Advanced rendering with Vision Engine

비전엔진을 활용한 향상된 렌더링

네오위즈 CRS 조경준

Overview

1. Deferred Rendering Approach
2. Light Propagation Volumes (Real-time Global Illumination)
3. Screen Space Sub-Surface Scattering
4. Water Rendering

Deferred Rendering Approach

- **Previous Work**

  Forward Rendering

  - Single Pass
  - Multi Pass

  Deferred Rendering

  - Deferred Shading
  - Light Pre-pass Rendering
  - Inferred Rendering

Deferred Rendering Approach

◆ Forward Rendering

Single Pass

For each object
- Object에 영향을 주는 모든 Light를 단일 Shader로 Render

- Shader Combination Explosion
- 모든 Shadow Map을 Memory에 유지해야 함
- 보이지 않는 면에 대한 불필요한 연산

Multi Pass

For each light
- For each object
  - Frame buffer에 Lighting 누적

- Pass 반복에 의한 Vertex Shader, Texture filtering 등 낭비
- 보이지 않는 면에 대한 불필요한 연산

Deferred Rendering Approach

◆ Deferred Shading

For each object
- G-Buffer에 Geometry 및 Material 정보를 Render

For each light
- G-Buffer를 참조하여 Lighting 및 Shading 수행, Frame Buffer에 누적

- Pros
  - 많은 수의 Light에 대응 가능
- Cons
  - Transparent 처리 불가
  - MSAA 처리가 까다로움
  - Material에 제약이 따름
    - 한정된 G-Buffer 공간
    - Material 분기에 따른 Shader 복잡도 증가

Deferred Rendering Approach

◆ Light Pre-pass Rendering

For each object
- G-Buffer에 Geometry 및 Material 정보를 Render

For each light
- G-Buffer를 참조하여 Lighting 수행, Light Accumulation Buffer에 누적

For each object (2nd Pass)
- Light Accumulation Buffer를 참조하여 Material Shading 완료

- Pros
  - 많은 수의 Light에 대응 가능
  - MSAA 대응 가능
  - Material 제약이 없음
  - G-Buffer가 가벼움 - MRT도 요구되지 않음

Deferred Rendering Approach

◆ Light Pre-pass Rendering Cont'd

- Cons
  - Transparent 처리 불가
  - 2nd Pass에 따르는 부담
    - Skinning/Tessellation 적용 시 더욱 증가
    - GPU 성능이 증가할수록 Bottleneck으로 작용
  - Specular 연산이 부정확함
    - 애초에 널리 사용되는 Specular Model이 별로 정확하지 않으므로 무시?
      - Blinn-Phong, Cook-Torrance, …
    - Light Rendering 시 MRT를 이용하여 해결 가능
      - Diffuse(RT0) + Specular(RT1)
      - Light Rendering의 부담이 증가됨

Deferred Rendering Approach

◆ Inferred Rendering

Light Pre-pass Rendering의 확장
저해상도 G-Buffer 및 Light Accumulation Buffer 사용
DSF(Discontinuity Sensitive Filtering)로 Upsampling

- Pros
  - 많은 수의 Light에 대응 가능
  - MSAA 대응 가능
  - Material 제약이 없음
  - G-Buffer가 가벼움
  - Transparent 처리 가능
    - 제한된 수의 Layer(불투명 포함 4)만 표현 가능
    - 점묘법이므로 품질이 떨어짐
  - Light Rendering이 가벼움
    - Screen Resolution과 별도로 운용 가능

Deferred Rendering Approach

◆ Inferred Rendering Cont'd

- Cons
  - 2nd Pass에 따르는 부담
  - Specular 연산이 부정확함
  - High Frequency 정보가 손실됨
  - DSF Filter가 매우 느림

Deferred Rendering Approach

◆ Our Work

Motivation

- 다양한 Material의 표현이 필요함
- Light Pre-pass의 2 pass는 부담
  - Skinning/Tessellation이 적용되면 부담이 더욱 증가

Hybrid Deferred Rendering

- Deferred Shading + Light Pre-pass Rendering
  - 대부분의 Object 들은 General Material (Terrain, 나무, 풀, 돌, …)
    - Deferred Shading을 적용
  - 일부의 Object 들은 Special Material (Skin, Hair, Glossy, …)
    - Light Pre-pass Rendering을 적용
- How?

Deferred Rendering Approach

◆ Hybrid Deferred Rendering

For each object
- G-Buffer에 Geometry 및 Material 정보를 Render
- G-Buffer에 Deferred ID를 기록 (e.g. General = 0, Special > 0)

For each light
- G-Buffer를 참조하여 Lighting 수행, Light Accumulation Buffer에 누적
  - Deferred ID가 0이면 Deferred Shading 수행
  - Deferred ID가 0보다 크면 Light Pre-pass 수행

Screen Space Pass to Accumulation Buffer
  - Deferred ID가 0이면 Ambient + Light Accumulation 결과 출력
  - Deferred ID가 0보다 크면 Ambient 출력

For each object (2^{nd} Pass) with Special Material
- Light Accumulation Buffer를 참조하여 Special Material Shading 완료

Deferred Rendering Approach

◆ Hybrid Deferred Rendering Cont'd

- Pros
  - 많은 수의 Light에 대응 가능
  - Material 제약이 없음
  - 부분적인 2nd Pass 처리
    - 대부분은 General Material, 일부의 Special Material
  - General Material은 Specular 연산이 정확함
    - General Material은 상대적으로 넓은 면적을 차지함
      - 여러 개의 Light가 중첩될 가능성 높음
- Cons
  - Transparent 처리 불가
  - MSAA 처리가 까다로움
  - G-Buffer가 무거움

Deferred Rendering Approach

◆ Hybrid Deferred Rendering Cont'd

- 3개의 Render Target을 사용
  - Light Map 또는 Emissive 적용 시 Accumulation Buffer 추가 사용
- 2개의 FP16 Buffer에 View-space Normal의 X/Y 저장
  - Spherical Coordinates를 적용하여 Packing 및 Z Reconstruct
- Native Depth Buffer 활용
  - 이후 활용을 위해 Linear Depth로 Resolve

G-Buffer	R8	G8	B8	A8
RT0	Diffuse R	Diffuse G	Diffuse B	Deferred ID
RT1	Normal X (FP16)		Normal Y (FP16)	
RT2	Specular R	Specular G	Specular B	Specular Power
RT3(Optional)	Accumulation R	Accumulation G	Accumulation B	Accumulation A
Depth/Stencil	Depth			Stencil

