FORWARD+: BRINGING DEFERRED LIGHTING TO THE NEXT LEVEL

AMD
Takahiro Harada

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

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

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

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

$$m \leq \tilde{n} \leq n$$

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

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

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

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

$$m \leq \tilde{n} \leq n$$

REAL-TIME SOLUTION COMPARISON

- Forward+

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

- Rendering equation

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

$$m \leq \tilde{n} \leq n$$

REAL-TIME SOLUTION COMPARISON

- Forward+

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

- Rendering equation

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

- Forward

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

- Deferred

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

$$m \leq \tilde{n} \leq n$$

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

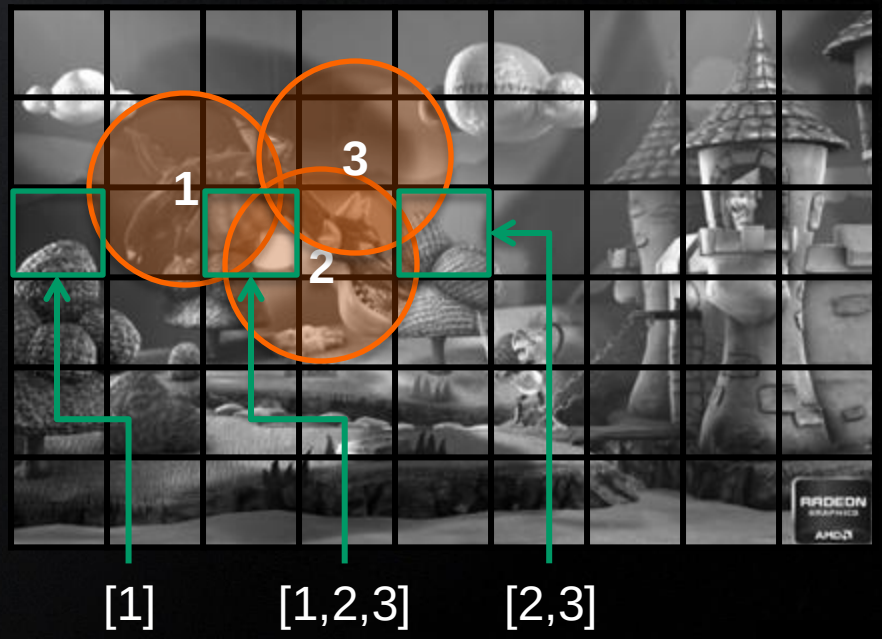
FORWARD RENDERING PIPELINE

- 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)

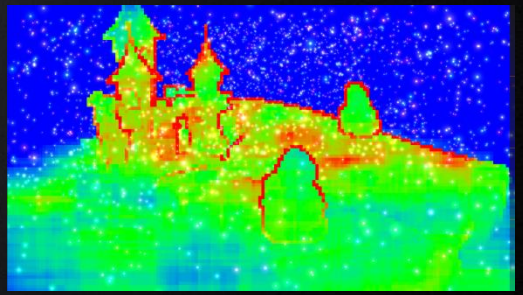
BENEFITS

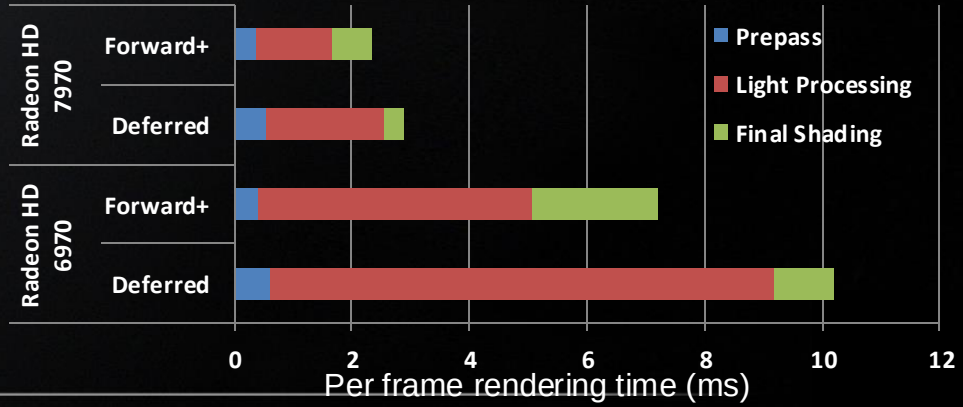
- Material variety
  - All the information is available in pixel shader
    - No separation of lighting and shading
    - No limitation to BRDFs
  - Improves the pixel quality

- 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?

