Hierarchical Raster Occlusion Culling

Gi Beom Lee, Moonsoo Jeong, Yechan Seok, Sungkil Lee
Sungkyunkwan University, South Korea

Eurographics 2021
Occlusion Culling (OC)

- Bypass rendering of occludees hidden by visible occluders
  - Crucial in accelerating geometry rendering
Online Occlusion Culling

- **On-the-fly visibility query from a view point**
  - No preprocessing
  - **Screen-space tests** are common

- **Two categories**
  - Per-bound test within the axis-aligned screen-space bounds
  - Per-fragment test within rasterized per-object bounds

- ![Axis-aligned bounds](image1)
- ![Rasterization of per-object bounds](image2)
Per-Bound Test

• Test the axis-aligned bounds
  • Highly scalable, resulting from batch tests
  • Lower culling efficiency with wider conservative bounds
  • Reprojection often replaces occlusion map [Sousa 2011; Lee 2018]

• Screen-space hierarchy is common for efficient tests
  • [Zhang et al. 1997; Coorg et al. 1997]

Per-Fragment Test

• In the rasterized object bounds
  • Test any of the fragments are visible against the occlusion map
  • Higher culling efficiency from tighter bounds than axis-aligned bounds

• GPU-Driven Techniques
  • Hardware occlusion query (HOQ): counter-based
  • Raster occlusion culling (ROC): direct tagging
Hardware Occlusion Query (HOQ)

- Count the fragments that pass the depth test
  - The query itself is fast, but read-back of the results causes stalls

- Hierarchical Queries
  - CHC, CHC++ [Bittner et al. 04; Mattausch et al. 08]
  - Cuts are maintained based on temporal coherence.
  - Can use batch queries, and suppress redundant queries largely
  - Too many queries (for complex scenes) lead to non-trivial latency.

Raster Occlusion Culling (ROC)

- **Direct Tagging to Vis. Buffer** [Kubisch et al. 2014; Boudier et al. 2015]
  - Object visibilities are tagged, only when any of fragments pass the early-Z.
  - Indirect multidraw can avoid read-back of the counters.

- **Iterating over every objects** can be costly for complex scenes.
  - Motivate our hierarchy-based approach for scalability

```glsl
// GLSL fragment shader
// from ARB_shader_image_load_store
layout(early_fragment_tests) in;

buffer visibilityBuffer{
  int visibility[]; // cleared to 0
};

flat in int objectId; // unique per box

void main()
{
  visibility[objectId] = 1;
  // no atomics required (32-bit write)
}
```

Boudier, Pierre, and Cristoph Kubisch. "GPU Driven Large Scene Rendering." Presentation at the GPU Technology Conference (GTC 2015), San Jose, CA, USA. 2015.
Our Contributions

- **Hierarchical ROC (HROC)**
  - Potential occludees are grouped in BVH and tested in a batch.

- **Hybrid culling: rasterization + ray casting**
  - Rasterize the bounds of the groups
  - Per-pixel ray casting tests visibilities of the individual objects.
  - The hierarchy is traversed *implicitly* in the ray casting, which avoids the hierarchical iteration in the host.
HROC (Ours) vs. ROC

- **Ours rasterizes fewer bounds for occlusion test**
  - Example scene: 420 frames, 73.8K objects, 147.6K BVH nodes
  - HROC (Ours): 272 boxes / frame
  - ROC: 73.8K boxes / frame (i.e., all objects are rasterized)
Algorithms
Occlusion Map Rendering

• Selection of potential occluders
  • Exploit temporal coherence

• Selection of effective occluders
  • Potential occluders passing early-Z
Extraction of Occludee Groups

• Find potential occludee group
  • Bottom-up traversal on the object-level BVH
  • Ancestors of visible node (n) are also visible (V)
  • The visibility of the remainder (G) are unknown
Batch Occlusion Test

- Rasterize the bounds of occludee groups with Early-Z
  - Fully hidden occludee groups are implicitly culled.
- Cast rays for each fragment
  - Fine-grained tests for individual occludee objects
  - Ray casting should not stop at the nearest intersection.
Occludee Rendering

• Render potentially visible occludees with early-Z
  • Add visible occludees to effective occluders rendered in Occlusion Map Rendering stage

• Set newly potential occluders for next frame
Accelerations
Packed Multidraw with Counter Read-Back

- GPUs do not well handle *void objects* in the command buffer.

<table>
<thead>
<tr>
<th>object indices</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>visibility</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>...</td>
</tr>
</tbody>
</table>
Packed Multidraw with Counter Read-Back

- Packing can avoid handling void objects.