Hybrid Deferred Rendering G-Buffer Layout

Deferred Rendering Approach

- **Light Pre-pass VS Hybrid Deferred Rendering Performance**

Comparison Test

System 1

- Intel Core2 Quad Q9500
- Radeon HD5770 1G
- GeForce GTX460 1G

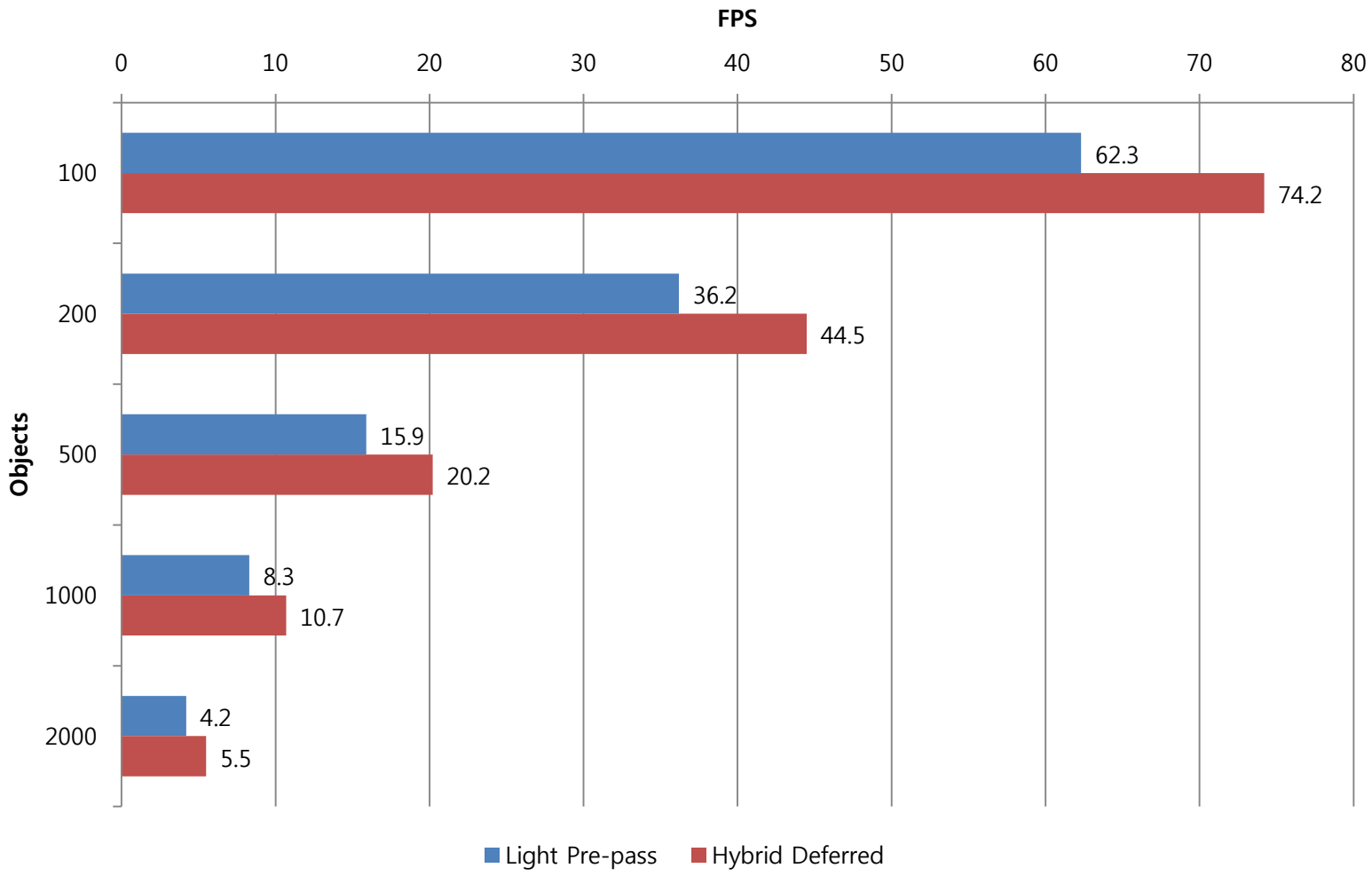
System 2

- Intel Core2 Duo E6750
- Radeon HD5850 1G

Test Scene

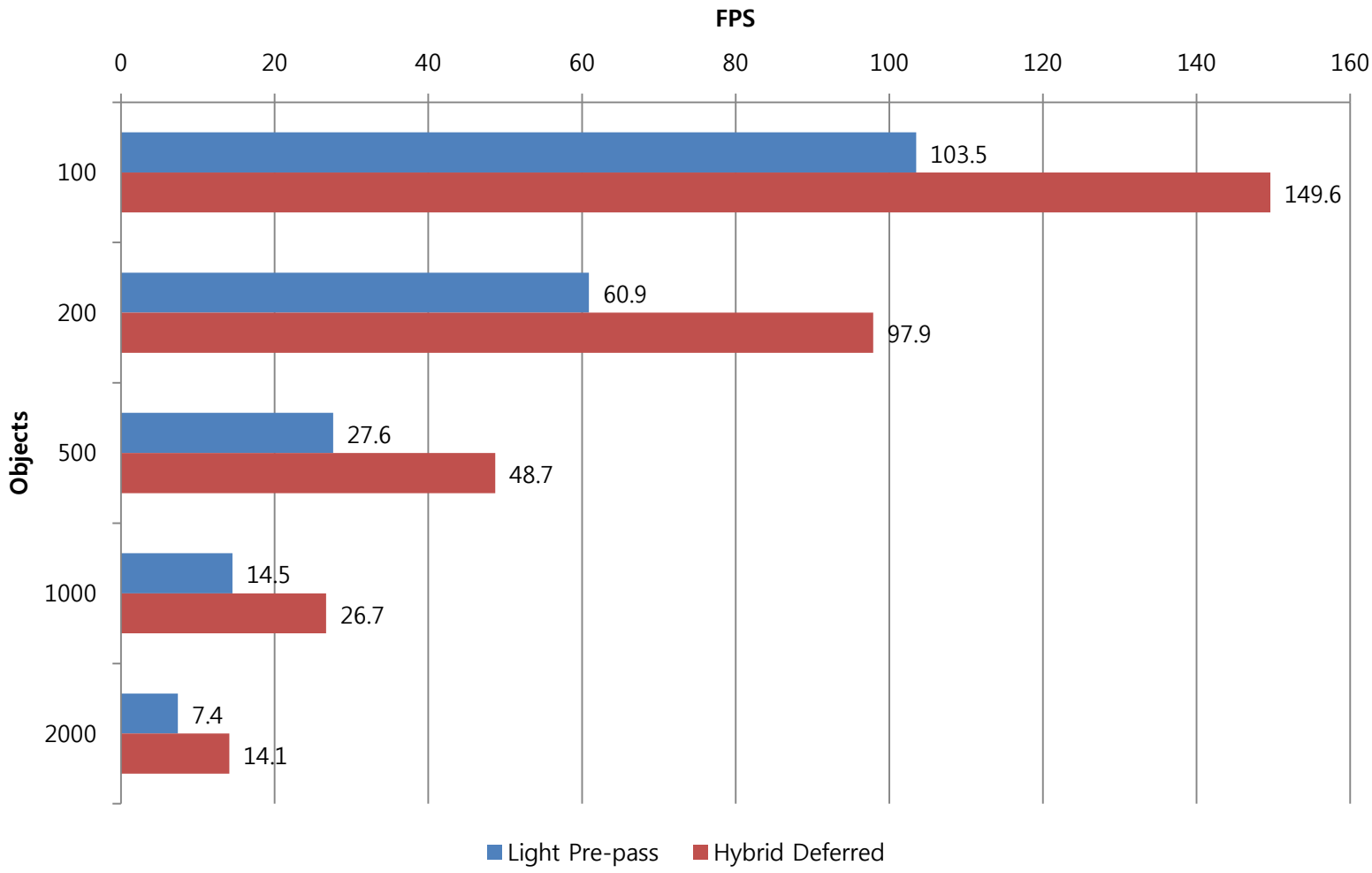
1024 X 768 @ 32BPP
64 Point Lights
1 Directional Light With/Without Shadow Map (1024)
Object With 1 General and 1 Special Material
100 / 200 / 500 / 1000 / 2000 Objects

