대화형 광선 추적법을 위한 그림자 컬링 알고리즘※

나재호○*, 박우찬**, 한탁돈*
연세대학교 컴퓨터과학과*, 세종대학교 컴퓨터공학과**
jhnah@msl.yonsei.ac.kr, pwchan@sejong.ac.kr, hantack@msl.yonsei.ac.kr

A Shadow Culling Algorithm for Interactive Ray Tracing

Jae-Ho Nah○*, Woo-Chan Park**, Tack-Don Han*
Dept. of Computer Science, Yonsei University*
Dept. of Computer Engineering, Sejong University**

요 약

본 논문은 대화형 광선 추적법에서 사용 가능한 새로운 그림자 컬링 알고리즘을 제안한다. 본 접근 방법은 그림자 자료 구조의 구축에 의한 전처리 방법 대신 프레임 간의 일관성을 이용하므로 동적 장면 상에서의 광선 추적법 처리에 적합하다. 본 알고리즘에서 그림자 계산 결과는 각각의 프리미티브 상에 저장이 되며 이 결과는 다음 프레임에 다시 사용된다. 또한 본 논문은 새로운 차폐 검사 방법을 제안한다. 이는 본 컬링 알고리즘에서 생길 수 있는 그림자 오류를 보정하며, 낮은 추가 비용을 요구한다. 실험 결과, 본 그림자 컬링 알고리즘은 7-19%의 탐색 비용 감소와 9-24%의 교차 비용 감소를 가져오는 것으로 나타났다.

ABSTRACT

We present a novel shadow culling algorithm for interactive ray tracing. Our approach exploits frame-to-frame coherence instead of preprocessing of building shadow data, so this algorithm is suitable for dynamic ray raying. In this algorithm, shadow processing results are stored to each primitive and used in the next frames. We also present a novel occlusion testing method. This method corrects potential shadow errors in our culling algorithm and requires low overhead. Experiment results show that our algorithm reduced both the traversal cost by 7-19 percent and the intersection cost by 9-24 percent.

Keyword : Ray tracing, real-time rendering, shadow algorithm

접수일자 : 2009년 09월 18일
심사완료 : 2009년 10월 26일
※ 이 논문은 연세대학교 대학원 재원으로 대학원 총학생회의 지원을 받아 연구되었음.

1. 서론 및 관련 연구

1.1 서론

미국 인텔社의 Jim Hurley가 “광선 추적법(ray tracing)은 주류로 간다[1].”고 주장한 것을 비롯해, 많은 연구 집단에서 광선 추적법이 가까운 미래에 실시간 렌더링의 주류 알고리즘으로 자리잡을 것이라 확신하고 있다. 이는 광선 추적법이 오프라인 렌더링(offline rendering) 위주로 쓰여왔던 것을 넘어 서서 차세대 실시간 렌더링(real-time rendering) 알고리즘으로도 각광을 받고 있음을 의미한다. 이러한 주장은 다음과 같은 근거를 가지고 있다[2].

첫째, 광선 추적법은 대규모 데이터 세트(massive data set)의 렌더링에 이점이 있다. 현재 실시간 렌더링에서의 주요 알고리즘으로 사용되고 있는 래스터라이제이션(rasterization)과 달리, 광선 추적법은 프리미티브의 개수에 대해 대수 복잡도(logarithmic complexity)를 가지기 때문이다.

둘째, 광선 추적법은 반사, 굴절, 그림자와 같은 물리적 현상에 기반한 효과(physically-based effects)를 자연스럽게 지원한다. 이는 이러한 효과를 복잡한 멀티-패스(multi-pass) 렌더링으로 근사(approximation)하는 래스터라이제이션 알고리즘과 차별화된다. 따라서 광선 추적법은 화질 측면과 컨텐츠 개발 측면에서 모두 이점이 있다.

셋째, 광선 추적법은 삼각형 이외에도 원, 구, 삼각뿔, 2차 곡면 등 다양한 프리미티브 형식을 지원한다. 이는 점, 선, 다각형만 지원하는 래스터라이제이션 알고리즘과 차별화된다.

이러한 이점들에도 불구하고 광선 추적법은 방대한 계산 비용이라는 중대한 문제점을 가지고 있다. 이는 광선 추적법이 현실감 넘치는 영상을 만들기 위해 많은 광선-프리미티브 간 교차 검사를 요구하기 때문이다. 이러한 특성 때문에 그 동안 광선 추적법은 오프라인 렌더링 위주로 쓰여 왔다.

