Direction

- Location of the $i$-th sample is $O_{UV} + i \cdot \vec{D}_{UV}$
Slice Origin and Direction

\[ \hat{S}_0 = \frac{L - C}{\|L - C\|} \]
\[ \hat{S}_1 = \frac{S_{\text{Exit}} - C}{\|S_{\text{Exit}} - C\|} \]
\[ \vec{N}_{\text{Slice}} = \hat{S}_1 \times \hat{S}_0 \]
\[ \vec{D} = \vec{N}_{\text{Slice}} \times \vec{N}_{\text{Light}} \]
Slice Origin and Direction
Slice Origin and Direction

\[ \mathbf{P}_0 = c_1 \mathbf{N}_{\text{Slice}} + c_2 \mathbf{N}_{\text{Light}} + t \mathbf{D} \]

\[ \mathbf{O} = \mathbf{P}_0 + t_{\text{min}} \cdot \mathbf{D} \]
Slice Origin and Direction

\[ \vec{S}_1 \quad \vec{N}_{\text{Slice}} \]

\[ \vec{D} \quad \vec{L} \]

[Diagram showing the relationship between \( \vec{S}_1 \), \( \vec{N}_{\text{Slice}} \), \( \vec{D} \), and \( \vec{L} \).]
Colored Light Shafts
## Implementation Details

### Auxiliary textures

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Back Buffer</td>
<td>RGBA_UNORM8</td>
<td>$W_{Screen} \times H_{Screen}$</td>
</tr>
<tr>
<td>Camera Space Z</td>
<td>R_FLOAT32</td>
<td>$W_{Screen} \times H_{Screen}$</td>
</tr>
<tr>
<td>Camera Depth Stencil</td>
<td>D24_S8</td>
<td>$W_{Screen} \times H_{Screen}$</td>
</tr>
<tr>
<td>Coordinate texture</td>
<td>RG_FLOAT32</td>
<td>$N_{Samples} \times N_{Slices}$</td>
</tr>
<tr>
<td>Epipolar depth-stencil</td>
<td>D24_S8</td>
<td>$N_{Samples} \times N_{Slices}$</td>
</tr>
<tr>
<td>Epipolar camera space Z</td>
<td>R_FLOAT32</td>
<td>$N_{Samples} \times N_{Slices}$</td>
</tr>
<tr>
<td>Interpolation source</td>
<td>RG_UINT16</td>
<td>$N_{Samples} \times N_{Slices}$</td>
</tr>
<tr>
<td>Inscattering $\times 2$</td>
<td>RGBA_FLOAT16</td>
<td>$N_{Samples} \times N_{Slices}$</td>
</tr>
<tr>
<td>Slice End Points</td>
<td>RGBA_FLOAT32</td>
<td>$N_{Slices} \times 1$</td>
</tr>
<tr>
<td>Slice UV origin &amp; direction</td>
<td>RGBA_FLOAT32</td>
<td>$N_{Slices} \times 1$</td>
</tr>
<tr>
<td>1D min/max shadow map $\times 2$</td>
<td>RG_FLOAT16</td>
<td>$D_{ShadowMap} \times N_{Slices}$</td>
</tr>
</tbody>
</table>
Implementation

1. Reconstruct camera space Z

2. Render Slice End Points in Screen Space

3. Render coordinate texture

4. Refine Sample Locations

5. Render Slice Origin and Direction in Shadow Map Space

Camera depth buffer

Slice UV Orig & Dir

Interpolation source texture

Epipolar coordinates

Epipolar camera space Z

Epipolar depth stencil
Implementation

6. Build 1D min/max mipmap

7. Mark Ray Marching Samples

8. Do Ray Marching

Initial inscattering

Camera space Z

Epipolar coordinates

Interpolation source texture

Slice UV Orig & Dir

1D min-max mipmap

Shadow map

Epipolar depth stencil
Implementation

Initial inscattering

9. Interpolate inscattering

Interpolated inscattering

9. Interpolate inscattering

Interpolation source texture

Epipolar camera space Z

10. Transform to rectangular coordinates

Slice End Points

Camera space Z

Source Color Buffer
Implementation

Camera space Z

Source Color Buffer

Shadow map

12. Fix inscattering

Final image
Implementation Details

Discontinuities search with CS

- Each ray section is processed by one thread group
- Each thread processes one sample
Implementation Details

