Ordered Depth-First Layouts for Ray Tracing

Jae-Ho Nah\textsuperscript{1}, Jeong-Soo Park\textsuperscript{1}, Jin-Woo Kim\textsuperscript{1}, Chanmin Park\textsuperscript{2}, Tack-Don Han\textsuperscript{1}

Yonsei University, Korea\textsuperscript{1}
Samsung Electronics, Korea\textsuperscript{2}
Contents

- Background & Related Work
- Proposed method: Ordered depth-first layout (ODFL)
  - Comparison with depth-first layout
  - Tree construction and traversal for ODFL
- Experimental results
- Conclusions
Contents

- Background & Related Work
  - Proposed method: Ordered depth-first layout (ODFL)
    - Comparison with depth-first layout
    - Tree construction and traversal for ODFL
  - Experimental results
  - Conclusions
Background

- Kd-tree
  - Axis-aligned BSP (binary space partitioning) tree
  - Widely used for computer graphics (e.g. ray tracing)

- Ray tracing with kd-trees
  - Requires many visits to nodes.
  - Important to design cache-efficient layouts
Related Work

Depth-first layouts [PH10]
- Basic tree representation from recursive tree building
- Locality between the parent and the left child node
- One pointer per node

Subtree layouts [Hav99]
- Made by clustering nodes
- Locality between the parent and two child nodes
- Two pointers per node (ordinary subtree)

These images are excerpted from [Hav 99].
Contents

- Background & Related Work
- Proposed method: Ordered depth-first layout (ODFL)
  - Comparison with depth-first layout
  - Tree construction and traversal for ODFL
- Experimental results
- Conclusions
Proposed Method

• Goals
  • Improve the cache efficiency of depth-first layouts
  • No additional memory space (8 bytes per node)

• Our approach
  • The probability of a ray intersecting with a node is proportional to its surface area. [MB90]
  • Change the arrangement criterion of child nodes: geometric position $\rightarrow$ surface area
Traditional Depth-First Layout

- Child nodes are arranged by their geometric position [PH10] (left node ≤ split plane ≤ right node)
Ordered Depth-First Layout (ODFL)

- Child nodes are arranged by their surface area (SA) (left node > right node)
Tree Construction and Traversal for ODFL

- Tree construction
  - SA values is obtained by a surface area heuristic (SAH)
  - Add a 1-bit reorder flag (embedded into a 8-byte node)
- Tree Traversal
  - For front-to-back traversal, the reorder flag is referenced
Contents

- Background & Related Work
- Proposed method: Ordered depth-first layout (ODFL)
  - Comparison with depth-first layout
  - Tree construction and traversal for ODFL
- Experimental results
- Conclusions
Experimental Setup

- Whitted ray tracer
  - Single-ray recursive tracing
  - Recursion depth 4
  - SAH-based tree build
- Benchmark scenes (512x512 resolution)
  - Kitchen
  - Fairy
  - Sponza
- Dinero IV cache simulator [EH98]
  - 8KB size
  - 4-way set associative
  - 64byte block size
Results

ODFL reduced the required memory bandwidth by
• 15-30% compared with the depth-first layout
• 10-21% compared with the ordinary subtree layout.
Results

- 40% less than the ordinary subtree layout
  - DFL/ODFL: 8 nodes per 64B cache block
  - Ordinary subtree: 5 nodes per 64B cache block
Contents

- Background & Related Work
- Proposed method: Ordered depth-first layout (ODFL)
  - Comparison with depth-first layout
  - Tree construction and traversal for ODFL
- Experimental results
- Conclusions
Conclusions

- Maximize parent-child locality using simple node ordering
- Platform independent
  - Widely applicable to ray tracers based on CPUs, GPUs, and dedicated hardware
- Can be useful for other applications utilizing depth-first search
  - Collision detection, photon mapping, etc.
Acknowledgements

- This work was supported by Samsung Electronics Co., Ltd
Q&A