그동안 광선 추적법의 성능상 문제점을 해결하기 위한 가속 기법들에 대해 다양한 연구가 진행되어 왔다. 이러한 기법들은 크게 교차 검사 가속(faster intersections), 광선 감소(fewer rays), 일반화된 광선(generalized rays)으로 나뉜다[3]. 그 중 그림자 광선의 감소와 같은 대부분의 광선 감소 기법들은 정적 장면(static scene) 위주로 연구되었다. 하지만 이러한 기법들은 동적 장면(dynamic scene)에는 바로 적용하기 힘들다. 그 이유는 이러한 기법들에서 사용하는 전처리(preprocessing)를 통한 자료 구축이 동적 장면에서는 매 프레임마다 수행되어야 하기 때문이다.

하지만 효과적인 그림자 광선 처리는 동적 장면 기반의 대화형 광선 추적법에서도 여전히 중요하다. 휘티드 방식 광선 추적법(Whitted-style ray tracing)[4]에 따르면, 그림자 광선의 개수는 광원의 수와 비례한다. 그러므로 광원의 수가 많아질수록 그림자 광선의 비율은 높아지게 된다. 예를 들어 광원이 9개가 있을 경우 그림자 광선이 전체 광선에서 차지하는 비율은 90%이다. 그 이유는 하나의 비 그림자 광선(1차 광선, 반사 광선, 굴절 광선 등)이 광원의 개수(9개)만큼 그림자 광선을 파생시키기 때문이다. 물론 이러한 특성은 동적 장면에서도 변하지 않는다.

본 논문은 동적 장면 기반 대화형 광선 추적법을 위한 새로운 그림자 컬링(shadow culling) 알고리즘을 제안한다. 본 알고리즘은 그림자 여부를 각각의 프리미티브 별로 누적시키고, 이 정보를 다음 프레임에서 불필요한 그림자 광선 추적을 방지하는 데 사용한다. 만약 그림자가 동적인 물체에 의해 변경될 경우, 그림자 처리는 관련된 프리미티브에 대해서만 수행된다. 또한 가려진 영역의 그림자가 계산되지 않는 특성을 감안하여, 현재 시점에서 프리미티브가 다른 물체에 가려지지 않았을 경우에만 그림자 정보를 갱신한다. 이러한 차폐 여부의 탐지는 새롭게 제안하는 차폐 검사 방법을 통해 이루어진다. 이 방법은 인접한 픽셀 상의 교차된 프리미티브들 간의 간단한 비교를 통해 현재 교차된 프리미티브의 차폐 여부를 결정한다.

본 그림자 처리 알고리즘은 다음과 같은 이점을 지닌다. 첫째, 본 알고리즘은 효과적으로 그림자

광선 처리 비용을 감소시켜 광선 추적법의 성능 향상에 도움을 준다. 둘째, 본 알고리즘은 별도의 전처리를 필요로 하지 않으므로 정적 장면 뿐 아니라 동적 장면에도 적합하다. 셋째, 본 알고리즘은 광선 추적기에 쉽게 구현할 수 있으며 추가 비용도 적다.

1.2 관련 연구

라이트 버퍼(The light buffer)[5]는 광선 추적법 상의 그림자 처리를 위한 초기 연구이다. 라이트 버퍼는 각각의 광원 상에 추가된 자료 구조의 이름이다. 각각의 라이트 버퍼는 3차원 그리드(grid) 6면체로 구성되어 있고, 각각의 그리드 셀(cell)은 광원으로부터 가시권에 있는 물체들의 면(surface)을 목록으로 가진다. 이러한 라이트 버퍼가 만들어진 이후 렌더링 시의 그림자 검사는 빠르게 결정된다. 그 이유는 그림자 광선이 전체 프리미티브가 아닌 셀 안에 저장된 프리미티브에 대해서만 교차 검사를 수행하면 되기 때문이다. 그러나 이 알고리즘은 두 가지 문제점을 지니고 있다. 첫째, 라이트 버퍼의 효율은 셀 크기에 반비례하나, 작은 셀을 가지는 그리드 구조는 그만큼 큰 메모리 용량을 필요로 한다. 둘째, 라이트 버퍼의 생성은 물체의 면 목록을 구축름이 위해 스캔 컨버전(scan conversion) 및 정렬을 필요로 한다. 이러한 라이트 버퍼의 구축 비용은 매우 비싸다.

그림자 캐싱(shadow caching)은 라이트 버퍼와 같은 논문[5]에서 제시된 알고리즘이다. 이 알고리즘은 물체 간의 일관성(coherence)을 이용한다. 물체 간의 일관성은 어떤 물체가 어떠한 광선에 대해 다른 물체에 의해 가려져 있는 것으로 판명이 되었다면 인접한 광선에서도 계속 가려질 가능성이 높다는 것을 의미한다. 그림자 캐싱은 이러한 일관성을 이용하기 위해 그림자를 발생시킨 차폐 물체(occluder)를 캐시에 저장한다. 그리고 다음 번 그림자 광선 추적시, 일반적인 교차 검사 전에 캐시에 저장된 차폐 물체를 검사한다. 만약 광선이 이 물체와 교차를 한다면 그림자 검사는 종료되고 그림자가 지는 것으로 결정된다. 이 알고리즘은 간단하고 그림자 검사에도 유용하다는 장점을 가지고 있지만, 몇 가지 한계를 가지고 있다[6]. 첫째, 일관적이지 않은 그림자 광선에는 적합하지 않다. 둘째, 가려지지 않는 그림자 광선에는 적용할 수 없다. 셋째, 물체가 작은 프리미티브로 구성되면 일관성이 떨어지므로 복잡한 장면일수록 효율이 낮아진다.

