광선 추적법 텍스쳐 매핑을 위한 MIP-Map 수준 선택 알고리즘 연구※

박우찬, 김동석
세종대학교 컴퓨터공학과 미디어프로세서 연구실
pwchan@sejong.ac.kr, dskim@rayman.sejong.ac.kr

An Algorithm of MIP-Map Level Selection for Ray-Traced Texture Mapping

Woo-Chan Park, Dong-Seok Kim
Media Processor Lab. ,Dept. of Computer Engineering, Sejong Univ.

요 약

본 논문은 광선 추적법 텍스쳐 매핑에서 MIP-Map 알고리즘 사용 시 텍스쳐 이미지들의 MIP-Map 수준을 선택하는 효과적인 방식을 제안한다. 이는 렌더링 시 물체와 교차하는 광선의 길이만을 사용하여 해당 물체의 텍스쳐 MIP-Map 수준을 선택하는 방법이다. 본 방식은 MIP-Map을 지원하지 않는 방식에 비하여 텍스쳐 알리아싱 면에서 우수하고 성능 저하는 미비하다.

ABSTRACT

This paper proposes an effective method to select MIP-Map level of texture images for ray-traced texture mapping. This MIP-Map level selection method requires only the total length of intersected ray. By supporting MIP-Map for texture mapping, we can reduce the texture aliasing effects, while our approach decreases rendering performance very slightly.

Keywords : 3D Graphics(3차원 그래픽스), Ray Tracing(광선 추적법), Texture Mapping (텍스쳐 매핑)

접수일자 : 2010년 06월 17일 심사완료 : 2010년 07월 05일
교신저자(Corresponding Author) : 박우찬
※ 본 연구는 2008년도 세종대학교 교내연구비 지원을 받아 수행된 연구 결과입니다.

1. 서 론

현재 거의 모든 그래픽 프로세서는 스캔변환(scan conversion)을 기반으로 하는 래스터(rasterization) 방식을 채택하고 있다. 한편, 광선 추적 방식은 각 픽셀에 대해서 광선을 생성하여 이에 영향을 미치는 삼각형들을 역 추적하는 방식이기 때문에 전역 조명(global illumination) 효과가 가능하다. 이는 기존의 래스터 방식에서 구현하기 힘든 그림자(shadow)효과, 반사(reflection)효과, 굴절(refraction)효과, 투과(transparent)효과 등을 기본적으로 제공하며, 또한 콘텐츠 제작의 용이성 등의 다양한 장점을 가지고 있다[1,2]. 하지만 이는 방대한 연산량을 요구하기 때문에 오프라인 프로세싱 위주로만 이용되어 왔다.

반도체 기술의 발전으로 인하여 광선 추적을 실시간으로 처리하고자 하는 연구들이 생겨나고 있으며, 향후에는 광선 추적 방식이 래스터 방식을 대체할 것으로 예상하고 있다[1]. 최근 Intel의 Larrabee와 Nvidia의 GPU 등 매니코어(many-core)의 등장으로 인하여 실시간 광선 추적에 대한 연구가 활발히 진행되고 있다[3,4].

고품질의 그래픽 영상을 제공하기 위하여 텍스쳐 매핑(Texture Mapping)은 기본적으로 지원해야 되는 기능이다. 텍스쳐 매핑을 효과적으로 지원하기 위하여 원본 텍스쳐 이미지를 단계별로 줄여서 계층화한 MIP-Map을 기본적으로 사용한다. 텍스쳐 매핑시 MIP-Map 수준을 적절하게 선택하여 사용함으로써 알리아싱(aliasing) 효과 및 텍스쳐 데이터의 참조량을 줄일 수 있다. 근래 들어 화질 면에서 우수한 이방형 필터링 (anisotropic filtering)도 MIP-Map에 기반으로 구현된다. [5]에서는 텍스쳐 매핑시 MIP-Map 수준을 결정하는 다양한 방식이 소개되어 있며, 이는 OpenGL에서 사용되는 래스터 방식에 기반하고 있다. MIP-Map 수준의 선택에는 기본적으로 텍스쳐 공간과 픽셀 공간과의 비율이 사용된다.