Deferred Rendering Approach



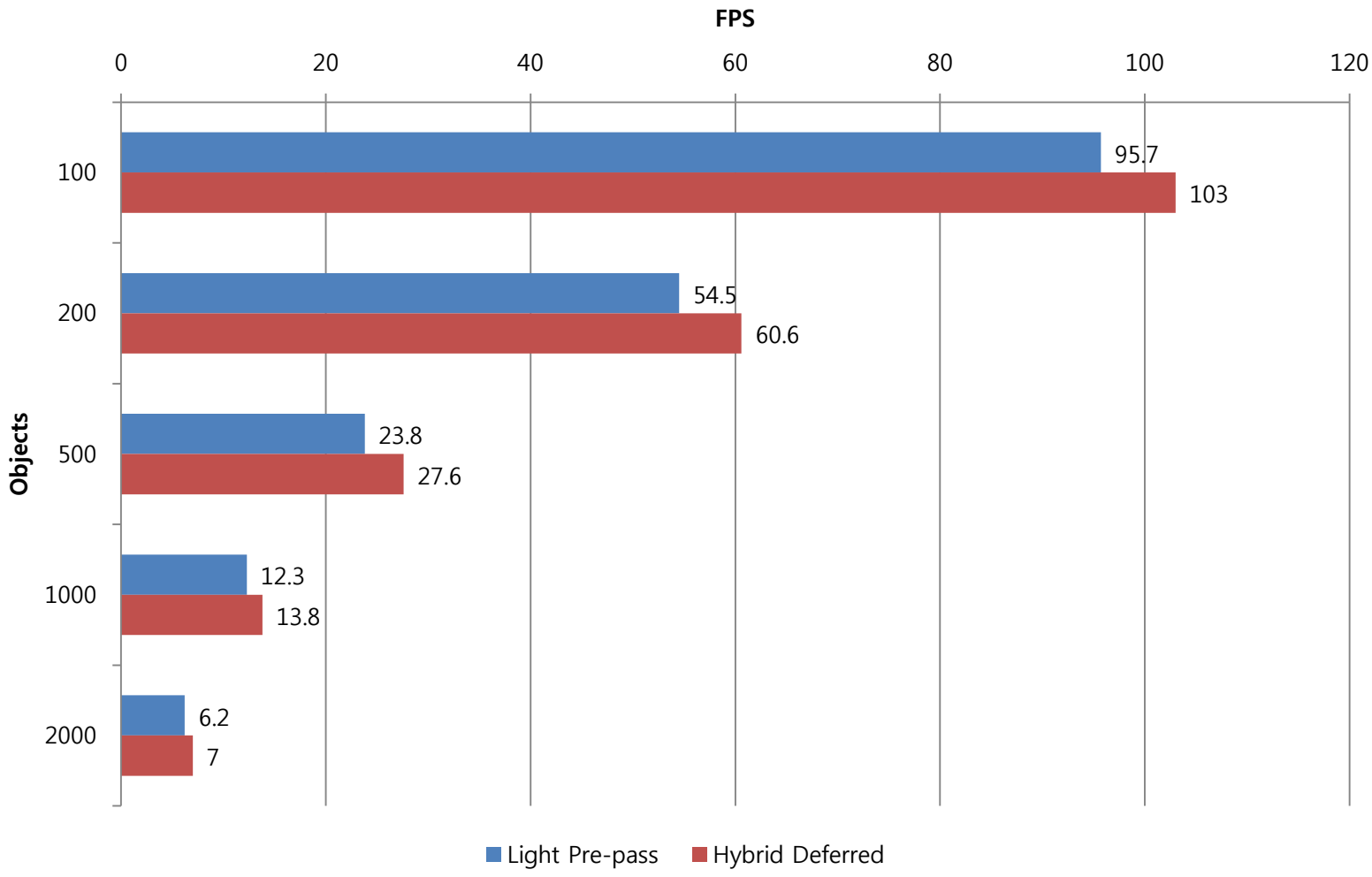
Radeon HD5770 With Shadow Map

Deferred Rendering Approach



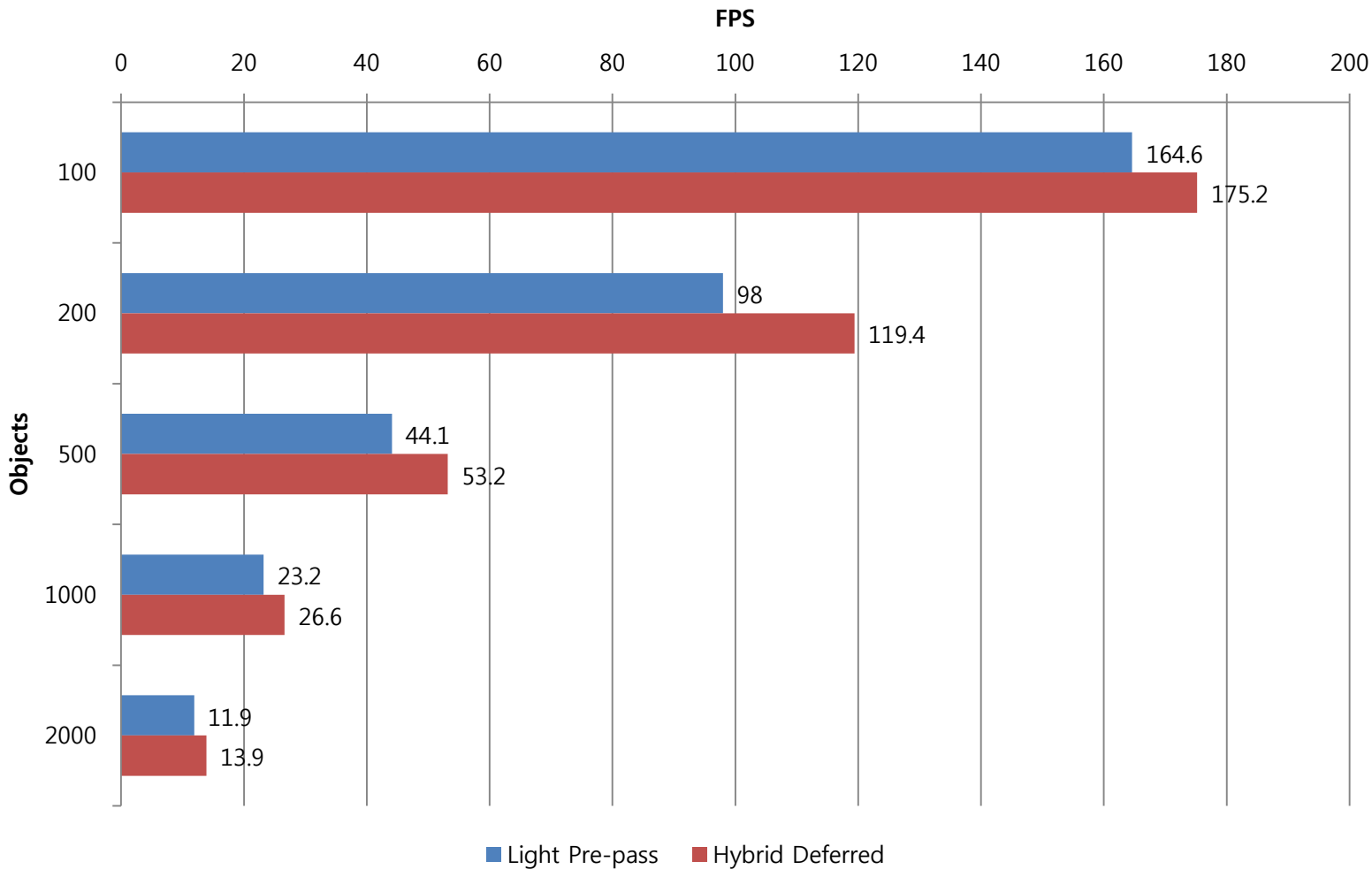
Radeon HD5770 Without Shadow Map

Deferred Rendering Approach



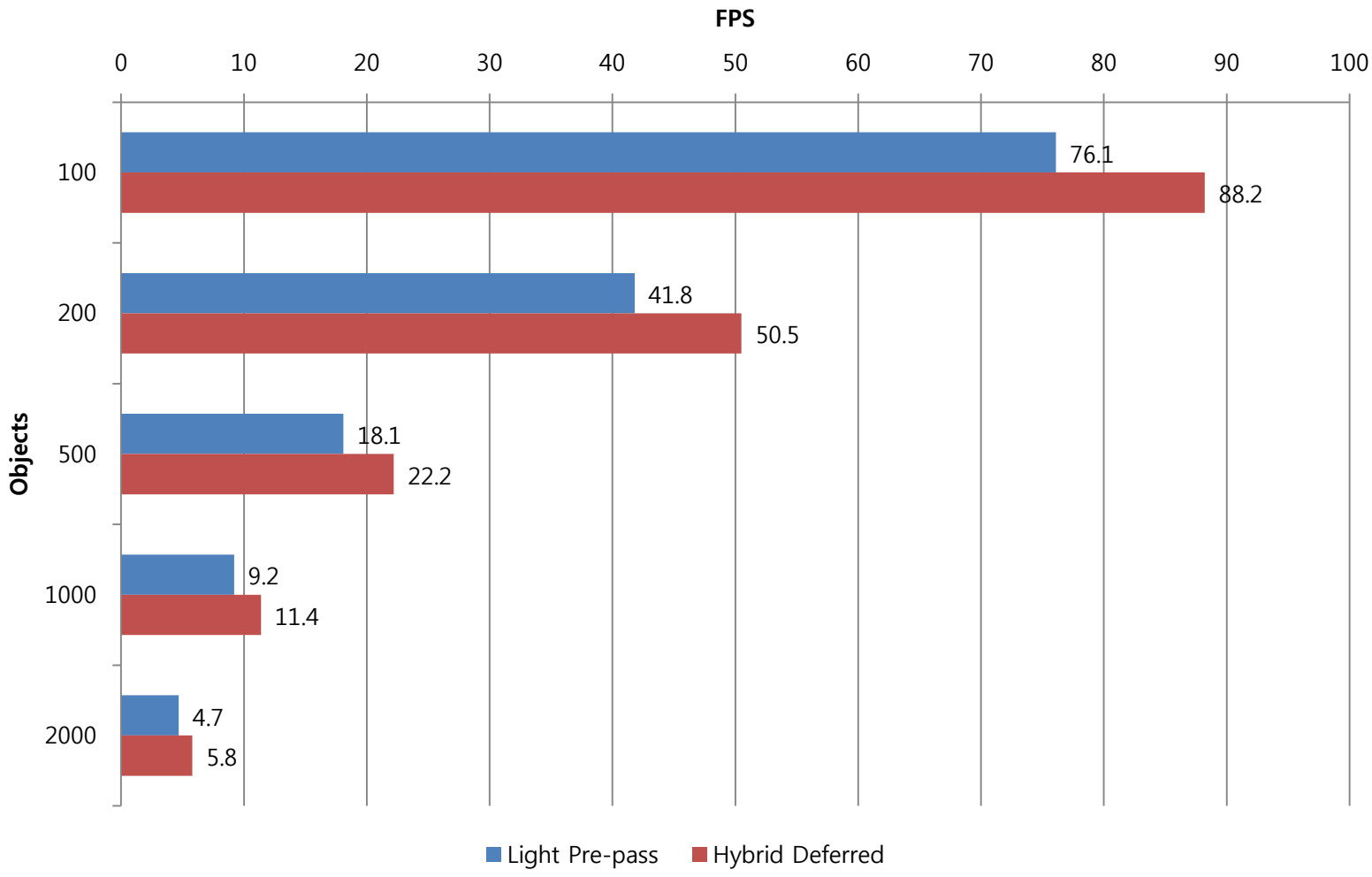
GeForce GTX460 With Shadow Map

Deferred Rendering Approach



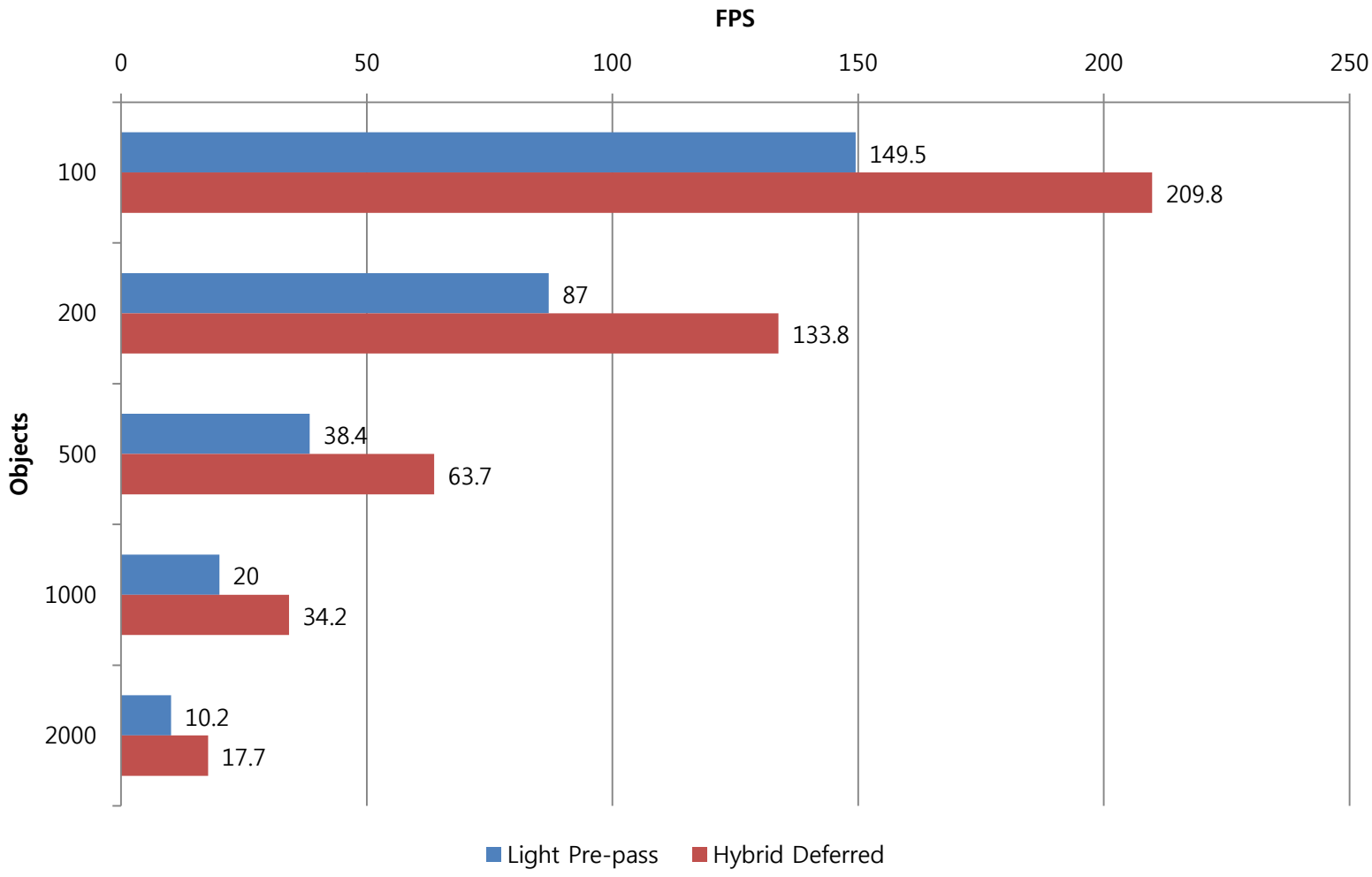
GeForce GTX460 Without Shadow Map

Deferred Rendering Approach



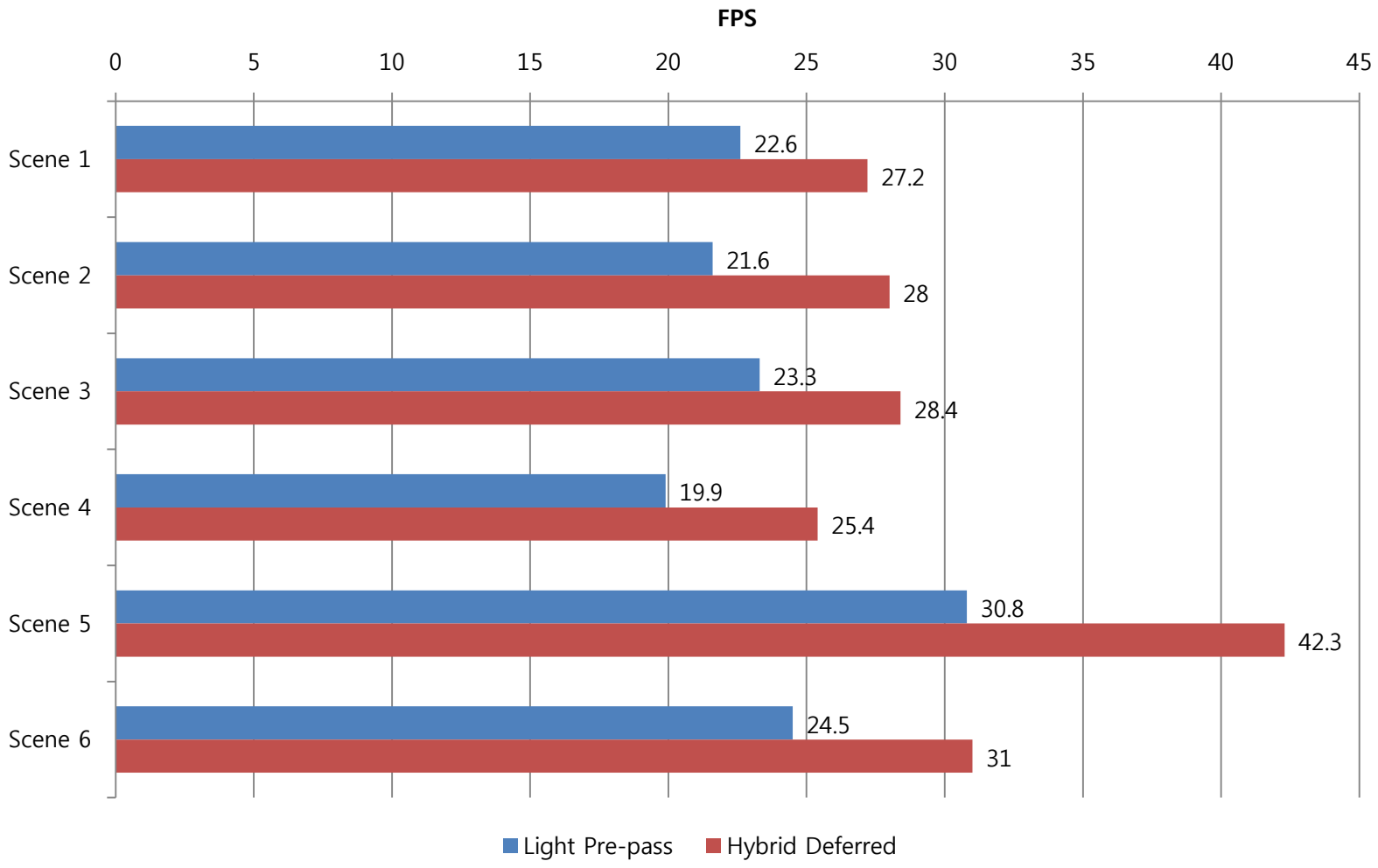
Radeon HD5850 With Shadow Map

Deferred Rendering Approach



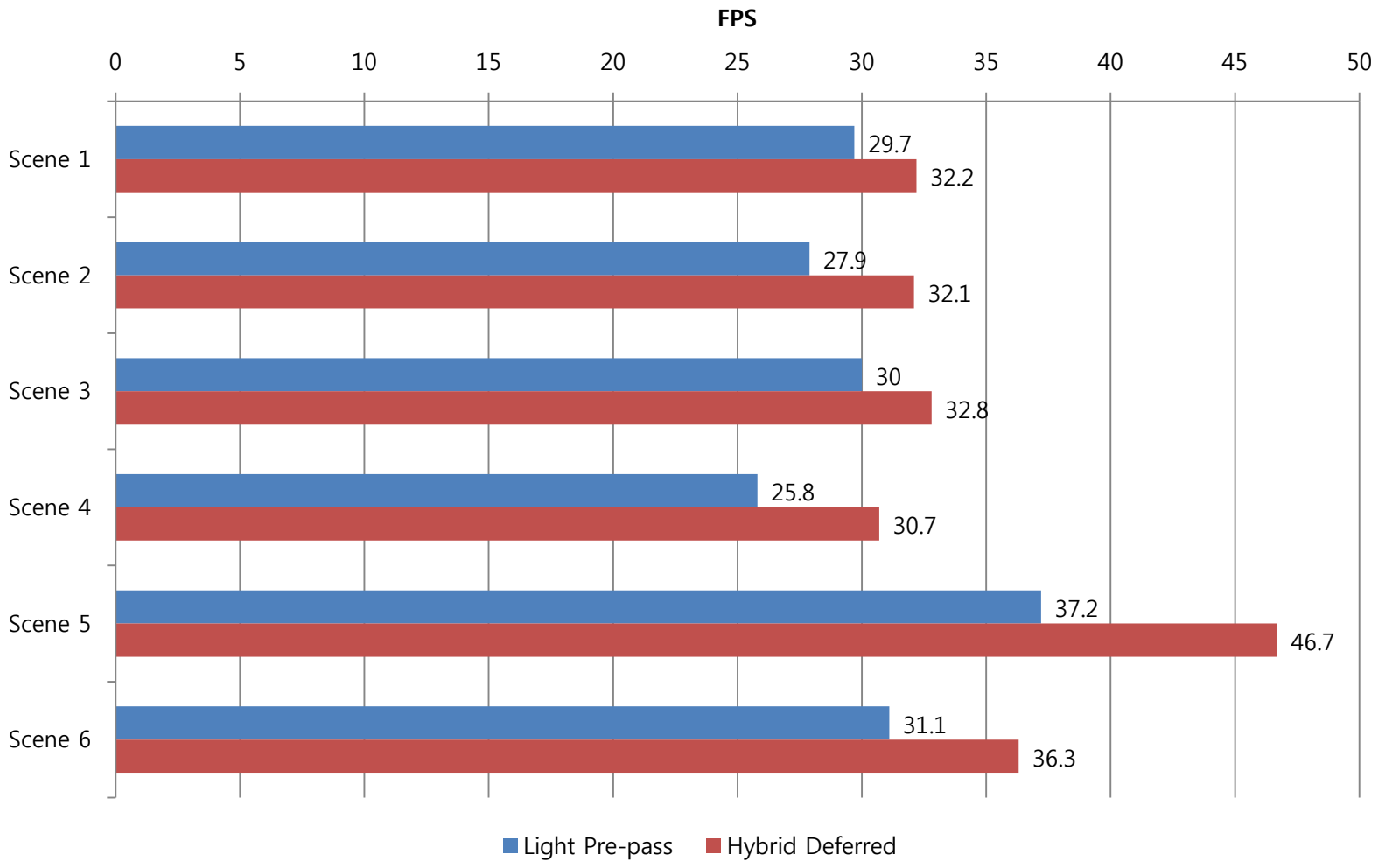
Radeon HD5850 Without Shadow Map

Deferred Rendering Approach



Radeon HD5770 Practical Map

Deferred Rendering Approach

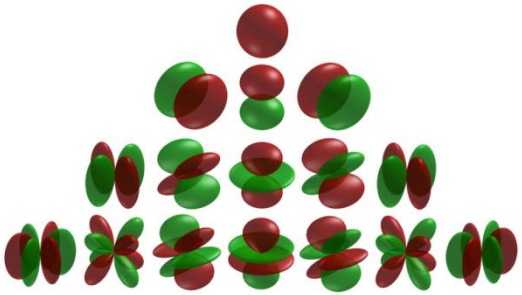


GeForce GTX460 Practical Map

Light Propagation Volumes