복셀 차폐 검사(voxel occlusion test)[7]는 라이트 버퍼와 달리 전처리 결과를 가속 구조에 저장한다. 전처리 단계에서, 그리드 가속 구조의 복셀은 full, null, complicated occlusion의 세 가지 모드를 가진다. 그 중 full 그리드와 null 그리드는 그리드 탐색 중에 그림자 검사를 필요로 하지 않는다. 따라서 그림자 검사는 complicated occlusion 그리드에 대해서만 수행된다. 하지만 이 방식은 가속 구조를 성능이 떨어지는 정규 그리드(uniform grid)로만 한정시킨다는 단점이 있다. 그러므로 가속 구조가 kd-tree나 BVH(bounding volume hierarchy)와 같은 계층적 트리 구조이고, 동적 장면 상에서 광선 추적법이 수행될 경우에는 이 복셀 차폐 검사의 적용이 어렵다. 그 이유는 동적 장면에서는 가속 구조가 매 프레임마다 갱신되어야 하기 때문이다.

지역 조명 환경(LIEs : local illumination environments)[8]은 기하 정보와 지역 조명 정보를 캐싱하는 자료 구조이다. 각각의 LIE는 옥트리(octree) 셀 안에 저장이 되며, LIE는 차폐 모드와 차폐 물체의 목록을 포함한다. LIE 구축은 시점에 의존적이므로 이 정보는 매 프레임마다 갱신되어야 한다. 그러므로 이 알고리즘은 정적 장면에서 카메라 위치만 바뀌는 워크쓰루(walkthrough)에는 적합하나 동적 장면에는 적합하지 않다.

Lightcuts[9]는 이진 광선 트리 군집 구조이다. 이 알고리즘은 많은 점광원(point lights)을 렌더링할 때 효율이 높아지는 구조이다. 왜냐하면 광선 트리의 불필요한 부분이 적응적으로 절단되기 때문이다. 실험 결과에 따르면 이 알고리즘은 광원의

수에 대해 하위-선형(sub-linear) 비용을 제공한다. 하지만 이 알고리즘은 광원의 수가 적을 때에는 성능 향상이 없다. 그러므로 본 알고리즘은 초고화질 영상을 필요로 하는 오프라인 렌더링에는 적합하나 광원의 수가 적은 대화형 광선 추적법에는 적합하지 않다.

2. 제안하는 그림자 컬링 알고리즘

2.1 전체 알고리즘의 흐름

제안하는 그림자 컬링 알고리즘은 프레임 간의 일관성을 이용한다. 본 논문에서 말하는 프레임간의 일관성이란, 정적 물체 간의 그림자 계산 결과가 다음 프레임에서도 변하지 않음을 뜻한다. 본 알고리즘은 이러한 특성을 불필요한 그림자 검사를 줄이는 데 이용한다. 즉, 그림자 계산 결과는 각각의 프리미티브 상에 저장이 되며, 이는 다음 프레임에서 재사용된다.

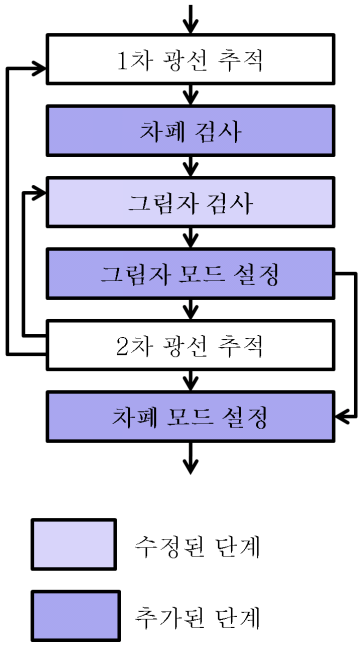
[그림 1]은 본 알고리즘의 전체 흐름이다. 이는 일반적인 광선 추적법과 유사하나 다음과 같은 차이점이 존재한다.

먼저, 차폐 검사(occlusion test) 단계가 추가된다. 이 단계에서는 인접한 픽셀간의 렌더링된 프리미티브들의 차폐 모드를 결정하기 위한 차폐 검사가 수행된다. 이 단계의 상세한 내용은 2.3절에서 기술될 것이다.