광선 추적 방식에 기반 한 것으로써 [6,7,8]에서는 광선 변화량 (ray differential)을 구함으로써 MIP-Map 수준을 결정한다. 하지만, 이는 계산량이 너무 많다는 단점이 있다. 이들 방식은 주로 고화질 3차원 애니메이션 제작 같은 오프라인 렌더링에 주로 사용되었다[9]. 현재까지 실시간 광선 추적을 위한 매니코어 방식 및 전용 하드웨어 방식에서 MIP-Map을 지원한 사례는 발견할 수 없었다.

본 연구에서는 광선 추적 방식에 기반한 텍스쳐 매핑시 MIP-Map 수준을 효과적으로 선택하는 알고리즘을 제안한다. 이는 먼저 전처리 시 각 프리미티브(primitive)에 대하여 시점과 해당 프리미티브의 MIP-Map 수준 0과의 거리를 계산한다. 렌더링 시 광선과 교차한 물체에 대하여 전처리 시 계산된 MIP-Map 수준 0과의 거리 값과 수행 중(on-the-fly)에 계산된 전체 광선의 길이 값만을 사용하여 해당 물체의 텍스쳐 MIP-Map 수준을 선택한다. 이는 텍스쳐 축소 비율에 대한 계산 기준을 면적에 기반하여 수행하는 것으로써, 기존의 광선 변화량 값을 이용한 방식에 비하여 정확도는 다소 떨어지지만 연산량을 대폭 줄임으로 인하여 빠른 처리가 가능하다. 제안하는 알고리즘은 매니코어[3,4]와 전용 하드웨어[10,11]에서 광선 추적을 가속하는 다양한 방식에 매우 유용하게 적용될 수 있을 것이다.

본 논문의 구성은 다음과 같다. 제 2장에서는 텍스쳐 매핑 소개 및 관련 연구에 대해 기술한다. 제 3장에서는 제안한 MIP-Map 수준 선택 알고리즘에 대해 기술한다. 제 4장에서는 실험 결과 및 분석에 대해 기술한다. 제 5장에서는 결론 및 향후 연구에 대해 논한다.

2. 관련 연구 및 배경

2.1 일반적인 텍스쳐 매핑

텍스쳐 매핑은 컴퓨터 그래픽스 분야에서 가상의 3차원 물체의 표면에 세부적인 질감의 묘사를

하거나 색을 칠해 마치 실제의 물체처럼 느껴지게 끔 그 세부묘사를 하는 것이다. 텍스쳐 매핑에 대한 알리아싱 효과를 줄이기 위한 방법으로 MIP-Map을 기본적으로 사용 한다[12]. 이는 원본 텍스쳐 이미지를 단계별로 줄여서 계층화한 것이다. 즉, 원본 텍스쳐 이미지를 수준 0로 지정하고 상위 수준의 텍스쳐 이미지는 하위 수준의 텍스쳐 이미지를 두개의 축에 대하여 1/2씩 축소하여 생성한다. 일반적으로 MIP-Map은 렌더링 이전에 필터링되어 생성된다.

렌더링 시 MIP-Map 수준을 선택하는데 있어 많은 알고리즘들이 있지만 가장 많이 쓰이고 있는 해당 픽셀의 변화량과 이에 따른 텍셀(texel: 텍스쳐 이미지의 픽셀)의 변화량에 대한 비율을 이용하여 수준을 정한다. 이를 텍스쳐 축소(texture minification) 비율이라고 한다. 이에 대하여 다양한 방식이 [5]에 소개되었다.

