Accelerated Single Ray Tracing for Wide Vector Units

Valentin Fuetterling*, Carsten Lojewski*, Franz-Josef Pfreundt*, Bernd Hamann^ and Achim Ebert*
Introduction – Ray Tracing

• Ray
• Set of triangles
• Find first intersection

• Acceleration
  • Bounding Volume Hierarchy (BVH)
  • Requires traversal
Motivation

• Accelerating single ray traversal on data-parallel hardware

• Why single rays?
  • Ease of use
  • Consistent performance
  • Immediate results
Dissection

CURRENT SINGLE RAY TRAVERSAL
State-Of-The-Art

- Multi-branching BVH
- Data parallelism
- Vector instructions (SIMD)
- Less traversal iteration
- More coherent memory access

[Dammertz et al. 2008; Ernst and Greiner 2008; Wald et al. 2008]
Building Blocks

- Ray Setup
- Inner Node Iteration
  - Bounding Box Intersection
  - Node Ordering
  - Stack Operation
- Leaf Node
  - Triangle Intersection
  - Stack Compaction

Diagram:

1. Ray Setup
2. Inner Node Iteration
3. Bounding Box Intersection
4. Node Ordering
5. Stack Operation
6. Leaf Node
7. Triangle Intersection
8. Stack Compaction

Fraunhofer
Node Ordering: Heuristics

- Ideal traversal order
  - Minimal traversal iterations
  - Unknown

- Distance heuristic
  - Intersection distance

- Sign heuristic
  - Pre-compute order
Data-parallel Limitations?

- Variable number of active nodes $N$
- Treat $N < 3$ as special cases for efficiency
- Branchy code
Accelerated Single Ray Tracing for \textbf{W}ide \textbf{V}ector Units

\textbf{WIVE}
Contributions

- WiVe: Single Ray Traversal Algorithm
- Encoding for sign-based ordering heuristic
- Full data-parallel traversal for multi-branch BVHs
- Constant-time (i.e. branch-less) inner node iteration
WiVe Key Ideas

- Encode fixed traversal orders into nodes
  - One order per ray signs combination (8)
  - Order defined by permute vector
- Map node ordering to a single permute vector operation
- Map stack operation to a single vector compress operation
WiVe Algorithm

• Initially nodes are in memory order
• Permute nodes (A)
• Intersection test (B)
• Compress (C)
• Store to stack (D)
WiVe Node Order

- Binary treelet with split axis labels (a)
- Leaves form BVH8 node cluster
- Split hierarchy determines permute vectors
- BVH8 node cluster in spatial domain (b)

- Ray with +x and -y signs (octant is 10b = 2)
- If signs[split axis] negative: Swap default order

Permute_vector[2] = 1 0 6 7 3 2 5 4
WiVe Configurations (BVH8-half)

- BVH branching-factor equals half the vector width
- E.g BVH8 with 16-wide vector instructions
- $t_{\text{min}}$ and $t_{\text{max}}$ values interleaved in register
- Child offset and $t_{\text{min}}$ interleaved on stack
WiVe Configurations (BVH8-full)

- BVH branching-factor equals full vector width
- E.g. BVH8 with 8-wide vector instructions
- $t_{\text{min}}$ and $t_{\text{max}}$ values in separate registers
- Child offset and $t_{\text{min}}$ on separate stacks
WiVe
EVALUATION
Experimental Setup

- Compare WiVe to Embree 2.15.0 single ray
  - Integrated into Embree Protoray benchmark
  - WiVe BVH derived from Embree BVH, same topology
  - BVH uses SAH-based binning (no spatial splits)
  - Path tracing with 8 bounces
- Two implementations of WiVe
  - BVH8-half for AVX-512 (16-wide)
  - BVH8-full for AVX2 (8-wide)
Input / Output

- White environment light
- Diffuse shading
- Normal colors
- 3840 x 2160 resolution

Mazda (5.7M)  Art Deco (10.5M)
San Miguel (10.7M)  Villa (37.5M)
Powerplant (12.8M)
Results – Order Heuristic

- Sign-based vs. distance-based heuristic
- Per-ray average indicators
- Very close, slight bias towards distance
- Worst case is San Miguel with 6-7% discrepancy