◆ Real-time Global Illumination

- Anton Kaplanyan에 의해 제안 (Crytek) [2009]
- CryENGINE 3(Crysis 2)에 적용됨
- Reflective Shadow Map에 기반
  - Color(Diffuse * Albedo), Normal, Depth로 구성
  - VPL(Virtual Point Light)의 개념 적용
- Spherical Harmonics에 의한 전파
  - 2nd Band까지 적용 (4 Coefficients)

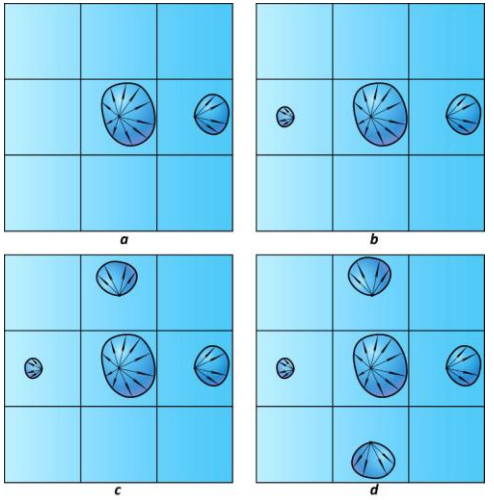


Spherical Harmonics

Light Propagation Volumes

◆ Overview

- Reflective Shadow Map 생성
  - Color, Normal, Depth
- Radiance Injection
  - VPL이 속하는 LPV에 SH Coefficients로 변경 및 누적
- Radiance Propagation
  - 인접 Cell로의 Radiance 전파
- Scene Lighting
  - G-Buffer를 참조하여 Render

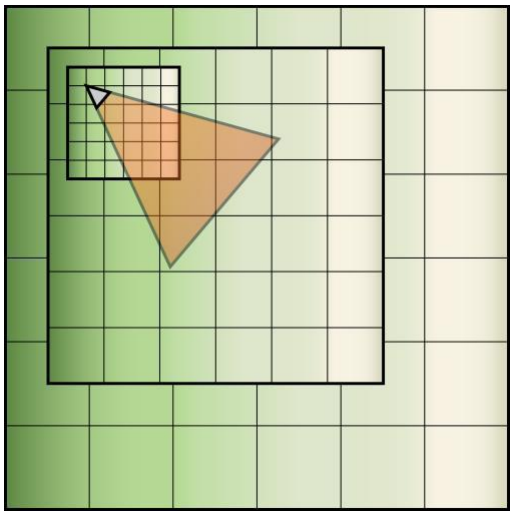


Radiance Propagation

Light Propagation Volumes

◆ Cascaded Light Propagation Volumes

- Cascaded Shadow Map과 유사
- 하나의 LPV로는 넓은 영역에 적용이 어려움
  - 3차원 Grid이므로 $O(N^3)$의 복잡도
  - 근경에는 세밀한 LPV를 사용
