Specialized Acceleration Structures for Ray-Tracing

Warren Hunt

Bill Mark
Forward: Flavor of Research

• **Build is cheap** (especially with scan, lazy and build from hierarchy)
  – Grid build and BVH refit are really cheap

• **Many** specialized acceleration structures are required for maximal rendering performance
  – Already true for game rendering algorithms
One System, Many (Layered) Structures

- Static structures for static objects
- Structures with static topology for dynamic objects with static topology
  - Sub-division surfaces
- Structures for collections of dynamic objects or particles
  - Also used in collision detection
- Structures for clusters of highly coherent rays
  - Primary, shadows, large flat reflectors
- Structures for GI (probably not ray-tracing)
Adaptive Acceleration Structures in Perspective Space

Warren Hunt
Bill Mark
Introduction

• Use the **perspective transform** to accelerate (near) common origin ray-tracing
  – Higher performance for **DoF and Soft Shadows**

• Use an adaptive acceleration structure
  – Avoids “Teapot in a stadium” problems

• Use a **cost metric** to improve the effectiveness of the acceleration structure
  – Derived from first principles
Motivation

- Perspective space holds promise for improving visibility performance for near-common origin rays

- Uniform grid isn’t adaptive enough to trace off-axis rays efficiently
Background: Adaptive Acceleration Structures

- Adaptive acceleration structure builders use cost metrics to improve quality
  - Present many candidate acceleration structures and pick the best (according to the metric)

- Commonly used in a top-down greedy way
  - E.g. The Surface Area Heuristic
kD-Tree Build With a Metric
The Surface Area Metric

- Most commonly used cost metric for building acceleration structures

\[
\text{Cost}(s) = P(s_{\text{left}})\text{Cost}(s_{\text{left}}) + P(s_{\text{right}})\text{Cost}(s_{\text{right}})
\]

- \(P(x)\) is the probability of a ray hitting \(x\)
- Assumes uniform ray distribution
- Assumes rays don’t interact with objects
The SAM in Perspective Space

- Uniform distribution assumption for the SAM makes even less sense in perspective space
- Response: Start over from scratch
  - Derive a new cost metric in perspective space
  - We call the new metric the PSAM
The PSAM: New Assumptions

• Assume rays are originating from a rectangular source
  – Area camera or light
  – Dimensions $2A_x$ by $2A_y$
Ray Distribution Assumptions

SAM:

PSAM:
The PSAM: New Assumptions

• Assume rays are originating from a rectangular source
  – Area camera or light
  – Dimensions $2A_x$ by $2A_y$

• Use one perspective space structure per area camera or light
Many Structures:

- One for each Camera or Light
- Each structure makes unique assumptions about ray distribution
Deriving the PSAM: Setup

- **Goal:** Compute the probability of a ray intersecting an axis aligned voxel
  - Given a voxel and ray distribution
- **Measure the number of rays entering a voxel**
  - By integrating the intersection function over all rays
- **The child/parent ratio of these measures determines the probability of ray intersection**
Measure of Rays:

- Given area light with dimensions $2A_x$ by $2A_y$
- Given box with dimensions $\Delta x'$ by $\Delta y'$ by $\Delta z'$

The measure $(G)$ of rays intersecting the box:

$$G(\text{box}, A) = \Delta x'\Delta y' + (A_y/2)(\Delta x'\Delta z') + (A_z/2)(\Delta y'\Delta z')$$

- Derivation in the paper
Measure of Rays:

- Given area light with dimensions $2A_x$ by $2A_y$
- Given box with dimensions $\Delta x'$ by $\Delta y'$ by $\Delta z'$

- The measure ($G$) of rays intersecting the box:

$$G(\text{box},A) = \Delta x'\Delta y' + (A_y/2)(\Delta x'\Delta z') + (A_z/2)(\Delta y'\Delta z')$$

- Derivation in the paper
The PSAM:

- Cost(s,A) = P(s_{left})Cost(s_{left}) + P(s_{right})Cost(s_{right})

- Given G(box, A):

- Cost(s,A) = (G(s_{left},A)/G(s,A))Cost(s_{left})
  + (G(s_{right},A)/G(s,A))Cost(s_{right})

  - Still assumes that rays don’t interact with objects
The PSAM: Use

- Use the traditional SAH build algorithm with the PSAM in place of the SAM

- Clip the world to avoid the perspective singularity (cube map for point lights)

- Use back-face culling in all cases
Results

• Perspective space and the PSAH improves performance by 20%-40%
  – For soft shadows and DoF

• In many cases build and render in perspective space costs less than just render in world space
  – Back-face culling reduces build cost by ~50%
## Results Subset Table (1 core)

<table>
<thead>
<tr>
<th>Scene</th>
<th>Type</th>
<th>ISect</th>
<th>Traverse</th>
<th>Render</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>FForest</td>
<td>Eye/SAH</td>
<td>9.369 M</td>
<td>23.15 M</td>
<td>1.16 s</td>
<td>8.37 s</td>
</tr>
</tbody>
</table>
## Results Subset Table (1 core)

<table>
<thead>
<tr>
<th>Scene</th>
<th>Type</th>
<th>lSect</th>
<th>Traverse</th>
<th>Render</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFForest</td>
<td>Eye/SAH</td>
<td>9.369 M</td>
<td>23.15 M</td>
<td>1.16 s</td>
<td>8.37 s</td>
</tr>
<tr>
<td>FFForest</td>
<td>Eye/PSAH</td>
<td>5.482 M</td>
<td>14.49 M</td>
<td>0.71 s</td>
<td>1.29 s</td>
</tr>
</tbody>
</table>
## Results Subset Table (1 core)