MIP-Map 수준 선택시 가장 많이 사용되는 방식은 텍스쳐 공간에서의 장축에 대하여 픽셀과 텍셀의 변화량의 비율을 이용하는 것이다. 이에 대하여 제시된 수식은 다음과 같다[7].

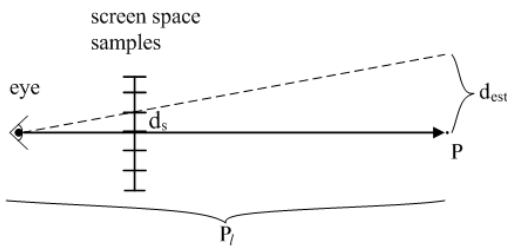
$$lod = \log_2(\max(|du|,|dv|)) \qquad \text{(식 1)}$$

여기서, (du, dv)는 해당 픽셀이 스크린 공간에서 텍스쳐 공간에 매핑 되었을 때 텍스쳐 좌표계 (u, v)에 대한 증분 벡터 값이다.

[5]에서는 이외에 텍스쳐 공간과 픽셀 공간과의 면적 비율을 사용하는 경우인 픽셀 클리핑(*Pixel Clipping*을을 언급하였다. 하지만, 면적의 비율에 기반한 방식은 텍스쳐의 변화량이 얇은 사변형 형태인 경우 알리아싱이 발생할 수 있다고 [12]에서는 언급되었다.

2.2 광선 추적 방식에 대한 MIP-Map 수준 선택 알고리즘

고화질 3차원 애니메이션 제작에 사용되는 RenderMan 같은 오프라인(off-line) 렌더링에 적용된 방식으로써 [6,7]에서는 광선 변화량을 구함으로써 MIP-Map 수준을 결정한다. 이들은 각각의 광선과 이웃하는 광선들과의 변화량을 계산한다. 광선들이 반사나 굴절로 인하여 전파됨에 따라서 이를 지속적으로 계산하고 유지한다. 최종적으로 계산된 광선 변화량을 기반으로 텍스쳐의 변화량 값을 계산한 후 (식 1)을 이용하여 MIP-Map 수준을 결정한다.



[그림 1] 이미지 평면에 투영에 기반의 기존 방법

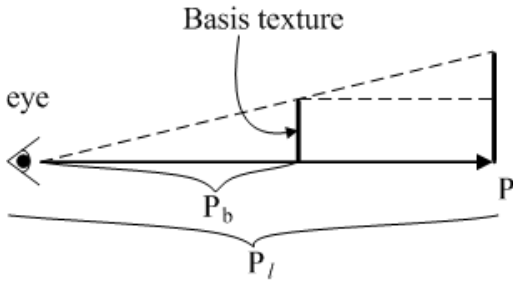
[8,9]은 곡면에서 광선이 반사와 굴절이 되면서 변화하는 광선의 변화량을 구함으로써, 곡면에서도 정확한 MIP-Map 수준 선택을 할 수 있는 방법을 제시하였다. 하지만, 이 방법은 계산량을 많이 필요로 하기 때문에 실시간 처리에는 적합하지 않다.

[6]에서는 폴리곤 렌더링에서 사용된 방식과 유사하며 텍스쳐의 좌표를 이미지 평면 (image plane)에 투영에 기반한 방법을 제시하였다. [그림 1]에서 제시된 바와 같이 먼저 광선에 길이 $\mathbf{P}_l$에 대하여 스크린 공간 샘플 대비 월드 공간 길이인 $\mathbf{d}_{est}$를 구한다. 이 값을 기반으로 텍스쳐 좌표인 u, v에 대하여 평면의 편미분 값을 계산한 후 텍스쳐의 변화량인 du와 dv를 구한다. 이 값을 가지고 (식 1)에 대입하여 MIP-Map 수준을 결정한다.