Discontinuities search with CS

- **Step 1**: each thread sets 1-bit flag in group shared array indicating if there is a depth break
  - `InterlockedOr()` must be used

- **Step 2**: find closest depth break or section end for each sample using bit manipulation and `firstbitlow()` and `firstbithigh()` intrinsics

Up to 6x speed-up
Implementation Details

Min/max binary tree construction
Implementation Details

Min/max binary tree construction

- Step 1: construct 1\textsuperscript{st} tree level using shadow map
  - Use \texttt{Gather()} instructions to read min/max z for each pair of two adjacent samples
  - Compute conservative min/max z

- Step 2: Construct coarser tree levels
  - Use temporary texture to render to
Implementation Details

Optimized sample placement

Gives more than 5% speedup
## Performance

**2560 × 1600 resolution, NVIDIA GeForce 680 GTX, spot light**

<table>
<thead>
<tr>
<th>Profile</th>
<th># slices</th>
<th># samples</th>
<th>SM</th>
<th>Time, ms</th>
<th>Mem, MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brute Force</td>
<td></td>
<td></td>
<td>4096²</td>
<td>91.18</td>
<td></td>
</tr>
<tr>
<td>High Quality</td>
<td>2048</td>
<td>1024</td>
<td>4096²</td>
<td>5.986</td>
<td>88.0</td>
</tr>
<tr>
<td>Balanced</td>
<td>1024</td>
<td>1024</td>
<td>2048²</td>
<td>3.269</td>
<td>44.0</td>
</tr>
<tr>
<td>High performance</td>
<td>1024</td>
<td>512</td>
<td>1024²</td>
<td>2.267</td>
<td>18.0</td>
</tr>
</tbody>
</table>

- **ms**

- **Reconstruct cam Z**
- **Render coord texture**
- **Detect depth breaks**
- **Build 1D min/max trees**
- **Ray march**
- **Interpolate**
- **Transform to rect coords**
- **Inscattering fix-up**
Brute Force Ray Marching
High Quality
Balanced
High Performance
# Performance

**1366 x 768 resolution, Nvidia Quadro 1000M, spot light**

<table>
<thead>
<tr>
<th>Profile</th>
<th># slices</th>
<th># samples</th>
<th>SM</th>
<th>Time, ms</th>
<th>Mem, MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brute Force</td>
<td></td>
<td></td>
<td>2048²</td>
<td><strong>164</strong></td>
<td></td>
</tr>
<tr>
<td>High Quality</td>
<td>1024</td>
<td>1024</td>
<td>2048²</td>
<td><strong>18.9</strong></td>
<td>44.0</td>
</tr>
<tr>
<td>Balanced</td>
<td>512</td>
<td>512</td>
<td>1024²</td>
<td><strong>8.37</strong></td>
<td>11.0</td>
</tr>
<tr>
<td>High performance</td>
<td>512</td>
<td>256</td>
<td>512²</td>
<td><strong>5.38</strong></td>
<td>4.5</td>
</tr>
</tbody>
</table>

![Graph showing performance times for different processes](image)

- Reconstruct cam Z
- Render coord texture
- Detect depth breaks
- Build 1D min/max trees
- Ray march
- Interpolate
- Transform to rect coords
- In-scattering fix-up
Brute Force Ray Marching
High Quality
Balanced
High Performance
## Performance

### 1024 × 768, Intel CPU with 3rd Gen HD Graphics, spot light

<table>
<thead>
<tr>
<th>Profile</th>
<th># slices</th>
<th># samples</th>
<th>SM</th>
<th>Time, ms</th>
<th>Mem, MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brute Force</td>
<td>2048²</td>
<td></td>
<td>2048²</td>
<td>184.79</td>
<td></td>
</tr>
<tr>
<td>High Quality</td>
<td>1024</td>
<td>1024</td>
<td>2048²</td>
<td>28.83</td>
<td>44.0</td>
</tr>
<tr>
<td>Balanced</td>
<td>512</td>
<td>512</td>
<td>1024²</td>
<td>12.85</td>
<td>11.0</td>
</tr>
<tr>
<td>High performance</td>
<td>512</td>
<td>256</td>
<td>512²</td>
<td>8.85</td>
<td>4.5</td>
</tr>
</tbody>
</table>