<table>
<thead>
<tr>
<th>Scene</th>
<th>Type</th>
<th>lSect</th>
<th>Traverse</th>
<th>Render</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>FForest</td>
<td>Eye/SAH</td>
<td>9.369 M</td>
<td>23.15 M</td>
<td>1.16 s</td>
<td>8.37 s</td>
</tr>
<tr>
<td>FForest</td>
<td>Eye/PSAH</td>
<td>5.482 M</td>
<td>14.49 M</td>
<td>0.71 s</td>
<td>1.29 s</td>
</tr>
<tr>
<td>Bunny69</td>
<td>Eye/SAH</td>
<td>2.838 M</td>
<td>10.43 M</td>
<td>0.492 s</td>
<td>2.62 s</td>
</tr>
<tr>
<td>Bunny69</td>
<td>Eye/PSAH</td>
<td>1.515 M</td>
<td>5.981 M</td>
<td>0.272 s</td>
<td>0.43 s</td>
</tr>
</tbody>
</table>
## Results Subset Table (1 core)

<table>
<thead>
<tr>
<th>Scene</th>
<th>Type</th>
<th>ISect</th>
<th>Traverse</th>
<th>Render</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>FForest</td>
<td>Eye/SAH</td>
<td>9.369 M</td>
<td>23.15 M</td>
<td>1.16 s</td>
<td>8.37 s</td>
</tr>
<tr>
<td>FForest</td>
<td>Eye/PSAH</td>
<td>5.482 M</td>
<td>14.49 M</td>
<td>0.71 s</td>
<td>1.29 s</td>
</tr>
<tr>
<td>Bunny69</td>
<td>Eye/SAH</td>
<td>2.838 M</td>
<td>10.43 M</td>
<td>0.492 s</td>
<td>2.62 s</td>
</tr>
<tr>
<td>Bunny69</td>
<td>Eye/PSAH</td>
<td>1.515 M</td>
<td>5.981 M</td>
<td>0.272 s</td>
<td>0.43 s</td>
</tr>
<tr>
<td>FForest</td>
<td>DoF/SAH</td>
<td>148.0 M</td>
<td>369.6 M</td>
<td>18.5 s</td>
<td>25.8 s</td>
</tr>
<tr>
<td>FForest</td>
<td>DoF/PSAH</td>
<td>92.60 M</td>
<td>314.7 M</td>
<td>14.5 s</td>
<td>15.1 s</td>
</tr>
<tr>
<td>Bunny69</td>
<td>DoF/SAH</td>
<td>81.19 M</td>
<td>255.1 M</td>
<td>12.7 s</td>
<td>14.8 s</td>
</tr>
<tr>
<td>Bunny69</td>
<td>DoF/PSAH</td>
<td>46.29 M</td>
<td>141.1 M</td>
<td>7.8 s</td>
<td>8.0 s</td>
</tr>
</tbody>
</table>
Notes:

- Build Only Gets Cheaper!
  - Build times are not using lazy build
  - Build times are not using build from hierarchy

- Hard/Soft shadows behave much like primary/DoF rays (elided for time/space)
# Heuristic Compare on FForest

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Build</th>
<th>Isects</th>
<th>Traverse</th>
<th>Render</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eye/Median</td>
<td>0.209 s</td>
<td>63.85 M</td>
<td>48.76 M</td>
<td>4.16 s</td>
<td>4.38 s</td>
</tr>
<tr>
<td>Eye/SAH</td>
<td>0.449 s</td>
<td>11.46 M</td>
<td>31.64 M</td>
<td>1.43 s</td>
<td>1.88 s</td>
</tr>
<tr>
<td>Eye/PSAH</td>
<td>0.581 s</td>
<td>5.482 M</td>
<td>14.49 M</td>
<td>0.71 s</td>
<td>1.29 s</td>
</tr>
</tbody>
</table>
# Heuristic Compare on FForest

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Build</th>
<th>Isects</th>
<th>Traverse</th>
<th>Render</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eye/Median</td>
<td>0.209 s</td>
<td>63.85 M</td>
<td>48.76 M</td>
<td>4.16 s</td>
<td>4.38 s</td>
</tr>
<tr>
<td>Eye/SAH</td>
<td>0.449 s</td>
<td>11.46 M</td>
<td>31.64 M</td>
<td>1.43 s</td>
<td>1.88 s</td>
</tr>
<tr>
<td>Eye/PSAH</td>
<td>0.581 s</td>
<td>5.482 M</td>
<td>14.49 M</td>
<td>0.71 s</td>
<td>1.29 s</td>
</tr>
<tr>
<td>DoF/Median</td>
<td>0.208 s</td>
<td>999.1 M</td>
<td>794.4 M</td>
<td>65.8 s</td>
<td>66.0 s</td>
</tr>
<tr>
<td>DoF/SAH</td>
<td>0.472 s</td>
<td>177.2 M</td>
<td>519.5 M</td>
<td>23.3 s</td>
<td>23.8 s</td>
</tr>
<tr>
<td>DoF/PSAH</td>
<td>0.580 s</td>
<td>92.60 M</td>
<td>314.7 M</td>
<td>14.5 s</td>
<td>15.1 s</td>
</tr>
</tbody>
</table>
Conclusions

- The perspective transform can produce more effective (ray-specialized) acceleration structures for off-axis rays.

- A new cost metric produces better results in perspective space.
Section Conclusions
Section Conclusions

• Many specialized acceleration structures are required for maximal rendering performance
  – Already true for game rendering algorithms

• Build is cheap (especially with scan and lazy build from hierarchy)
  – Grid build and BVH refit are really cheap