다음으로, 중복되는 그림자 계산을 감소시키기 위해 그림자 검사 단계가 수정된다. 본 단계는 그림자 컬링과 그림자 광선 추적을 포함한다. 이 단계의 상세한 내용은 2.4절에서 기술될 것이다.

그림자 검사가 끝나면, 그림자 모드 설정 단계에서는 그림자 계산 결과를 이용하여 교차된 프리미티브의 그림자 모드를 변경시킨다. 이 단계의 상세한 내용은 2.5절에서 기술될 것이다.

마지막으로, 차폐 모드(occlusion mode) 설정 단계가 프레임의 종료 전에 추가된다. 이 단계는 전체 렌더링된 프리미티브의 차폐 모드를 결정한다. 이 단계의 상세한 내용은 2.6절에서 기술될 것이다.



[그림 1] 전체 알고리즘의 흐름
– 일반적인 광선 추적법과 유사하나, 차폐 및 그림자 처리 부분에서 차이가 있다.

2.2 추가로 필요한 정보

2.1절에서 기술한 것과 같이, 본 알고리즘의 핵심은 그림자 처리 결과를 재사용하는 것이다. 이러한 재사용을 위해, 각 프리미티브는 다음과 같은 추가 정보를 가져야 한다.

- 차폐 모드 (2bit)
- 그림자 모드 (2bit*광원의 개수)
- 현재 렌더링된 픽셀 수 (1 int)
- 지금까지 렌더링된 픽셀 수의 최고치(1 int)

[그림 2]는 차폐 모드와 그림자 모드를 표현한다. 차폐 모드는 크게 INIT(초기 상태), PARTIALLY_OCCLUDED(부분적으로 가려진 상태), FULLY_RENDERED (완전히 가려진 상태),

DYNAMIC(동적 물체)으로 나눠진다. 그림자 모드는 INIT(초기 상태), FULL_SHADOW(완전히 그림자가 진 상태), NON_SHADOW(그림자가 지지 않은 상태), PARTIAL_SHADOW(부분적으로 그림자가 진 상태)로 나눠진다. 차폐 모드와 그림자 모드는 단지 몇 bit만 필요로 하므로, 광원의 개수가 10개보다 적을 때에는 하나의 32bit int로 표현될 수 있다.

차폐 모드는 프리미티브의 차폐 정보와 정적/동적 정보를 포함한다. 이 모드가 FULLY_RENDERED일 경우에만 그림자 모드는 그림자 검사 단계에서 그림자 컬링을 위해 쓰인다. 만약 그림자 모드가 FULL_SHADOW일 경우에는 그림자 처리를 하지 않으며, 그림자 모드가 NON_SHADOW일 경우에는 교차 지점과 동적 물체의 관계만 고려해 그림자 검사를 수행한다. 만약 그림자 모드가 PARTIAL_SHADOW일 경우 그림자 검사의 수행은 기존 광선 추적법과 동일하다.

차폐 모드 (2진 값)	
	INIT(00)
	PARTIALLY_ OCCLUDED(01)
	FULLY_ RENDERED(10)
	DYNAMIC(11)

그림자 모드 (2진 값)	
	INIT(00)
	FULL_SHADOW (01)
	NON_SHADOW (10)
	PARTIAL_ SHADOW (11)

[그림 2] 차폐 모드와 그림자 모드

현재 프레임에서 렌더링된 픽셀의 개수와 지금까지 렌더링되었된 픽셀 개수의 최고치는 차폐 모드 설정 단계에서 쓰인다. 만약 현재 프레임에서 투영된 프리미티브의 크기가 현재까지 렌더링 되었던 가장 큰 프리미티브 크기의 4배보다 클 경우, 차폐 모드는 INIT로 초기화된다. 이는 줌-인(zoom-in)된 프리미티브에 생길 수 있는 그림자 결점을 방지한다.

2.3 차폐 검사

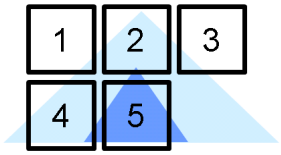
은면 제거(hidden surface removal)를 위해 깊이 검사(depth test)를 필요로 하는 래스터라이제이션 알고리즘과 달리, 광선 추적법은 깊이 검사와 같은 종류의 차폐 검사를 필요로 하지 않는다. 그 이유는 광선 추적이 광선과 가장 가까운 교차 지점에서 종료되기 때문이다. 그러나 본 그림자 컬링 알고리즘은 차폐 검사를 필요로 한다. 왜냐하면, 만약 프리미티브가 다른 프리미티브에 의해 가려져 전체 영역이 렌더링 된 적이 없을 경우, 프리미티브 상에 저장된 그림자 계산 결과는 유효하지 않기 때문이다.