<table>
<thead>
<tr>
<th></th>
<th>Inner Nodes</th>
<th>Leaves</th>
<th>Triangles</th>
<th>Inner Nodes</th>
<th>Leaves</th>
<th>Triangles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mazda</td>
<td>14.1</td>
<td>3.6</td>
<td>4.1</td>
<td>14.1</td>
<td>3.6</td>
<td>4.1</td>
</tr>
<tr>
<td>San Miguel</td>
<td>21.2</td>
<td>4.5</td>
<td>5.4</td>
<td>21.2</td>
<td>4.2</td>
<td>5.1</td>
</tr>
<tr>
<td>Art Deco</td>
<td>11.1</td>
<td>2.3</td>
<td>2.9</td>
<td>11.0</td>
<td>2.2</td>
<td>2.8</td>
</tr>
<tr>
<td>Powerplant</td>
<td>20.5</td>
<td>5.6</td>
<td>9.3</td>
<td>20.3</td>
<td>5.6</td>
<td>9.2</td>
</tr>
<tr>
<td>Villa</td>
<td>17.4</td>
<td>4.6</td>
<td>5.5</td>
<td>17.4</td>
<td>4.5</td>
<td>5.4</td>
</tr>
</tbody>
</table>
Results – Performance (BVH8-half)

- AVX-512 implementation
- Native permute and compress instructions
- Measured on Intel® Xeon Phi™ 7250 @ 1.4GHz (Knight’s Landing)
- Timings include full (off-screen) rendering

<table>
<thead>
<tr>
<th></th>
<th>MAZDA</th>
<th>SAN MIGUEL</th>
<th>ART DEC0</th>
<th>POWERPLANT</th>
<th>VILLA</th>
</tr>
</thead>
<tbody>
<tr>
<td># triangles[M]</td>
<td>5.7</td>
<td>10.5</td>
<td>10.7</td>
<td>12.8</td>
<td>37.5</td>
</tr>
<tr>
<td>AVX-512</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WiVe</td>
<td>126.7</td>
<td>73.1</td>
<td>165.0</td>
<td>85.4</td>
<td>87.4</td>
</tr>
<tr>
<td>Embree</td>
<td>110.0</td>
<td>63.2</td>
<td>143.4</td>
<td>68.4</td>
<td>76.3</td>
</tr>
<tr>
<td>WiVe[+]</td>
<td>15</td>
<td>16</td>
<td>15</td>
<td>25</td>
<td>15</td>
</tr>
</tbody>
</table>
Results – Performance (BVH8-full)

- AVX2 implementation
- Native permute instruction, compress emulated with permute + shift + look-up table (1kb)
- Measured on dual-socket Intel® Xeon™ E5 2680v3 @ 2.5GH (Haswell)

<table>
<thead>
<tr>
<th></th>
<th>MAZDA</th>
<th>SAN MIGUEL</th>
<th>ART DECO</th>
<th>POWERPLANT</th>
<th>VILLA</th>
</tr>
</thead>
<tbody>
<tr>
<td># triangles[M]</td>
<td>5.7</td>
<td>10.5</td>
<td>10.7</td>
<td>12.8</td>
<td>37.5</td>
</tr>
<tr>
<td>AVX2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WiVe</td>
<td>74.0</td>
<td>46.0</td>
<td>97.2</td>
<td>57.4</td>
<td>48.8</td>
</tr>
<tr>
<td>Embree</td>
<td>70.9</td>
<td>43.0</td>
<td>93.7</td>
<td>51.9</td>
<td>46.2</td>
</tr>
<tr>
<td>WiVe[+%]</td>
<td>4</td>
<td>7</td>
<td>4</td>
<td>11</td>
<td>6</td>
</tr>
</tbody>
</table>
Conclusion & Outlook

• Introduced the WiVe algorithm
  • Fully vectorized SIMD single ray traversal
  • Branch-free traversal iteration for multi-branching BVHs
  • Fastest on CPUs

• Combine WiVe with compression
• Investigate WiVe on GPUs
  • Trade throughput for latency
Thank you!

Email: valentin.fuetterling@itwm.fraunhofer.de
Blog: http://rapt.technology