Node Culling Multi-Hit BVH Traversal: Supplemental Materials

Christiaan Gribble†

Applied Technology Operation
SURVICE Engineering

As noted in Section 3 of the main text, we provide graphs comparing average frame time for the find-some-intersections case for each test scene in Figure 1. Similarly, we provide data for ray/node intersection tests, node traversal operations, ray/primitive intersection tests, and average frame time in Table 1.

We also provide source code for the experimental reference implementation used to evaluate our node culling multi-hit BVH traversal algorithm in Section 3 of the main text. The key elements of this project include:

- **BVH.h** defines and implements a BVH with optional (compile-time) support for every- and leaf-node TCBs, support for unconstrained ICBs, and convenience functions for standard first-hit traversal. As noted in Section 3 of the main text, the BVH structure is built using a readily available surface area heuristic construction algorithm [WBS07].
- **CMakeLists.txt** provides content for compiling the reference implementation using the CMake build system.
- **Constants.h** defines useful values employed throughout the reference implementation, including the default number of primitives at which to construct a leaf node.
- **Node.h** defines and implements an intersectable BVH node using Williams-style ray/box intersection [WBMS05].
- **Renderer.{h,cc}** define and implement an abstract base class from which specific renderers are derived, as well as several inner classes, including per-ray payload data.
- **RendererFH.{h,cc}** define and implement a standard first-hit renderer with simple eyelight shading.
- **RendererMH.{h,cc}** define and implement the various multi-hit renderers described in this work, including naive multi-hit and uICB-, eTCB-, and lTCB-based node culling multi-hit renderers. Each multi-hit renderer defines callback functions appropriate to its intended behavior, and all multi-hit renderers inherit an intermediate multi-hit base class that provides a shader implementing either eyelight shading at first-hit visible surfaces or alpha-blending across multiple surfaces, depending on the current rendering configuration.
- **mhBVH.cc** implements a driver program for rendering images using any of the renderers described above, configurable at runtime via various command line arguments.

The README file in the top-level source directory provides instructions for building and running the driver program.

Source code in the common/ subdirectory provides common elements used by the reference implementation (file I/O, math operations, and so forth), but is not directly related to the multi-hit ray traversal techniques described in this work.

Similarly, the scenes/ subdirectory provides example scene, view, and geometry/material files for the Cornell Box scene. To render different scenes, generate files that mimic the basic structure and format of the Cornell Box example.

Access to the most recent stable release of the reference implementation is available via the project homepage at:

http://www.rtvtk.org/~cgribble/research/mhBVH/

Additionally, read-only access to the reference implementation development repository is available via HTTP with git:

    git clone http://www.rtvtk.org/code/mhBVH.git

Unless otherwise stated directly in the source, the reference implementation is distributed under the BSD 3-Clause License. Please see the LICENSE file distributed with the source for more information.

References


Figure 1: Performance of multi-hit variants for find-some-intersections. Here, graphs compare multi-hit performance in seconds per frame (spf) among multi-hit implementations for various values of $N_{\text{query}}$ in each test scene.
<table>
<thead>
<tr>
<th>n-trav</th>
<th>ICB</th>
<th>naive</th>
<th>ICB</th>
<th>naive</th>
<th>ICB</th>
<th>naive</th>
<th>ICB</th>
<th>naive</th>
</tr>
</thead>
<tbody>
<tr>
<td>naive</td>
<td>0.10</td>
<td>0.10</td>
<td>0.10</td>
<td>0.10</td>
<td>0.10</td>
<td>0.10</td>
<td>0.10</td>
<td>0.10</td>
</tr>
<tr>
<td>fair</td>
<td>5.50</td>
<td>5.50</td>
<td>5.50</td>
<td>5.50</td>
<td>5.50</td>
<td>5.50</td>
<td>5.50</td>
<td>5.50</td>
</tr>
<tr>
<td>conf</td>
<td>4.50</td>
<td>4.50</td>
<td>4.50</td>
<td>4.50</td>
<td>4.50</td>
<td>4.50</td>
<td>4.50</td>
<td>4.50</td>
</tr>
<tr>
<td>truck</td>
<td>1.06</td>
<td>1.06</td>
<td>1.06</td>
<td>1.06</td>
<td>1.06</td>
<td>1.06</td>
<td>1.06</td>
<td>1.06</td>
</tr>
<tr>
<td>tank</td>
<td>7.97</td>
<td>7.97</td>
<td>7.97</td>
<td>7.97</td>
<td>7.97</td>
<td>7.97</td>
<td>7.97</td>
<td>7.97</td>
</tr>
<tr>
<td>plant</td>
<td>10.03</td>
<td>10.03</td>
<td>10.03</td>
<td>10.03</td>
<td>10.03</td>
<td>10.03</td>
<td>10.03</td>
<td>10.03</td>
</tr>
</tbody>
</table>

---

**Table 1: Key metrics for multi-hit performance.**

For each scene, we report the number of ray/node intersection tests (n-ise), node traversal operations (n-trav), ray/primitive intersection tests (p-ise), and average frame time in seconds (avg(f)) before correcting the multi-hit query for each implementation. As expected, calling culling significantly reduces the metrics for multi-hit work, but is identical across node culling implementations. Differences in average frame time among culling techniques thus arise from the relative number of times the required callback functions are invoked.