이러한 이유로, 본 절에서는 본 그림자 컬링 알고리즘을 위한 새로운 차폐 검사 알고리즘을 제안한다. 이 차폐 검사 방법은 1차 광선 추적시, 현재 렌더링 중인 픽셀과 두 개의 참조 픽셀(왼쪽 픽셀과 위쪽 픽셀) 상의 프리미티브를 비교한다. 만약 현재 픽셀과 참조 픽셀 상의 프리미티브가 다를 경우, 현재의 1차 광선과 참조 픽셀 상의 프리미티브에 대한 교차 검사가 수행된다. 만약 교차 검사가 참이라면, 참조 픽셀 상의 프리미티브는 현재 픽셀의 프리미티브에 의해 부분적으로 차폐된 것으로 판단되어 차폐 모드가 PARTIALLY_OCCLUDED로 설정된다. 또한 현재 픽셀이 스크린의 가장자리에 있을 경우에도 같은 모드로 설정이 된다. 그 이유는 이러한 프리미티브들이 화면에 의해 절단되기 때문이다. [표 1]은 본 차폐 검사 알고리즘의 의사 코드이고, [그림 3]은 차폐 검사의 예를 나타낸다.

본 차폐 검사 알고리즘은 참조 픽셀에 대해 최소 0번에서 최대 2번까지의 추가 교차 검사를 필요로 한다. 이러한 검사는 프리미티브의 가장자리 부분에서만 이루어지므로, 추가 교차 검사 비용은 매우 낮다. 여기에 대한 실험 내용은 3.3절에서 기술한다.

[표 1] 알고리즘 1 – 차폐 검사의 의사 코드

```
1.  if currentPixel.hitTriangle.isDynamic() then
2.    currentPixel.hitTriangle.occlusionMode
        = DYNAMIC
3.  end
4.  else if
      currentPixel.isLocatedonEdgeofScreen()
      == true then
5.    currentPixel.hitTriangle.occlusionMode
        = PARTIALLY_OCCLUDED
6.  end
7.  else
8.    referencePixel
        = frameBuffer.getLeftPixel(currentPixel)
9.    if referencePixel.hitTriangle.OcclusionMode
        == INIT then
10.     if currentPixel.hitTriangle !=
          referencePixel.hitTriangle then
11.     if intersectionTest
          (ray,referencePixel.hitTriangle) then
12.       referencePixel.hitTriangle.occlusionMode
            = PARTIALLY_OCCLUDED
13.     end
14.   end
15. end
16. referencePixel =
      frameBuffer.getUpperPixel(currentPixel)
17. Repeat line 9-15
18. end
```



[그림 3] 차폐 검사의 예

– 만약 현재 렌더링 중인 픽셀이 5일 경우, 픽셀 4(5의 왼쪽 픽셀)과 픽셀 2(5의 위쪽 픽셀)에 대해 차폐 검사가 실행된다. 이 그림에서는 픽셀 2와 픽셀 4에 교차된 삼각형이 동일하므로 교차 검사는 한번만 수행된다. 이 검사가 끝나면, 렌더러는 흐린 삼각형이 어두운 삼각형에 의해 가려졌음을 알게 된다.

2.4 그림자 검사

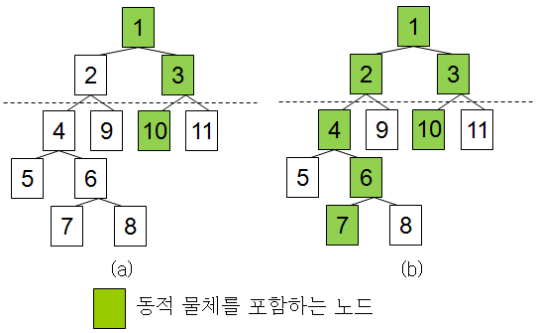
그림자 검사 단계는 처리중인 그림자 광선의 그림자 여부와 그림자 컬링 여부를 판단한다. 이 그림자 컬링의 수행 여부는 교차된 프리미티브의 차폐 모드와 그림자 모드에 따라 결정된다. [표 2]에 본 그림자 검사의 상세한 의사 코드가 기술되어 있다. 만약 현재 프리미티브 전체에 그림자가 져 있을 경우에는 그림자 검사를 할 필요가 없다. 따라서 이 경우에는 그림자 광선의 추적을 수행하지 않고 그림자 광선이 교차된 것으로 설정한다.