### Timeline

```
ms

Reconstruct cam Z | Render coord texture | Detect depth breaks | Build 1D min/max trees | Ray march | Interpolate | Transform to rect coords | Inscattering fix-up
```

69
Brute Force Ray Marching
High Quality
Balanced
High Performance
Performance

- Spot light, balanced quality
- 1280 × 960 resolution, NVIDIA GeForce 480 GTX
- 1024 slices, 512 samples in slice, 2048 × 2048 Shadow Map

Time, ms

- Reconstruct cam Z: 0.07 ms
- Render slice enter/exit points: 0.01 ms
- Render coord texture: 0.09 ms
- Detect depth breaks: 0.31 ms
- Render slice origin and dir: 0.01 ms
- Build 1D min/max trees: 0.38 ms
- Ray march: 1.67 ms
- Interpolate: 0.06 ms
- Transform to rect coords: 0.52 ms
- Inscattering fix-up: 0.26 ms

Overall: 3.366 ms
Performance

- Directional light, high quality
- 1280 × 960 resolution, NVIDIA GeForce 480 GTX
- 1024 slices, 512 samples in slice, 4096 × 4096 Shadow Map

Time, ms

<table>
<thead>
<tr>
<th>Task</th>
<th>Time, ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reconstruct cam Z</td>
<td>0.07</td>
</tr>
<tr>
<td>Render slice enter/exit</td>
<td>0.01</td>
</tr>
<tr>
<td>Render coord texture</td>
<td>0.10</td>
</tr>
<tr>
<td>Detect depth breaks</td>
<td>0.31</td>
</tr>
<tr>
<td>Render slice origin and dir</td>
<td>0.01</td>
</tr>
<tr>
<td>Build 1D min/max trees</td>
<td>0.65</td>
</tr>
<tr>
<td>Ray march</td>
<td>2.12</td>
</tr>
<tr>
<td>Interpolate</td>
<td>0.06</td>
</tr>
<tr>
<td>Transform rect coords</td>
<td>0.57</td>
</tr>
<tr>
<td>Inscattering fix-up</td>
<td>0.20</td>
</tr>
</tbody>
</table>

4.076 ms
Performance

<table>
<thead>
<tr>
<th>Time, ms</th>
<th>No binary trees</th>
<th>With 1D min/max trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ray marching, 5.49</td>
<td>&gt; 2x faster</td>
<td>Ray marching, 1.67</td>
</tr>
<tr>
<td>Binary tree construction, 0.38 ms</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Performance

Colored light shafts

- Binary tree optimization is less efficient
  - Ray marching: 1.67 ms → 5.49 ms
  - Total time: 3.36 ms → 6.86 ms
- Still less than 7 ms on NVIDIA GeForce 480 GTX
Advantages Over Previous Approaches

• Semi-analytical solution is used to calculate the integral on each ray segment
  – No sophisticated mathematics like SVD etc.
• Epipolar scattering is combined with 1D min/max binary trees
  – Significant performance improvements
• Efficient implementation methods
Integration Into Existing Apps

- The technique is fully post-processing
- The only inputs are:
  - Depth buffer
  - Back buffer
  - Shadow map
  - [optional] Stained glass color
- The source code is available at
  http://software.intel.com/gamecode
Extensions

- Large outdoor environments
- Integration with cascaded SM, VSM/EVSM
- Heterogeneous (dynamic) media
  - Clouds
  - Smoke
  - Sparse octree/PRT
References


Acknowledgments

The presenter would like to thank:

- Artem Brizitsky
- Jeffery A. Williams
- Doug Mcnabb
- Dave Bookout
- Charu Chandrasekaran
- Quentin Froemke
- Glen Lewis
- Kyle Weicht
Thank you!