Clustered Deferred and Forward Shading

Clustered Shading

Ola Olsson, Markus Billeter and Ulf Assarsson
Chalmers University of Technology
Brief summary of properties
Tiled Shading recap
Tiled Shading Problems
Clustered Shading
  - Algorithm
  - Results
Real-time shading algorithm
- Thousands of lights
  - Limited range light
  - No shadows
- Fully dynamic
  - Lights and Geometry

Robust performance
- Low overhead
- Low view-dependence
- Scales to 1M lights.
- Handles noisy depth distributions
Screen Space Tiles
- E.g. 32x32 pixels
- Each contains list of lights

For each light
- Find screen space AABB
- Add to tiles
Tiled Shading Recap - The Light Grid

- Screen Space Tiles
  - E.g. 32x32 pixels
  - Each contains list of lights

- For each light
  - Find screen space AABB
  - Add to tiles
Tiled Shading Recap - The Light Grid

- Screen Space Tiles
  - E.g. 32x32 pixels
  - Each contains list of lights

- For each light
  - Find screen space AABB
  - Add to tiles
Tiled Shading Recap -The Light Grid

- Screen Space Tiles
  - E.g. 32x32 pixels
  - Each contains list of lights

- For each light
  - Find screen space AABB
  - Add to tiles
Tiled Shading Recap
-The Light Grid

- Screen Space Tiles
  - E.g. 32x32 pixels
  - Each contains list of lights

- For each light
  - Find screen space AABB
  - Add to tiles
Tiled Shading Recap
-The Light Grid

- Screen Space Tiles
  - E.g. 32x32 pixels
  - Each contains list of lights

- For each light
  - Find screen space AABB
  - Add to tiles
out vec4 resultColor;

void main()
{
    vec3 color = texelFetch(colorTex, gl_FragCoord);
    vec3 specular = texelFetch(specularTex, gl_FragCoord);
    vec3 normal = texelFetch(normalTex, gl_FragCoord);
    vec3 position = fetchPosition(gl_FragCoord);

    vec3 shading = accumulate for each light in tile;

    resultColor = vec4(shading, 1.0);
}
Tiled Shading Recap
- Depth Range Optimization
Tiles In 3D® - the movie™
UDK Necropolis CTF
Tiled Shading
-The Discontinuity Dysfunction

- Unreal 3 Scene
  - ‘Necropolis’
- Tricky view
  - But not artificial

- 2D tiles with depth bounds
- High light density

View from Camera

3D bounds for Tiles
Tiled Shading
-The Discontinuity Dysfunction

- Black = zero
- Green ~ 150
- White > 300

# Lights / Tile

- Tiled: 18.8ms
- Clustered: 9.4 ms

3D bounds for Tiles
Clustered Shading - Our solution
Clustered Shading
-Idea

- Add the 3rd dimension
  - Tile also in depth direction = cluster
  - Also > 3 dimensions (e.g. normals)
- Bounded volume around samples
  - Shading cost $\sim$ Light density.
- New Challenges
  - Many more (potential) clusters
    - Must find those actually used
  - Adding lights no longer screen space
Clustered Shading - Algorithm

1. Rasterize G-Buffers
   - (Forward: pre-z pass)
2. Cluster assignment
3. Find unique clusters
4. Assign lights to clusters
5. Shade view samples
Cluster Assignment

- Cluster key:
  - \((i, j, k, n)\)
  - \(i, j\) – 2D tile id
  - \(k\) – \(\log(\text{view z})\)
  - \(n\) – quantized normal

\[ d_k = h_k \]
In 2D just use full grid
With 3D, too many potential cells
  - Especially with normal
  - E.g. $60 \times 34 \times 300 \times 6 \times 3 \times 3 = 31\text{M}$

Two approaches tested
  - Sorting tiles locally
  - Global page table
    - *Virtual* Grid
Finding Unique Clusters - Tile Sorting

- Local Tile sorting
  - Shared memory
- Global prefix
  - small
- Meta data reduction
  - Normal Cones
  - Aabbs
### Finding Unique Clusters - Page Tables

- *Virtual* Grid (or range)
- Very Quick
- 2 Passes
  - Fermi
- 1 Pass
  - Kepler
Light Assignment

- More clusters
- More lights
- Hierarchical approach
  - Hierarchy over lights
  - Also possible
    - Hierarchy over clusters
    - Maybe better
  - Or Both
    - Probably best...
Results

- Crytek Sponza
  - +Trees
  - 10k Lights
Results

Animation

- UDK Necropolis
  - Real game scene (no trees added)
- ~2M polys
- Normal maps
- ~2500 lights
  - In scene ~650 lights
  - Canons adding ~1800 more
- Animation on your USB stick
Results

<table>
<thead>
<tr>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>20</td>
</tr>
</tbody>
</table>

- TiledDeferred
- ClusteredForward
- ClusteredDeferred
Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (ms)</th>
<th>Light Comps.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tiled Shading</td>
<td></td>
<td>~3M</td>
</tr>
<tr>
<td>Clustered Shading</td>
<td></td>
<td>~2.5M</td>
</tr>
</tbody>
</table>

*Note: Actual data points for time and light calculations are not shown in the image.*
Results

Tiled Shading
Frame: ~6 ms

Clustered Shading
Frame: 7 ms
Results

Time (ms)

TiledDeferred
ClusteredForward
ClusteredDeferred
Results

Tiled Shading

Frame: ~20 ms

Clustered Shading

Frame: 11 ms
Results

Tiled Shading
Light Comps. ~35M

Clustered Shading
Light Comps. ~15M
Results

![Graph showing time in milliseconds for different rendering methods: TiledDeferred, ClusteredForward, ClusteredDeferred.](image)
Results

Tiled Shading
Frame: ~21 ms

Clustered Shading
Frame: 15 ms
Results

Tiled Shading
Light Comps. ~33M

Clustered Shading
Light Comps. ~17M
Results

Time (ms)

TiledDeferred
ClusteredForward
ClusteredDeferred
Questions?

Demo implementation available soon
- Well, perhaps not until after summer...
http://www.cse.chalmers.se/~olaolss
Culled Slides…
■ Motivation
  ▪ MSAA G-Buffers (each): 
    \[1920 \times 1080 \times 8 \times 16 = 250\text{Mb}\]
  ▪ Custom Shaders.
  ▪ Transparency.
■ Tiled Forward Shading
  ▪ \(~4x\) vs. Clustered Deferred (necropolis)
■ Clustered Forward Shading
  ▪ \(~2x\) vs. Clustered Deferred
  ▪ Significant improvement.
Implicit Cluster Bounds

- Index gives
  - Sub-frustum

- Quantized normal gives
  - Cone
  - Quantized, e.g. 6x3x3
Explicit Cluster Bounds

- Cone for normals
- Aabb for positions
- Extra cost for construction
  - Not always offset
- Not easy with page tables.
  - Lots of atomics with collisions