[표 2] 알고리즘 2 – 그림자 검사의 의사 코드

```
1.  if currentRay.
      triangleAtOrigin.occlusionMode
      == FULLY_RENDERED then
2.    if currentRay.
        triangleAtOrigin.shadowMode
        (currentLight)==FULL_SHADOW then
3.      currentRay.hitResult = false
4.    end
5.    else if currentRay.
        triangleAtOrigin.shadowMode
        (currentLight)==NON_SHADOW then
7.      TraversalOnlyDynamicObject
          (currentRay)
8.    end
9.    else
10.     Traversal(currentRay)
11.   end
12. end
13. else
14.   Traversal(currentRay)
15. end
```

현재 처리중인 프리미티브의 그림자 모드가 NON_SHADOW일 경우에는 다른 동적 물체가 만드는 그림자에 대해서 처리를 해주어야 한다. 이 경우, 현재 처리중인 그림자 광선의 탐색은 동적 물체에 대해서만 수행된다. 이 때, 가속 구조의 노드가 동적 물체를 포함하지 않을 경우 그림자 광선 추적은 중단된다. 따라서 본 알고리즘은 가속 구조 생성 시 노드의 절단 면(split plane)이 프리미티브의 경계 대신 물체의 경계면으로 설정될 때 높은 효율을 갖는다. 그러므로 장면 그래프(scene

graph) 계층으로부터 가속 구조를 구축하는 알고리즘[10]은 본 그림자 컬링 알고리즘에 매우 적합하다. [그림 4]는 다르게 생성된 두 개의 트리를 예로 들어 본 그림자 컬링 알고리즘과 트리 구축 알고리즘간의 관계를 설명하고 있다.



[그림 4] 그림자 컬링 알고리즘과 트리 구축 알고리즘간의 관계

- (a)와 (b)는 같은 장면에 대해 다르게 만들어진 가속 구조로, (a)는 본 알고리즘에 적합한 트리 구조이고 (b)는 적합하지 않은 트리 구조이다. 예를 들어, 그림자 광선의 시작점에 위치하는 프리미티브가 NON_SHADOW 모드를 가지고 그림자 광선이 노드 3에 교차하지 않는 것으로 판단될 경우, 트리 탐색은 노드 1 또는 노드 2에서 중단된다. 반면 (b)에는 (a)와 달리 노드 10에 포함된 동적 물체가 노드 7로도 분리되어 있다고 가정하였다. 이 경우 (b)의 트리 탐색 비용은 (a)에 비해 훨씬 증가하게 된다. 이는 (a)와 달리 노드 7까지 탐색을 수행해야 할 수도 있기 때문이다. 가속 구조 구축 알고리즘으로 장면 그래프 계층을 이용하여 물체의 경계면 단위로 트리를 만드는 방법을 사용하여 정적 물체와 동적 물체를 가지는 노드를 분리하면, (b)와 같은 경우가 생길 확률이 줄어들게 된다. 따라서 이러한 방법은 본 그림자 컬링 알고리즘의 성능을 향상시키는 데 적합하다.

2.5 그림자 모드 설정

2.2절에서 언급한 것처럼, 그림자 모드의 크기는 2bit이다. 첫 번째 bit는 그림자가 짐을 의미하고 두 번째 bit는 그림자가 지지 않았음을 의미한다. 그러므로 그림자 모드 설정은 매우 간단한 OR 연산을 통해 이루어질 수 있다. 만약 프리미티브가 부분적인 그림자를 가질 경우에는, 그림자 모드는 01(FULL_SHADOW)과 10(NON_SHADOW)의 OR 연산을 통해 11(PARTIAL_SHADOW)이 된다. [표 3]은 그림자 모드 설정의 의사 코드를 나타낸다.

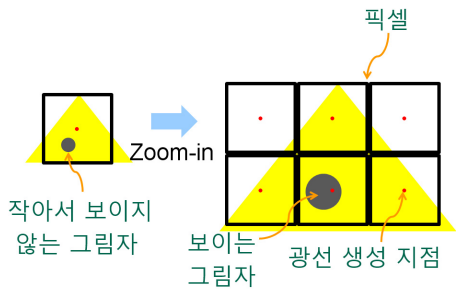
[표 3] 알고리즘 3 – 그림자 모드 설정의 의사 코드

```
1. if currentRay.cullingResult == false than
2.   if currentRay.hitResult == true than
3.     currentRay.triangleAtOrigin.
         OROperationShadowMode
         (currentLight, FULL_SHADOW)
4.   end
5.   else
6.     currentRay.triangleAtOrigin.
         OROperationShadowMode
         (currentLight, NON_SHADOW)
7.   end
8. end
```

2.6 차폐 모드 설정

이 단계는 프레임이 끝나기 전에 실행되어, 전체 렌더링 된 프리미티브의 차폐 모드를 설정한다. 만약 이 단계에서 프리미티브의 차폐 모드가 INIT라면, 이 모드는 FULLY_RENDERED로 바뀐다. 이는 이 프리미티브가 2.3절의 차폐 검사 결과 다른 어떠한 프리미티브에 의해서도 가려지지 않았음을 의미한다.

