GPU-based Ray Tracing of Dynamic Scenes
Workshop VR/AR 2008, Magdeburg

Martin Reichl · Robert Dünger · Alexander Schiewe · Markus Hartleb
Thomas Klemmer · Christopher Lux · Bernd Fröhlich

26th of September, 2008

Bauhaus-Universität Weimar · Fakultät Medien
Systeme der Virtuellen Realität
Motivation & Goals
## Related Work

<table>
<thead>
<tr>
<th>Static Scenes</th>
<th>Chip</th>
<th>Data Structures</th>
<th>Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horn et. al. 2007</td>
<td>GPU</td>
<td>Kd</td>
<td>Short Stack</td>
</tr>
<tr>
<td>Günther et. al. 2007</td>
<td>GPU</td>
<td>BVH</td>
<td>Shared Stack</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dynamic Scenes</th>
<th>Chip</th>
<th>Data Structures</th>
<th>Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenRT 2002</td>
<td>CPU</td>
<td>Kd + Kd</td>
<td>Full Stack</td>
</tr>
</tbody>
</table>
Related Work

<table>
<thead>
<tr>
<th>Static Scenes</th>
<th>Chip</th>
<th>Data Structures</th>
<th>Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horn et. al. 2007</td>
<td>GPU</td>
<td>Kd</td>
<td>Short Stack</td>
</tr>
<tr>
<td>Günther et. al. 2007</td>
<td>GPU</td>
<td>BVH</td>
<td>Shared Stack</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dynamic Scenes</th>
<th>Chip</th>
<th>Data Structures</th>
<th>Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenRT 2002</td>
<td>CPU</td>
<td>Kd + Kd</td>
<td>Full Stack</td>
</tr>
<tr>
<td>This work</td>
<td>GPU</td>
<td>BVH + Kd</td>
<td>Smart Stack</td>
</tr>
</tbody>
</table>
Approach – Two Level Hierarchy
Approach – Two Level Hierarchy
GPU-based Ray Tracing of Dynamic Scenes – GI VR/AR 2008

Load Model

• Load .obj + .mtl
• Generate Textures
  • Materials
  • Geometry (Vertices, Normals)
  • Transformation Matrices

Display

CPU  GPU
Load Model

Preprocessing

Build BVH

• OOBB Processing
• Build HQ Kd-tree for each object
• Serialize Kd-trees
• Generate Texture

Display

Build BVH

• Build BVH < 10ms
• Serialize BVH
• Generate Texture

CPU

GPU
GPU-based Ray Tracing of Dynamic Scenes – GI VR/AR 2008

Load Model

Preprocessing

Build BVH

OpenGL Prim. Rays

Raytrace Kernel

Display

Traversal
- Traverse BVH
- Transform Ray
- Traverse Kd-tree
- Intersect Triangles
- Shade + Shadow (Traversal)
- Process Secondary Rays

CPU

GPU
(Re-)Introducing CUDA
Branching Join Points

- Cast Ray
- Hit?
  - Shading
  - Push Shadow Ray
  - Transparency?
    - Push Refraction Ray
  - Reflectivity?
    - Push Reflection Ray
- Stack empty?
- Pop Ray

May be masked out

Early Out
CUDA Memory Pyramid (8800GTX)

- Main Memory: 768 MiB
- Constants: 64 KiB
- Shared Memory: 16 KiB
- Registers: 32 KiB
- Main Memory: 768 MiB

Latency vs. Size
Smart Stack

- Shared memory

BVH1 → BVH2 → BVH3 → BVH4 → BVH5

- Start

- Main memory

- End
Smart Stack

Shared memory

BVH1  BVH2  BVH3  BVH4  BVH5  Kd1

Start

End

Main memory
Smart Stack

Shared memory

Start == End

Main memory

GPU-based Ray Tracing of Dynamic Scenes – GI VR/AR 2008
Smart Stack

**Shared memory**

- BVH5
- Kd1

**Start**

**End**

**Main memory**

- BVH1
- BVH2
- BVH3
- BVH4
Smart Stack

- **Shared memory**
  - Kd2
  - Start
  - End
  - BVH5
  - Kd1

- **Main memory**
  - BVH1
  - BVH2
  - BVH3
  - BVH4
Smart Stack

- Shared memory
  - BVH5
  - Kd1
  - Start
  - End

- Main memory
  - BVH1
  - BVH2
  - BVH3
  - BVH4
Smart Stack

- Shared memory
  - Start == End
- Main memory
  - BVH1
  - BVH2
  - BVH3
  - BVH4
Smart Stack

Shared memory

BVH1  BVH2  BVH3  BVH4

Start

End

Main memory
Results & Performance Evaluation

Chevy • 43k tris • 79 objects

Chess • 46k tris • 34 objects
Results & Performance Evaluation

BART Robots · 110k tris · 162 objects

BART Kitchen · 71k tris · 5 objects
Performance Impact Shadow Rays

GPU-based Ray Tracing of Dynamic Scenes – GI VR/AR 2008

Robots
Kitchen
Chevy
Chess

fps(Kd)
fps(TLH)
Performance Impact Secondary Rays

The chart illustrates the performance impact of secondary rays on different scenes using two metrics: fps(Kd) and fps(TLH). The scenes compared are Robots, Kitchen, Chevy, and Chess. The chart shows that the performance impact varies significantly across these scenes, with the highest performance impact observed in the Chevy scene for both fps(Kd) and fps(TLH).
Conclusion & Future Work

- Dynamic Scenes / Two-Level Hierarchy
- Smart Stack
- Performance
- HW Occupancy

today

tomorrow

• Kd-Kd: The better approach?
• Kd-split / Kd-merge
Thank you!