3. MIP-Map 수준 선택 알고리즘

3.1 제안하는 알고리즘

제안하는 알고리즘에 대한 기본 아이디어는 [그림 2]에 제시되어 있다. 먼저, 해당 삼각형의 텍스쳐에 대하여 MIP-Map 수준 0인 기준 텍스쳐(basis texture)가 되는 시점에서의 거리 P_b를 계산한다. 이는 픽셀의 크기와 텍셀의 크기가 1:1이 되는 부분을 의미하며, 시점과의 상대적인 거리기 때문에 시점의 위치와는 무관하다. 이는 해당 삼각형의 정점의 정보가 변하지 않는다면 P_b 값 또한 변하지 않는다는 의미이다.



[그림 2] 제안하는 방식의 기본 아이디어

광선과 교차한 해당 삼각형에 대한 광선의 길이는 $\mathbf{P}_l$이라고 하고, 해당 삼각형은 시점 벡터에 수직이라고 가정한다. 이 경우 광선과 교차한 삼각형의 텍스쳐에 대한 MIP-Map 수준은 다음의 수식과 같이 간단하게 구할 수 있다.

$$lod = \log_2\left(\frac{P_l}{P_b}\right) = \log_2(P_l) - \log_2(P_b) \qquad \text{(식 2)}$$

(식 2)에서 $\log(P_b)$는 전처리 과정에서 각 삼각형에 대하여 계산하여 미리 저장할 수 있다. 즉, 정적인 장면일 경우 렌더링 이전에 1번만 전처리를 수행하면 되고, 동적인 장면일 경우 매 프레임마다 전처리를 수행해야 한다. 전처리가 수행된 후에는 렌더링 시 MIP-Map 수준 선택을 위하여 $\mathbf{P}_l$에 대한 로그 연산 후에 뺄셈 연산만을 필요로 한다.

본 방식의 단점으로는 광선과 교차된 삼각형이 시점과 수직이라는 가정으로 면적에 기반한 방식이기 때문에 기존 방식에 비하여 알리아싱이 다소 발생할 수 있다. 예를 들어, [7]에서도 언급한 상황인 텍스쳐의 변화량이 얇은 사변형 형태인 경우이다.

3.2 P_b를 구하는 과정

해당 삼각형의 텍스쳐에 대한 P_b를 구하는 과정은 다음의 5단계로 구성된다.

1. 픽셀에 대한 텍셀의 크기를 구한다.
2. 세 정점에 대한 삼각형의 내부에 포함되는 텍셀의 개수를 구한다.
3. 1과 2의 결과 값을 가지고 텍셀로 이루어진 삼각형에 대한 픽셀의 크기를 구한다.
4. 세 정점에 대한 삼각형의 크기를 구한다.
5. 3과 4의 결과 값을 가지고 픽셀과 텍셀의 비율이 1:1이 되는 거리 P_b를 구한다.

첫 번째 단계에서는 텍스쳐 크기와 화면 크기를 이용하여 픽셀에 대한 텍셀의 크기를 구하며, 이는 다음식과 같다.

$$X_{PS} = \frac{Texture\,size}{Resolution} \qquad \text{(식 3)}$$

주어진 삼각형의 세 정점에 해당하는 텍스쳐 좌표가 (s_0,t_0), (s_1,t_1), (s_2,t_2)인 경우, 두 번째 단계에서는 이 세 좌표를 가지고 삼각형에 들어가는 텍셀의 개수를 구한다. 이는 삼각형의 넓이를 구하는 공식이며, 다음 식과 같다.

$$T_{XN} = (((s_0 \cdot t_1) + (s_1 \cdot t_2) + (s_2 \cdot t_0) - (t_0 \cdot s_1) - (t_1 \cdot s_2) - (t_2 \cdot s_0))/2) \cdot Texturesize \qquad \text{(식 4)}$$

세 번째 단계에서는 첫 번째와 두 번째 단계에서 계산된 값을 가지고 텍셀로 이루어진 삼각형의 크기를 구한다. 이는 다음 식과 같다.