한편 2.2절에서 언급한 것과 같이, 차폐 모드가 FULLY_RENDERED이고, 현재 프레임에서 렌더링된 픽셀의 개수가 지금까지 렌더링 되었던 픽셀 개수의 최고치의 4배보다 클 경우에는 차폐 모드가 INIT로 초기화된다. 이는 [그림 5]에 나오는 것과 같은 줌-인(zoom-in)시에 생길 수 있는 그림자 결점을 방지하기 위해 꼭 필요한 과정이다.



[그림 5] 줌-인시 생길 수 있는 그림자 결점의 예
– 왼쪽 그림은 줌-인을 하기 전의 영상이고, 오른쪽 그림은 줌-인을 한 후의 영상이다. 줌-인을 통해 시점이 바뀐 이후에는, 픽셀보다 매우 작아 보이지 않던 그림자가 보여질 수 있다. 본 알고리즘은 이와 같이 생길 수 있는 일관성 문제를 차폐 모드 설정을 통해 해결한다.

3. 실험 및 결과

3.1 실험 설정

실험을 위해 사용한 환경은 Whitted 방식 광선 추적기이다. 광선 추적기의 자료 구조는 kd-tree를 사용하였으며, 자료 구조 구축 시 장면 그래프를 이용하여 트리를 분할하는 알고리즘을 사용하였다. 이는 동적인 물체와 정적인 물체를 엄격하게 구분하므로 본 그림자 컬링 알고리즘에 적합하다.

벤치마크로는 [그림 6]과 같은 BART[11]의 두 가지 장면을 사용하였다. 첫 번째 장면은 11만개의 삼각형을 가진 BART Kitchen으로, 80% 이상이 정적 물체로 이루어져 있고, 22 프레임으로 구성된 장면이다. 두 번째 장면은 7만여개의 삼각형을 가진 BART Robot으로, 80% 이상이 동적 물체로 이루어져 있고, 52 프레임으로 구성되어 있다. 실험에서 사용한 해상도는 512×512이고, 광선의 최고 재귀 깊이(recursion depth)는 10으로 설정하였다.

[그림 6] 벤치마크에 사용한 장면
– 정적 물체 위주의 BART Kitchen(좌), 동적 물체 위주의 BART Robot(우)

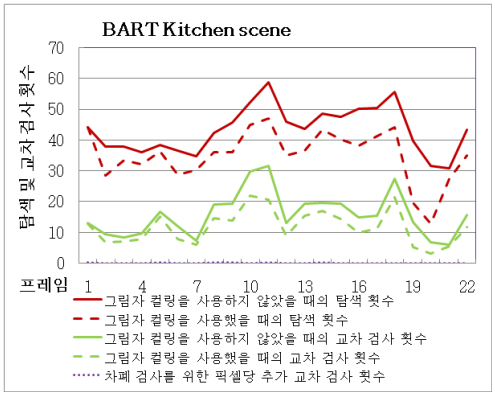
본 알고리즘의 성능 향상치를 측정하기 위해 사용한 측정 체계(metric)는 탐색 및 교차 검사의 횟수이다. 이러한 측정 체계를 사용한 이유는 탐색 및 교차 검사의 횟수가 플랫폼 독립적(platform-independent)이기 때문이다. 또한 본 알고리즘은 그림자 처리에 관한 내용이므로, 탐색 및 교차 검사 횟수는 그림자 광선에 대해서만 측정하였다.

3.2 비용 절감 결과

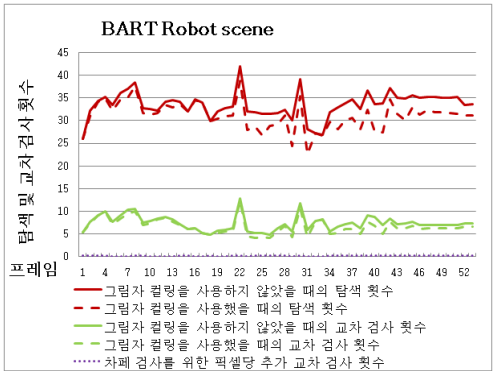
[표 4]는 비용 절감 결과를 요약한 도표이고, [그림 7]과 [그림 8]은 실험 결과를 프레임별로 표현한 그래프이다. 본 알고리즘은 BART Robot보다는 BART Kitchen에서 더 좋은 성능을 나타낸다. 그 이유는 전자는 동적 물체 위주인 반면, 후자는 정적 물체 위주이기 때문이다. 2.4절에 언급한 것처럼, 동적 물체는 컬링 효과를 떨어뜨린다.