<table>
<thead>
<tr>
<th>object indices</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>visibility</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>packed indices</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Packed Multidraw with Counter Read-Back

• Read-back of draw counter

```c
void glMultiDrawElementsIndirectCount( ..., drawcount, maxdrawcount, ... );
```

• drawcount can be dereferenced in GPU, but we still need to provide another counter, `maxdrawcount`.

• The standard usage is using the number of the entire objects, but this causes still handling the void objects.

• Hence, we read back the counter of valid objects, and use it for the count.

• We can read the counters three times:
Occluder Filtering

- Early-Z may include non-trivial false positives
  - Rendered earlier than their blockers

- Selecting only true positives with an item buffer
  - can reduce potential occluders, and more occludees can be culled.
Experimental Analysis
Test Configurations

• **Platform**
  • Intel Core i7-7800X 3.6GHz
  • NVIDIA RTX 2080 Ti
  • OpenGL API in Windows 10

• **Scenes**
  • Four camera-animated scenes: durations of 10s
  • Fantasy (FN), Factory (FC), Radial City (RC), Satellites (ST)
Fantasy (FN)
17.9M Tris, 7.4K Obj, 14.9K BVH nodes (0.5 MB)
Factory (FC)
5.7M Tris, 46.6K Obj, 93.2K BVH nodes (3.0 MB)
Radial City (RC)
1.2M Tris, 73.8K objs, 147.6K BVH nodes (4.7 MB)
Satellites (ST)
807.1M Tris, 120K objs, 240K BVH nodes (7.7 MB)
# Methods to Compare

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOC</td>
<td>• No-cull rendering</td>
</tr>
<tr>
<td>VFC</td>
<td>• View frustum culling</td>
</tr>
<tr>
<td>REF</td>
<td>• Ideal reference rendering</td>
</tr>
</tbody>
</table>
| CHC++   | • [Mattausch et al. 2008]  
• The state-of-the-art hierarchical culling |
| ROC     | • [Kubisch et al. 2014; Boudier et al. 2015] |
| WOC     | • [Lee et al. 2018]  
• The state-of-the-art image-space culling |
| IOC     | • [Lee et al. 2020]  
• Simple hierarchy-based ROC culling |
Performance (FHD)

- From small to complex scenes, ours performs best
  - Occluder filtering (OF): best for FN (small-scale scene)
  - Counter read-back (CR)+OF: best for medium to large-scale
  - Ours is less sensitive to the number of objects
• From small to complex scenes, ours performs best
  • Occluder filtering (OF): best for FN (small-scale scene)
  • Counter read-back (CR)+OF: best for medium to large-scale
  • Ours is less sensitive to the number of objects
Performance (FHD)

- From small to complex scenes, ours performs best
  - Occluder filtering (OF): best for FN (small-scale scene)
  - Counter read-back (CR)+OF: best for medium to large-scale
  - Ours is less sensitive to the number of objects
Performance (FHD)

- From small to complex scenes, ours performs best
  - Occluder filtering (OF): best for FN (small-scale scene)
  - Counter read-back (CR)+OF: best for medium to large-scale
  - Ours is less sensitive to the number of objects
Performance (FHD)

- From small to complex scenes, ours performs best
  - Occluder filtering (OF): best for FN (small-scale scene)
  - Counter read-back (CR)+OF: best for medium to large-scale
  - Ours is less sensitive to the number of objects
Performance (4K UHD)

- **Ours performs faster for complex scenes**
  - Per-pixel ray casting is still scalable in higher resolution
  - ROC performs slightly faster for FC.

---

Occluder Filtering (OF), Counter Read-back (CR)
Effects of Accelerations

• Counter read-back (CR)
  • 1.10X speedup with respect to NA on average

• Occluder filtering (OF): best for FN scene
  • 1.10X speedup with respect to NA on average

• CR + OF: best for FC, RC, ST scenes
  • 1.29X speedup with respect to NA on average
Camera Sequences

Performance (ms) along the camera sequences

- CHC++
- ROC
- WOC
- IOC
- HROC

FC (zoom-in)
Camera Sequences

Performance (ms) along the camera sequences

- CHC++
- ROC
- WOC
- IOC
- HROC

RC (zoom-in)
Limitations and Conclusion
Limitations

• Dynamic Object handling
  • Common limitation with hierarchy-based solutions
  • Potential solution: hybrid culling solution (HROC and ROC)

• When a camera lies inside a large occludee group
  • Excessively large bounds are tested with ray casting
  • Potential solution: subdividing the box down to its children
Conclusion

• Hierarchical ROC
  • Scalable batch tests of coarse occludee groups

• Two-phase hybrid culling
  • Rasterization of occludee groups to initiate per-pixel tests
  • Per-pixel ray casting for fine-grained visibility tests
Thank you for your attention!