$$T_{XS} = T_{XN} \cdot X_{PS} \quad \text{(식 5)}$$

주어진 삼각형의 세 정점에 해당하는 모델 좌표가 (x_0,y_0,z_0), (x_1,y_1,z_1), (x_2,y_2,z_2) 인 경우, 네 번째 단계에서는 이 값을 가지고 삼각형의 크기를 구한다.

$$(x_t,y_t,z_t) = \{(x_1,y_1,z_1)-(x_0,y_0,z_0)\} \times \{(x_2,y_2,z_2)-(x_0,y_0,z_0)\}$$
$$T_{area} = \sqrt{x_t^2+y_t^2+z_t^2} \quad \text{(식 6)}$$

다섯 번째 단계에서는 세 번째 단계와 네 번째 단계에서 계산된 값을 가지고 픽셀과 텍셀의 비율이 1:1이 되는 거리 P_b를 구한다. 이는 다음 식과 같다.

$$P_b = \sqrt{\frac{T_{XS}}{T_{area}}} \quad \text{(식 7)}$$

4. 실험 결과

실험을 위한 벤치마크는 광선 추적 방식에 대한 데모 및 성능 측정 시 많이 사용되는 BART[13]를 사용하였다. [그림 3]은 BART의 세 가지 모델 Kitchen, Robots, Museum이며, Kitchen은 두 개의 영상에 대하여 실험을 하였으며, 두 번째 영상인 경우 Dragon이 포함되어 렌더링 시간이 가장 오래 걸린다.

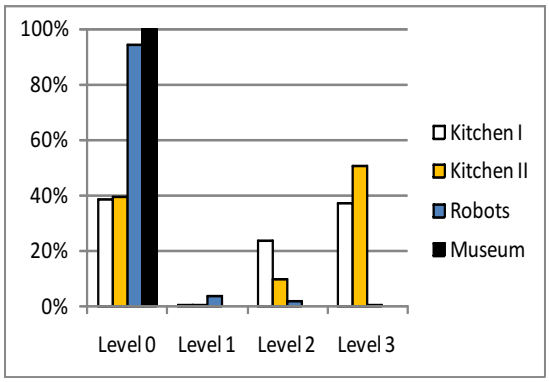
4.1 MIP-Map 수준 선택 비율

[그림 4]는 각 영상의 MIP-Map 수준의 선택 비율을 측정한 것이다. 실험 결과 Kitchen I과 Kitchen II는 반사와 굴절이 상대적으로 많기 때문에 높은 수준의 MIP-Map의 선택 비율이 다소 높았다. 반면에 Robot과 Museum은 반사 굴절이 상대적으로 적고 멀리 있는 오브젝트가 차지하는 화

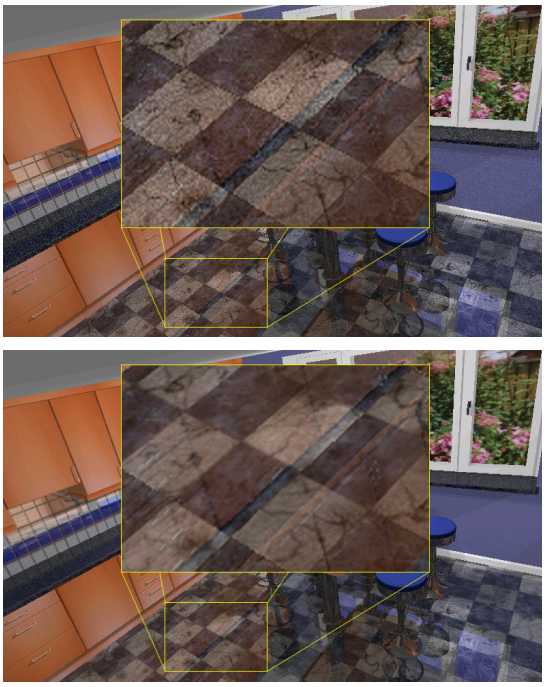
[그림 3] BART 벤치마크 : Kitchen I, Kitchen II, Robots, Museum

면의 비중이 작기 때문에 낮은 수준의 MIP-Map을 대부분 사용한 것을 확인 할 수 있다.