  - 원경에는 간격이 큰 LPV를 사용
- 현재 2개의 Cascade를 적용
  - Cascade간 매끄럽지 못한 연결이 다소 발생
    - LPV Propagation에서의 오차인지 확인 중
  - RSM(Reflective Shadow Map)을 Cascade 마다 생성해야 함



Cascaded LPV

Light Propagation Volumes

Without LPV

Light Propagation Volumes

With LPV

Light Propagation Volumes

With LPV And SSAO

Light Propagation Volumes

Without LPV Light Only

Light Propagation Volumes

With LPV Light Only

Light Propagation Volumes

With LPV And SSAO Light Only

Light Propagation Volumes

Without LPV

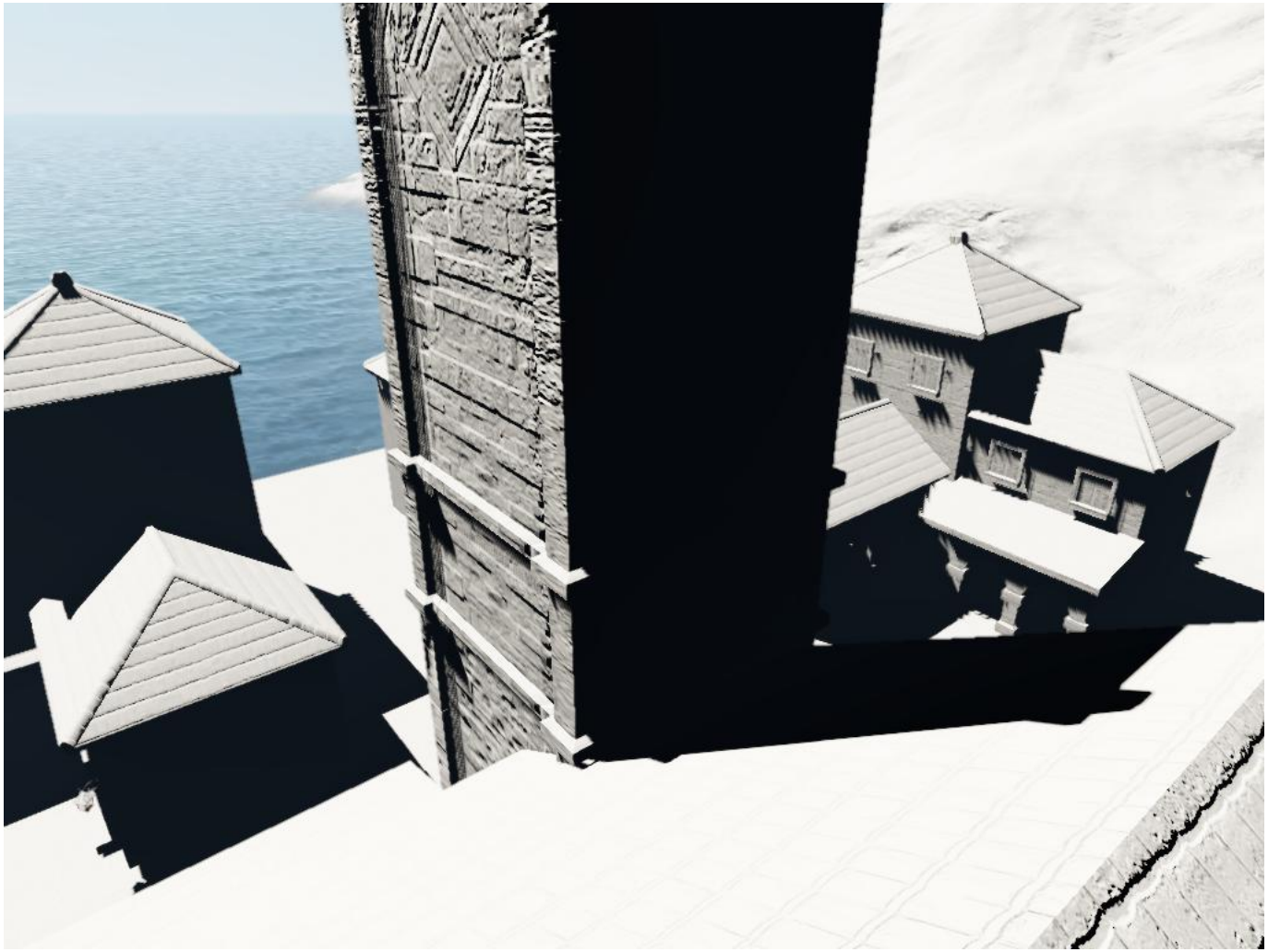
Light Propagation Volumes

With LPV

Light Propagation Volumes

With LPV And SSAO

Light Propagation Volumes



Without LPV Light Only

Light Propagation Volumes

With LPV Light Only

Light Propagation Volumes

With LPV Light Only

Light Propagation Volumes

LPV 0 Iteration (Injection Only)

