FORWARD+: BRINGING DEFERRED LIGHTING TO THE NEXT LEVEL

AMD
Takahiro Harada, Jay McKee, Jason Yang
DIRECT LIGHTING

- Lighting = direct lighting + indirect lighting
  - This paper focuses direct lighting

- For each light source, evaluate light intensity, BxDF, visibility.
- Accumulate multiply of three terms

\[ L = \sum_{i}^{n} \{ L_{ef}(x, w_i, w_o)V(w_o) \} \]

Light intensity, BxDF, Visibility
REAL-TIME SOLUTION FOR RENDERING EQUATION

- Forward rendering
  - Limit the number of lights to be evaluated
    - Pick $m$ lights for each object
  - Limited visibility computation
    - visibility is not calculated for all the lights

\[
L_{\text{forward}} = \sum_{i}^{m} \{ L_{ef}(x, w_i, w_o) V'(w_o) \}
\]

\[
m \leq \tilde{n} \leq n
\]
REAL-TIME SOLUTION FOR RENDERING EQUATION

- **Forward rendering**
  - Limit the number of lights to be evaluated
    - Pick m lights for each object
  - Limited visibility computation
    - Visibility is not calculated for all the lights

- **Deferred rendering**
  - Increase the number of lights
  - Separation of light term and BxDF (shading)

\[
L_{\text{forward}} = \sum_{i}^{m} \left\{ L_{e} f(x, w_{i}, w_{o}) V'(w_{o}) \right\}
\]

\[
L_{\text{deferred}} = \sum_{i}^{\tilde{n}} \left\{ L_{e} V'(w_{o}) \right\} f(x, w_{i})
\]

\[
m \leq \tilde{n} \leq n
\]
• Deferred works well for GPUs on console

• What is the rendering pipeline for today’s high end GPUs?
  – It is different from console GPUs
  – ALU/Mem ratio is higher
  – Branch penalty is smaller
REAL-TIME SOLUTION COMPARISON

- Forward+

\[ L_{\text{forward+}} = \sum_{i} \{ L_{ef}(x, w_i, w_o)V'(w_o) \} \]

- Rendering equation

\[ L = \sum_{i} \{ L_{ef}(x, w_i, w_o)V(w_o) \} \]

\[ m \leq \tilde{n} \leq n \]
REAL-TIME SOLUTION COMPARISON

- **Forward+**

\[
L_{\text{forward+}} = \sum_{i}^{\tilde{n}} \{L_e f(x, w_i, w_o)V'(w_o)\}
\]

- **Rendering equation**

\[
L = \sum_{i}^{n} \{L_e f(x, w_i, w_o)V(w_o)\}
\]

- **Forward**

\[
L_{\text{forward}} = \sum_{i}^{m} \{L_e f(x, w_i, w_o)V'(w_o)\}
\]

- **Deferred**

\[
L_{\text{deferred}} = \sum_{i}^{\tilde{n}} \{L_e V'(w_o)\} f(x, w_i)
\]

\[m \leq \tilde{n} \leq n\]
FORWARD+

- Extension of Forward rendering pipeline
  - Do not limit material usage
- Extension of Deferred rendering pipeline
  - Keep the capability of using many lights

Forward+ == Forward + Light Culling
FORWARD RENDERING PIPELINE

- Depth prepass
  - Fills z buffer
    - Prevent overdraw for shading

- Shading
  - Geometry is rendered
  - Pixel shader
    - Iterate through light list set for each object
    - Evaluates materials for the lights
FORWARD+ RENDERING PIPELINE

- Depth prepass
  - Fills z buffer
    - Prevent overdraw for shading
    - Used for pixel position reconstruction for light culling
- Light culling
  - Culls light per tile basis
  - Input: z buffer, light buffer
  - Output: light list per tile
- Shading
  - Geometry is rendered
  - Pixel shader
    - Iterate through light list calculated in light culling
    - Evaluates materials for the lights
LIGHT CULLING DETAIL

- Implemented using compute shader
- Gather, scatter implementation
  - Gather is simpler
  - See paper for scatter implementation
- Gather implementation
  - Single compute shader
  - A thread group is executed per tile
  - Calculate Z extent
  - Build frustum
  - 64 lights are culled in parallel
  - Overlapped light indices are accumulated in TLS
  - Export
    - One atomic add
    - Write light indices to a contiguous memory (↔ Linked list)
BENEFITS

- Material variety
  - All the information is available in pixel shader
    - No separation of lighting and shading
    - No limitation to BRDFs
  - Improves the pixel quality

- Smaller memory traffic compared to deferred
  - Good for low bandwidth GPUs (e.g., integrated GPUs)
  - Performance increase
BENCHMARK 3,074 DYNAMIC LIGHTS

Forward+ v.s. Compute-based Deferred lighting
WHY FORWARD+ FASTER?

- Depth prepass
  - Write: Depth buffer

- G prepass
  - Write: Depth buffer, Normal buffer

Diagram:
- Radeon HD 6970
- Forward+
- Deferred
- Prepass
- Light Processing
- Final Shading
### WHY FORWARD+ FASTER?

<table>
<thead>
<tr>
<th>Forward+</th>
<th>Deferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth prepass</td>
<td>G prepass</td>
</tr>
<tr>
<td>– Write: Depth buffer</td>
<td>– Write: Depth buffer, Normal buffer</td>
</tr>
<tr>
<td>Light culling</td>
<td>Light accumulation</td>
</tr>
<tr>
<td>– Read: depth, light geometry</td>
<td>– Read: depth, normal, light geometry, light property</td>
</tr>
<tr>
<td>– Compute: culling</td>
<td>– Compute: culling, lighting</td>
</tr>
<tr>
<td>– Write: light list</td>
<td>– Write: light accumulation buffer</td>
</tr>
</tbody>
</table>

#### Diagram:

![Diagram showing comparison between Forward+ and Deferred rendering pipelines]

- **Radeon HD 6970**
- **Prepass**: Red
- **Light Processing**: Pink
- **Final Shading**: Green

<table>
<thead>
<tr>
<th>Time (in 2-second intervals)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>Prepass</td>
</tr>
<tr>
<td>Light Processing</td>
</tr>
<tr>
<td>Final Shading</td>
</tr>
</tbody>
</table>
## WHY FORWARD+ FASTER?

### Forward+
- **Depth prepass**
  - Write: Depth buffer
- **Light culling**
  - Read: depth, light geometry
  - Compute: culling
  - Write: light list
- **Shading**
  - Read: light list, light property
  - Compute: lighting, shading

### Deferred
- **G prepass**
  - Write: Depth buffer, Normal buffer
- **Light accumulation**
  - Read: depth, normal, light geometry, light property
  - Compute: culling, lighting
  - Write: light accumulation buffer
- **Shading**
  - Read: accumulated light color
  - Compute: shading

---

### Graph

![Graph showing comparison between Forward+ and Deferred rendering](image)
AMD LEO DEMO

- Forward+ rendering pipeline
- Dynamic lighting from many lights
- Physically-based BRDFs
- Indirect illumination by dynamic VPLs
- AA

EXTENSIONS

- Deferred has advantages too
- Light culling can be used for deferred
  - G prepass, light culling, screen space shading
- Forward+ can be coupled with screen space effects
  - SSAO
  - Export normal buffer at prepass
  - Fetch AO value from pixel shader for final shading
QUESTIONS?