[그림 5]에서는 상단에 MIP-Map 적용 전과 하단에 적용 후의 영상을 확대하여 보여주고 있다. 전자가 후자에 비하여 영상에 알리아싱이 상대적으로 많기 때문에 다소 거친 영상을 보여주고 있다.



[그림 4] MIP-Map 수준 선택 비율



[그림 5] MIP-Map 적용 전과 후에 대한 영상

4.2 성능 측정

4.2.1 MIP-Map 적용에 따른 수행시간 비교

[표 1]에서는 MIP-Map을 적용하지 않았을 때와 적용했을 때의 수행 시간이 비교되었고, MIP-Map 적용으로 인한 성능 저하 비율 또한 제시되었다. 4가지 벤치마크에 대해 광원의 개수를 1~6개로 변화시켜가며 측정하였다. 3.2절에 제시한 P_b는 전처리를 통하지 않고 필요할 때 마다 매번 구하였으며, 이 과정 중에 T_{XN} 값은 렌더링 시 광선과 삼각형간의 교차점 검사 때 계산되는 값임으로 별도의 계산 없이 이를 그대로 사용하였다.

모든 장면에 공통적으로 광원의 개수가 늘어감에 따라 수행 시간이 증가하였고 MIP-Map을 적용한 상태의 수행 시간이 적용하지 않은 상태의 수행 시간보다 조금씩 더 길었다. [표 1]에서 보듯이 수행 시간 수치가 아주 작은 차이를 나타내고 비율로 보았을 때 0.1~2.6%정도의 시간이 더 걸리는 것을 확인 할 수 있다. MIP-Map을 추가함으로 인한 성능저하가 미미함을 확인 할 수 있다.

[표 1] 각 장면에 대한 수행 시간 비교 및 성능 저하 비율

# of light	Kitchen I	Kitchen II
1	2.19/2.23(1.54%)	4.61/4.64(0.61%)
2	3.32/3.36(1.43%)	8.55/8.60(0.52%)
3	4.39/4.46(1.64%)	12.28/12.39(0.91%)
4	5.40/5.45(1.05%)	15.92/15.96(0.25%)
5	6.41/6.46(0.85%)	19.52/19.58(0.30%)
6	9.61/9.68(0.71%)	28.63/28.68(0.19%)
# of light	**Robots**	**Museum**
1	2.88/2.95(2.61%)	1.04/1.06(1.74%)
2	4.87/4.95(1.55%)	1.85/1.87(0.97%)
3	6.79/6.86(0.91%)	2.78/2.79(0.33%)
4	8.62/8.74(1.44%)	3.54/3.56(0.42%)
5	10.46/10.55(0.81%)	4.41/4.43(0.34%)
6	13.98/14.07(0.59%)	6.51/6.51(0.07%)

5. 결론 및 향후 계획

본 논문에서는 광선 추적법 텍스쳐 매핑에서 사용되고 있는 MIP-Map 수준 선택을 효과적으로 할 수 있는 알고리즘을 제안하였다. 시점과 물체 사이의 거리 정보만을 이용하여 MIP-Map을 적용함으로써 기존의 방식에 비하여 효과적이다.

제안하는 방식과 광선 변화량 기법에 대한 화질과 속도 향상 측면에서의 정량적인 비교는 많은 실험과 분석을 요구하기 때문에 향후 연구를 통하여 별도의 논문에서 다루기로 하겠다. 추가적인 향후 연구로써 제안하는 방식을 FPGA 보드 상에서 하드웨어에 구현하여 제안하는 방식의 장점을 측정할 계획이다. 또한, 시점과 수직을 이루지 않고 기울어 있거나 회전되어있는 물체에 왜곡이 없는 텍스쳐를 입힐 수 있도록 하기 위해 필터링 기법 중 SAT(summed area table) 및 이방형 필터링(anisotropic filtering) 기법을 연구, 적용시킬 수 있도록 할 예정이다.