Light Propagation Volumes

LPV 2 Iterations

Light Propagation Volumes

LPV 4 Iterations

Light Propagation Volumes

LPV 6 Iterations

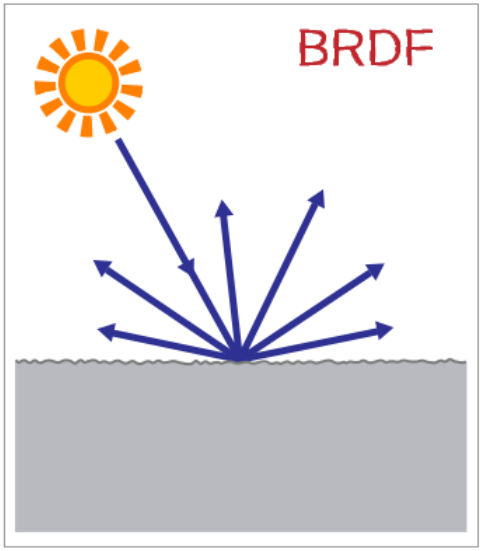
Light Propagation Volumes

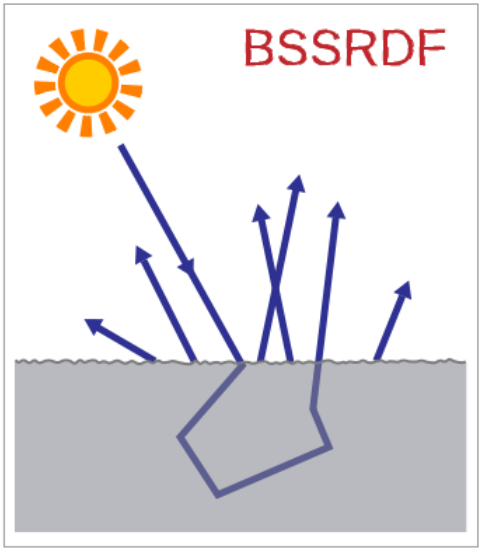
LPV 8 Iterations

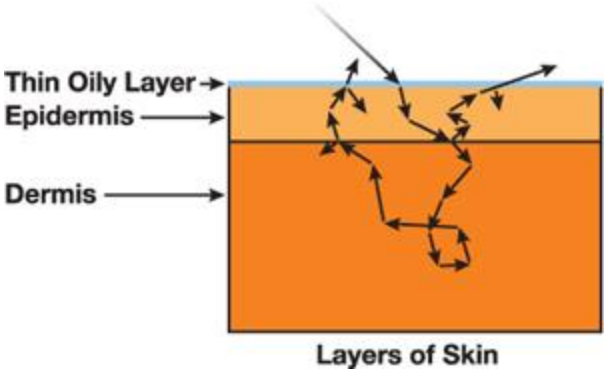
Screen Space Sub-Surface Scattering

◆ Why?

- 사람의 눈은 피부를 구별하는데 익숙함
- BRDF(Bidirectional Radiance Distribution Function)으로는 피하 산란을 표현할 수 없음
- BSSRDF(Bidirectional Surface Scattering Reflectance Distribution Function)
  - Subsurface Scattering







Screen Space Sub-Surface Scattering

◆ Previous Work

Texture-space Diffusion

- The Matrix Reloaded (Borshukov and Lewis [2003]), Eugene d'Eon et al. [2007]
- 모든 SSS 적용 모델 마다 개별 처리해야 함
  - Real-time에 적용하기에 적합하지 않음
- Convolution Kernel 왜곡으로 인한 Artifact 발생
  - Texture Space로 인한 한계

Screen Space SSS

- SSS가 적용될 모델들을 Masking 후 Post Process로 처리
- James Jimenez et al. [2009]
  - Screen Space Blur로 인한 실루엣 침범 발생
- Morten S. Mikkelsen (Naughty Dog Inc.) [2010]
  - Cross Birateral Filter 적용하여 실루엣 문제 해결

Screen Space Sub-Surface Scattering

◆ Our Work

Post-Scatter Texturing

- Hybrid Deferred Rendering의 Light Accumulation Buffer를 활용
- Pre-Scatter Texturing 적용 시 Diffuse Map의 Detail이 뭉개짐

Gaussian Convolution 중첩을 통한 Blur Sample 재활용

- 6 Pass Blur (X/Y * 3), 8 Pass for high quality

$$(f \otimes g) \otimes h = f \otimes (g \otimes h)$$

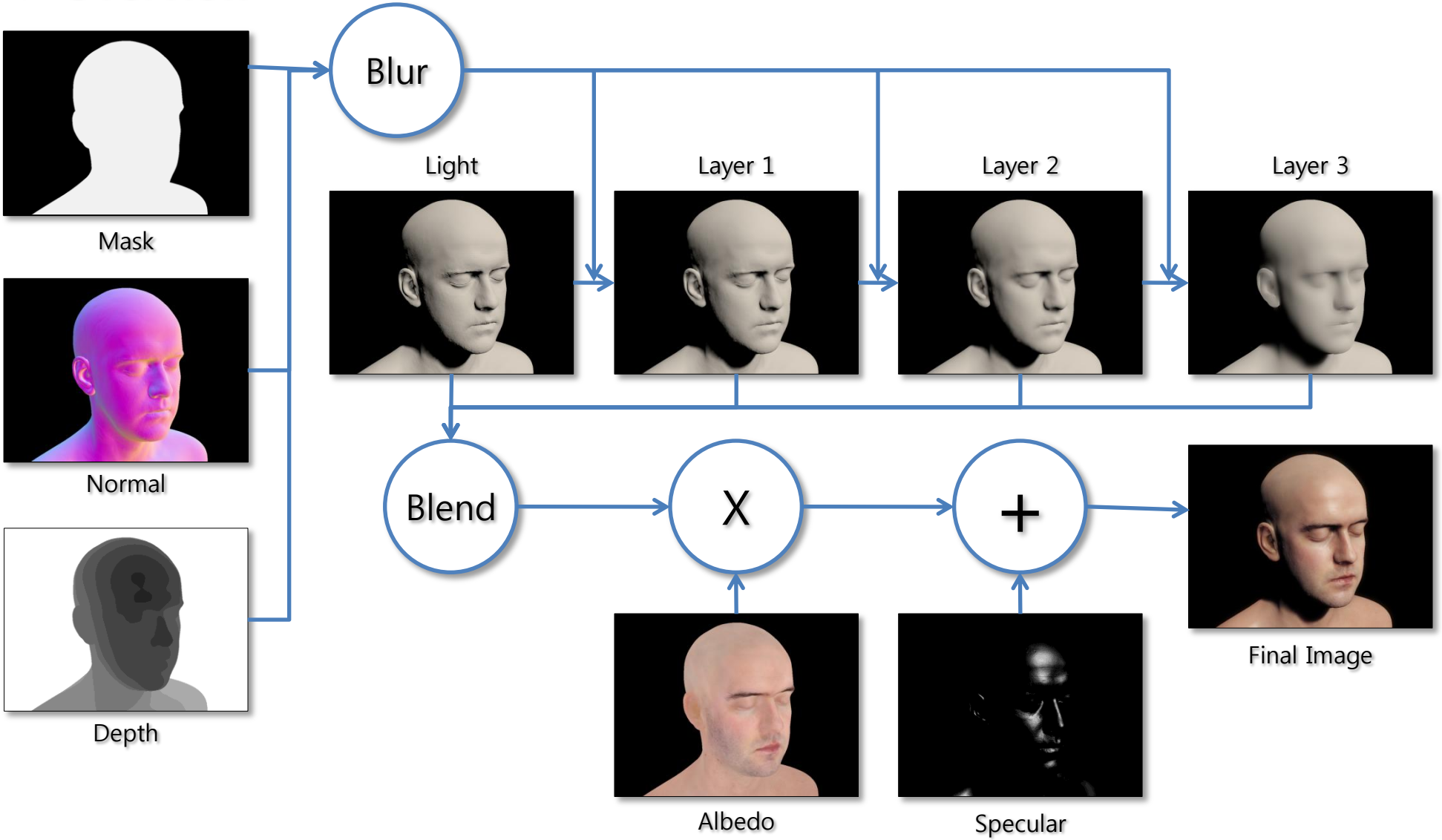
$$\sigma_{f \otimes g} = \sqrt{\sigma_f^2 + \sigma_g^2}$$

Stencil Buffer를 이용한 최적화

- Skin Mask Rendering시 Stencil Buffer에 Reference Value 기록
- Blur Pass에서 Stencil Test 활성화