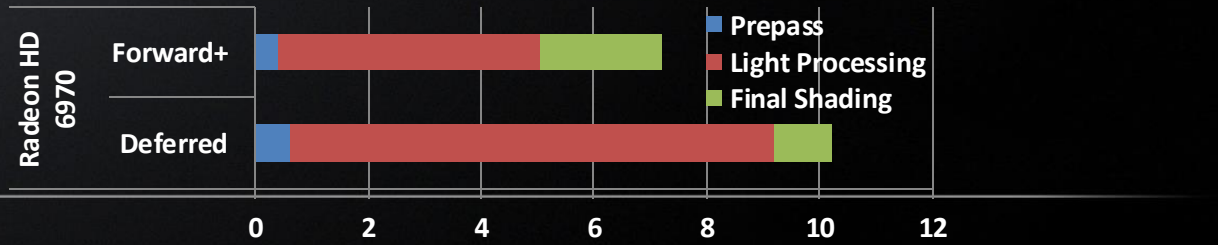
Forward+

- Depth prepass
  - Write: Depth buffer

<

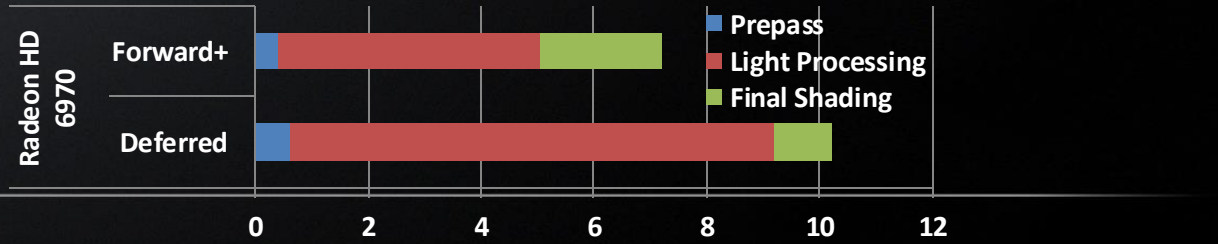
Deferred

- G prepass
  - Write: Depth buffer, Normal buffer



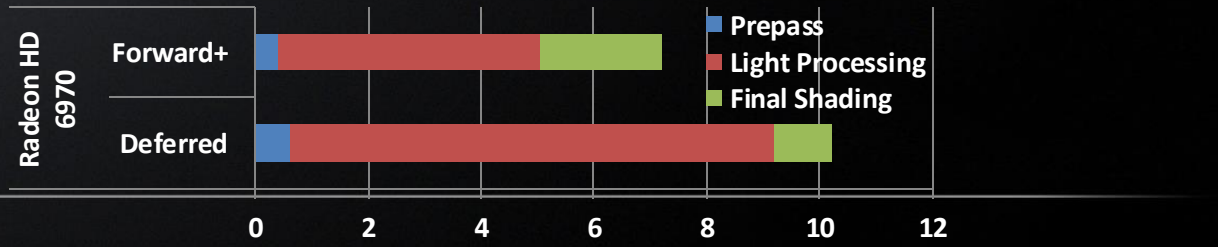
WHY FORWARD+ FASTER?

Forward+		Deferred
▪ Depth prepass		▪ G prepass
– Write: Depth buffer	<	– Write: Depth buffer, Normal buffer
▪ Light culling		▪ Light accumulation
– Read: depth, light geometry	<<	– Read: depth, normal, light geometry, light property
– Compute: culling	<	– Compute: culling, lighting
– Write: light list	<	– Write: light accumulation buffer



WHY FORWARD+ FASTER?

Forward+		Deferred
▪ Depth prepass		▪ G prepass
– Write: Depth buffer	<	– Write: Depth buffer, Normal buffer
▪ Light culling		▪ Light accumulation
– Read: depth, light geometry	<<	– Read: depth, normal, light geometry, light property
– Compute: culling	<	– Compute: culling, lighting
– Write: light list	<	– Write: light accumulation buffer
▪ Shading		▪ Shading
– Read: light list, light property	>	– Read: accumulated light color
– Compute: lighting, shading	>>	– Compute: shading



AMD LEO DEMO

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

http://developer.amd.com/samples/demos/pages/AMDRadeonHD7900SeriesGraphicsReal-TimeDemos.aspx

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