참고문헌

[1] J. Hurley, “Ray Tracing Goes Mainstream,” Intel Technology Journal, Vol. 9, No. 2, pp. 99-108, May 2005.

[2] K. Suffern, Ray Tracing from the Ground Up, Wellesley, AK Peters Ltd., 2007.

[3] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan, and P. Hanrahan, “Larrabee: A Many-core x86 Architecture for Visual Computing,” ACM Transactions on Graphics, Vol. 27, No. 3, pp. 1-15, Aug. 2008.

[4] K. Zhou, Q. Hou, R. Wang, and B. Guo, “Real-time KD-tree Construction on Graphics Hardware,” ACM Transactions on Graphics, Vol. 27, No. 5, pp. 1-11, Dec. 2008.

[5] J. P. Ewins, M. D. Waller, M. White, and P. F. Lister, "MIP-Map Level Selection for Texture Mapping," IEEE Transactions on Visualization and Computer Graphics, vol. 4, no. 4, pp. 317-329, Dec. 1998.

[6] L. Gritz and J. Hahn, "BMRT: A Global Illumination Implementation of the RenderMan Standard," Journal of Graphics Tools, Vol. 1, No. 3, pp. 29-47, 1996.

[7] H. Igehy, "Tracing Ray Differential," In Proceedings of SIGGRAPH, pp. 179-186, 1999.

[8] P. H. Christensen, D. M. Laur, J. Fong, W. L. Wooten, and D. Batali, "Ray Differentials and Multiresolution Geometry Caching for Distribution Ray Tracing in Complex Scenes," Computer Graphics Forum (Eurographics), Vol. 22, No. 3, pp. 543-552, Sept. 2003.

[9] P. H. Christensen, J. Fong, D. M. Laur, and D. Batali, "Ray Tracing for the Movie 'Cars'," In Proceedings of the IEEE Symposium on Interactive Ray Tracing 2006, pp. 1-6, Sept. 2006.

[10] J. Schmittler, I. Wald, and P. Slusallek, “SaarCOR-A Hardware Architecture for Ray Tracing,” In Proceedings of Graphics Hardware, pp. 27-36, 2002.

[11] Woo-chan Park, Jae-ho Nah, Jeong-soo Park, Kyung-ho Lee, Dong-seok Kim, Sang-duk Kim, Jin-hong Park, Yoon-sig Kang, Sung-bong Yang, and Tack-don Han, "An FPGA Implementation of Whitted-style Ray Tracing Accelerator," In Proceedings of IEEE Symposium on Interactive Ray Tracing, pp. 187, 2008.

[12] P. S. Heckbert, "Texture Mapping Polygons in Perspective," Computer Graphics Lab., New York Institute Technology, Technical Memo, No. 13, 1983.

[13] J. Lext, U. Assarsson, and T. Akenine -Moeller, "A Benchmark for Animated Ray Tracing," IEEE Computer Graphics and Applications, vol. 21, no. 2, pp. 22-31, Mar. 2001.

박 우 찬 (Park, Woo Chan)

1995 연세대학교 대학원(컴퓨터과학과 석사)
2001-2003 연세대학교 아식설계공동연구소 연구교수
2003-2007 세종대학교 컴퓨터공학과 조교수
2008 세종대학교 컴퓨터공학과 부교수

관심분야 : 3D Graphics Processor, Real-time Ray Tracer

김 동 석 (Kim, Dong Seok)

2006-2007 세종대학교 대학원(컴퓨터공학 석사)
2008 세종대학교 대학원(컴퓨터공학 박사과정)

관심분야 : 3D Graphics Processor, Real-time Ray Tracer