Screen Space Sub-Surface Scattering

◆ Overview



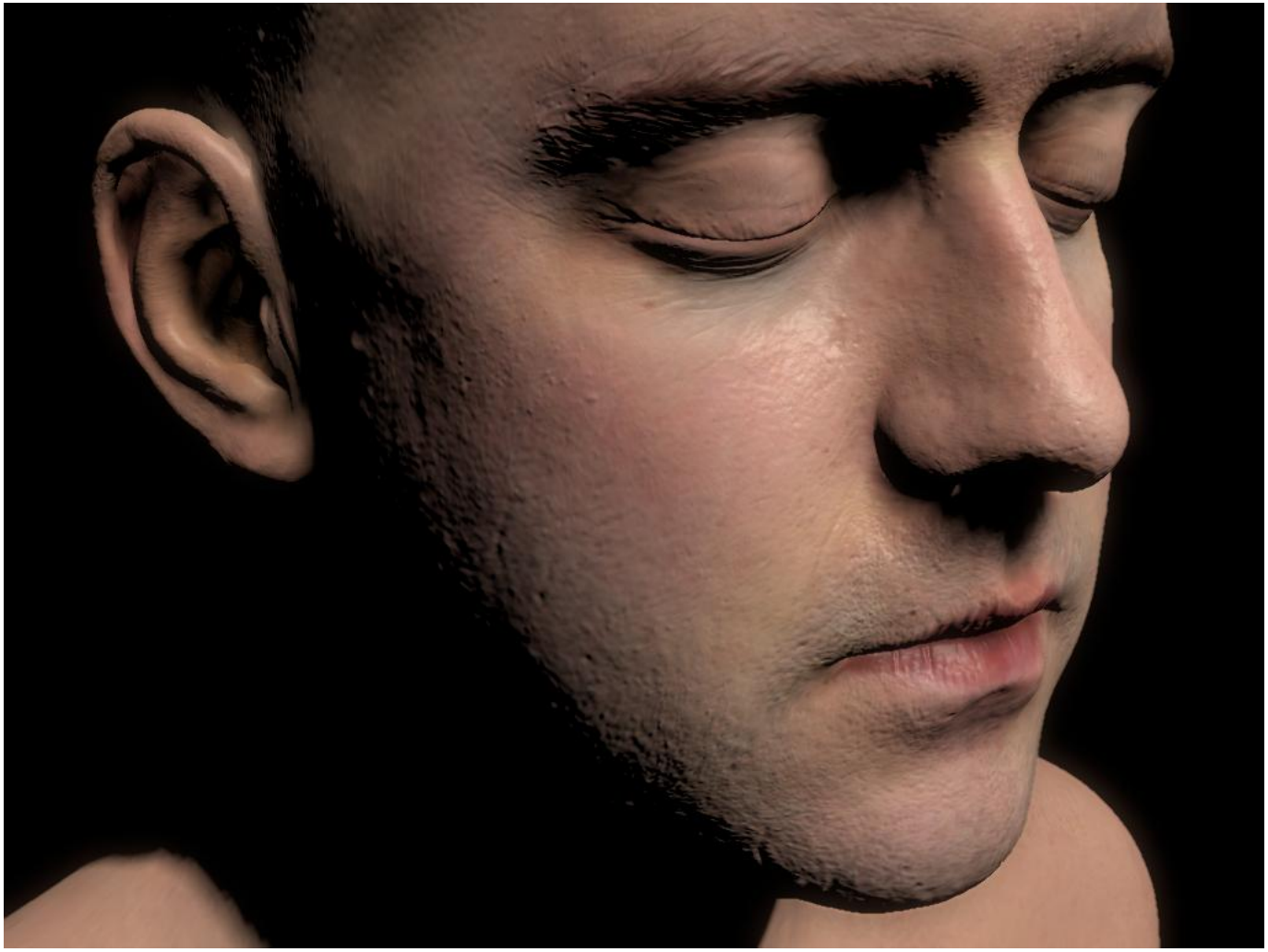
Screen Space Sub-Surface Scattering

Without SSSSS

Screen Space Sub-Surface Scattering

With SSSSS

Screen Space Sub-Surface Scattering



Without SSSSS

Screen Space Sub-Surface Scattering

With SSSSS

Screen Space Sub-Surface Scattering

Without SSSSS Light Only

Screen Space Sub-Surface Scattering

With SSSSS Light Only

Screen Space Sub-Surface Scattering

- **3D head scan from Infinite-Realities**
  - http://www.ir-ltd.net/infinite-3d-head-scan-released
  - Scan by Lee Perry-Smith, thanks very much
  - 17684 polygons
  - Diffuse/Normal/Displacement Map size 4096x4096
  - Tessellation 적용 (PN Triangle)

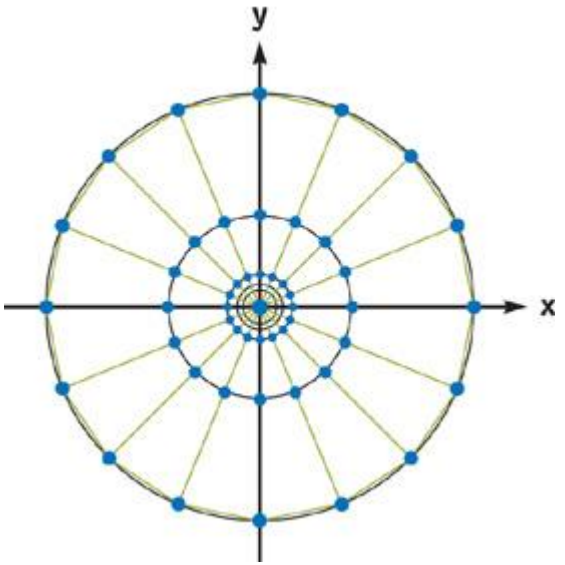
Water Rendering

- **Used Techniques**
  - Radial Grid
  - Vertex Displacement
  - Reflection
  - Refraction
  - Refraction Caustics
  - Scattering

Water Rendering

◆ Radial Grid

- Camera 중심의 동심원 형태의 Grid를 생성
- Camera의 X, Y좌표를 항상 따라 다님
  - Frustum과 수평일 경우 정확한 LOD 표현
  - Frustum과 수직일 경우에도 무난함
- GPU Friendly
  - 생성 후 Vertex 조작이 필요치 않음
- 구현하기 쉬움



Water Rendering

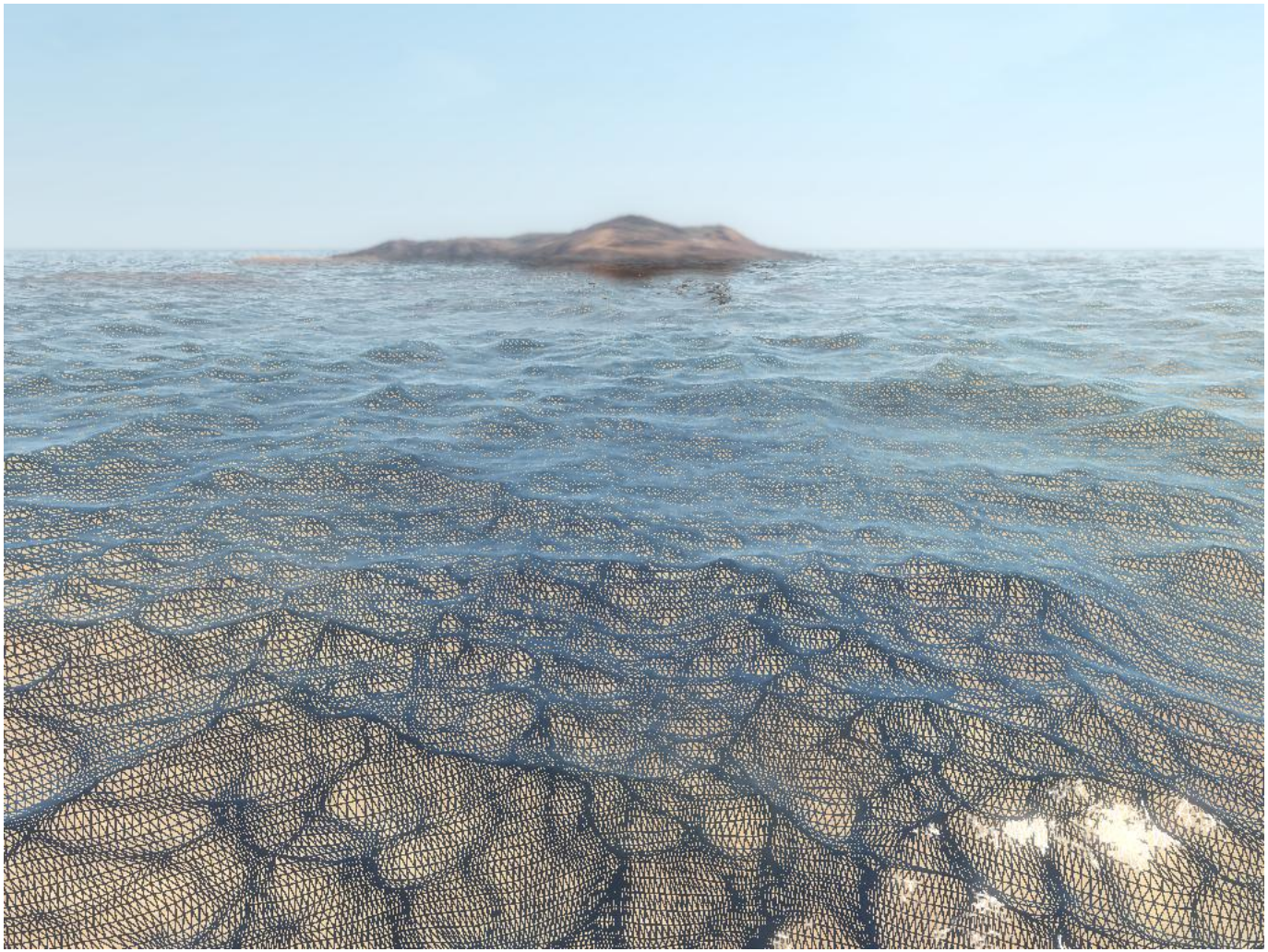
Radial Grid

Water Rendering

◆ Vertex Displacement

- Vertex Shader에서 Vertex Texture Fetch를 사용
- 4~5 번 정도의 Sampling으로 좋은 효과를 얻을 수 있음
- Displacement Map - Z축(수직) Displacement
- Normal Map – XY축(수평) Displacement

Water Rendering



Radial Grid With Vertex Displacement

Water Rendering

◆ Reflection

- Mirror Rendering으로 Reflection Image 생성
- Normal Map에 기반한 Disturbance 적용
- Fresnel Term
  - Schlick's approximation: $\mathbf{R}(\theta) = \mathbf{R0} + (1 - \mathbf{R0})(1 - \cos\ \theta)5$
  - Reflection과 Refraction을 혼합

    outColor = lerp(Refraction, Reflection, fresnel_term)

Water Rendering

Uniform Disturbance Reflection

Water Rendering

Real World Reflection

Water Rendering

◆ Reflection Cont'd

- Vertical Blur
  - Anisotropic Reflection을 표현
- Mirror Depth
  - Disturbance 적용 시 Mirror Space Depth를 반영

Water Rendering

Reflection With Vertical Blur And Mirror Depth Consideration

Water Rendering

◆ Refraction

- Dark water color로의 Exponential falloff 통해 Opacity 표현
- 수면 아래만 Rendering 하여 Refraction Image 생성
  - 추가적인 부담
  - 일반적으로 Low Resolution을 사용함
- Accumulation Buffer의 Rendering 결과를 재활용
  - 수면 위에 있는 물체 주변에 Artifact 발생
  - Alpha Blending을 이용하여 증상을 완화

    alpha = lerp(1 - opacity, 1, refraction_depth < water_depth)

    outColor *= 1 / alpha

Water Rendering

Refraction Artifact

Water Rendering

Refraction Artifact Removed By Alpha Blending

Water Rendering

◆ Refraction Caustics

- Normal Map에 간단한 연산을 통해 Pattern Texture 참조

  ```
  float2 refraction = refract(light_dir, normal, refraction_index) * 0.5 + 0.5
  caustics = tex2D(Pattern Texture, refraction)
  ```

- Pattern Texture에 의해 Caustics의 형태가 결정됨

Water Rendering

Without Refraction Caustics

Water Rendering

With Refraction Caustics

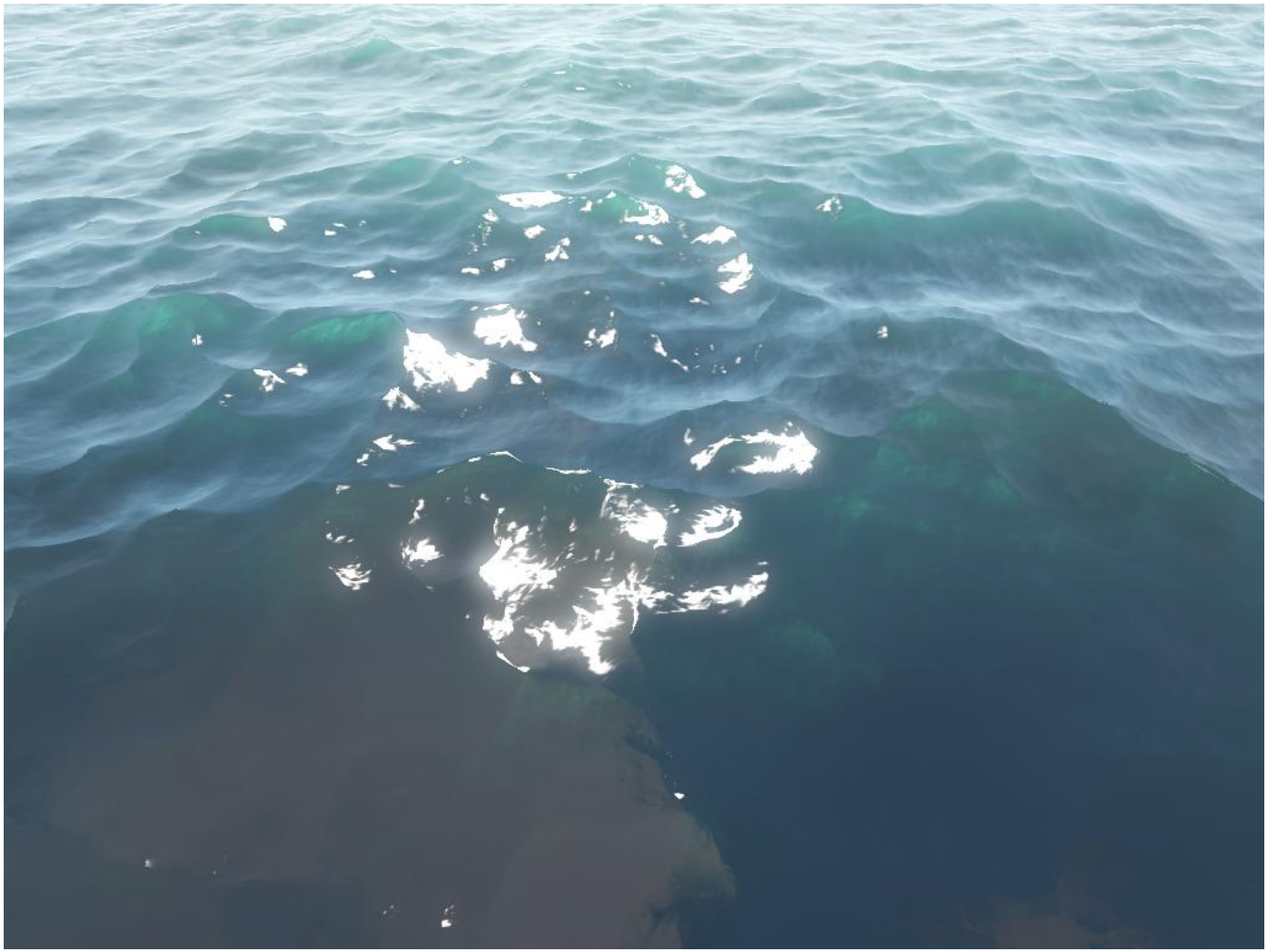
Water Rendering

- **Scattering**
  - Wave Height, Eye Direction, Light Direction에 기반한 Heuristic
  - Vertex Displacement와 어우러져 Dynamic한 연출
  - nVidia Direct3D 11 SDK 참고

Water Rendering

Without Scattering

Water Rendering



With Scattering

Water Rendering

References

◆ Deferred Rendering Approach

Hargreaves, Shawn, and Mark Harris. 2004. "6800 Leagues Under the Sea: Deferred Shading."

Oles Shishkovtsov. “Deferred Shading in S.T.A.L.K.E.R.”, GPU Gems 2. 2005.

Michal Valient. Deferred Rendering in Killzone 2. 2007.

Wolfgang Engel. “The Light Pre-Pass Renderer”, ShaderX7. 2008.

Scott Kircher, Alan Lawrance. Inferred Lighting: Fast dynamic lighting and shadows for opaque and translucent objects. SIGGRAPH 2009.

Martin Mittring. “A bit more Deferred” - CryEngine 3. Triangle Game Conference 2009.

References

◆ Light Propagation Volumes

Carsten Dachsbacher, Marc Stamminger. Reflective Shadow Maps. University of Erlangen-Nuremberg. 2005.

Anton Kaplanyan. Light Propagation Volumes in CryEngine 3. SIGGRAPH Course, 2009.

Anton Kaplanyan and Carsten Dachsbacher. Cascaded Light Propagation Volumes for Real-Time Indirect Illumination. 2010.

References

◆ Screen Space Sub-Surface Scattering

George Borshukov and J. P. Lewis. Realistic human face rendering for “The Matrix Reloaded". In SIGGRAPH '03: ACM SIGGRAPH 2003 Sketches & Applications.

Craig Donner and Henrik Wann Jensen. Light diffusion in multi-layered translucent materials. ACM Trans. Graph. 24,3,1032-1039, 2005.

Eugene d'Eon and David Luebke. GPU Gems 3, chapter 14, pages 293-347. Addison-Wesley Publishing, 2007.

Jorge Jimenez, Veronica Sundstedt, and Diego Gutierrez. Screen-space perceptual rendering of human skin. ACM Transactions on Applied Perception, 6(4), 2009.

Morten S. Mikkelsen. Skin Rendering by Pseudo-Separable Cross Bilateral Filtering. 2010.

References

◆ Water Rendering

Yuri Kryachko. GPU Gems 2, Chapter 18, Using Vertex Texture Displacement for Realistic Water Rendering.

Tim Tcheblokov. Island demo. nVidia Direct3D SDK 11.