[표 4] BART Kitchen 및 Robot 장면에서의 실험 결과

장면	탐색 비용 절감 비율 (단위: 퍼센트)		교차 검사 비용 절감 비율 (단위: 퍼센트)		추가교차검사횟수
	평균	최대	평균	최대	
BART Kitchen	19.8	59.5	24.5	59.8	픽셀당 0.173회
BART Robot	7.4	19.5	9.5	29.6	픽셀당 0.149회



[그림 7] BART Kitchen 장면에서의
프레임별 탐색 및 교차 검사 횟수



[그림 8] BART Robot 장면에서의
프레임별 탐색 및 교차 검사 횟수

BART Kitchen의 실험 결과는 다음과 같다. 노드 탐색은 평균 19.8%, 최고 59.5%(20번째 프레임) 감소했으며, 광선-삼각형 교차 검색은 평균 24.5%, 최고 59.8%(19번째 프레임) 감소하였다.

BART Robot의 실험 결과는 다음과 같다. 노드 탐색은 평균 7.4%, 최고 19.5%(31번째 프레임) 감소했으며, 광선-삼각형 교차 검색은 평균 9.5%, 최고 29.6%(41번째 프레임) 감소하였다.

3.3 추가 비용 측정

본 그림자 컬링 알고리즘은 [그림 1]에 나타낸 것처럼 세 가지 추가 단계를 필요로 한다. 그림자 모드 설정과 차폐 모드 설정은 단순한 대입 또는 OR 연산만 필요로 하므로 추가 비용이 매우 적으나, 차폐 검사는 추가적인 광선-프리미티브 교차 검사를 필요로 한다. 본 절에서는 이러한 차폐 검사의 추가 비용에 대해 측정한 결과를 기술한다.

비용 측정 결과는 앞 절의 결과와 마찬가지로 [표 4], [그림 7], [그림 8]에 나타나 있다. BART Kitchen 장면에서의 추가 교차 검사는 픽셀당 0.173번에 불과하며, BART Robot 장면에서의 추가 교차 검사는 픽셀당 0.149번에 불과하다. 이는 성능 저하에 영향을 많이 안 끼치는 매우 낮은 수준이다. 이와 같이 추가 비용이 낮은 이유는 2.3절에 기술한 바와 같이 1차 광선을 추적할 때 프리미티브의 가장자리 부분에서만 교차 검사가 이루어지기 때문이다.

4. 결론 및 향후 계획

본 논문은 대화형 광선 추적법을 위한 새로운 그림자 컬링 알고리즘을 제시하였다. 이 알고리즘은 그림자 처리 결과를 누적하고 이를 프레임 간의 일관성을 이용하기 위해 재사용한다. 그러므로 본 알고리즘은 이전의 다른 관련 연구와 달리 복잡한 전처리 과정을 필요로 하지 않아 동적 장면에 유리하다. 또한 본 논문은 정확한 그림자 표현을 위한 새로운 차폐 검사 알고리즘을 제시하였으며, 이 방법은 매우 미미한 추가 비용을 필요로 한다. 결론적으로 본 알고리즘은 광선 추적법의 그림자 계산 비용을 효과적으로 감소시킨다.

향후 연구로는 타 알고리즘과의 혼용을 통한 성능 개선에 주안점을 두고 있다. 먼저 본 알고리즘은 부분적으로 가려지는 프리미티브에 대해서는 컬링을 하지 못한다. 따라서 본 알고리즘과 1.2절에서 언급한 그림자 캐싱[5]을 함께 사용한다면, 부분적으로 가려지는 프리미티브의 그림자도 좀 더 효율적으로 처리할 수 있을 것이다. 다음으로, 본

알고리즘은 부분적으로 그림자가 진 프리미티브에 대해서 컬링을 하지 못한다. 하지만 대표적인 프리미티브 형식인 삼각형에 대해서는 삼각형의 분할 알고리즘[12]을 사용함으로써 부분적으로 그림자가 진 영역의 비율을 줄일 수 있다. 이는 본 알고리즘의 컬링 효율을 높이는 데 기여할 것이다.

참고문헌

[1] Hurley J., “Ray Tracing Goes Mainstream,” Intel Technology Journal, vol. 9, no. 2, pp. 99-107, May 2005.

[2] Slusallek P., Shirley P., Mark W., Stoll G., and Wald I., “Introduction to Realtime Ray Tracing,” ACM SIGGRAPH 2005 Course on Interactive Ray Tracing, Aug. 2005.

[3] Glassner A.S., An Introduction to Ray Tracing, Academic Press, 1989.

[4] Whitted J.T, "An Improved Illumination Model for Shaded Display," Communications of the ACM, vol. 23, no. 6, pp. 342-349, June 1980.

[5] Haines E.A and Greenberg D.P., "The Light Buffer: A Shadow-testing Accelerator," IEEE Computer Graphics and Applications vol. 6, no. 10, pp. 6-16, Sep. 1986.

[6] Wald I., Realtime Ray Tracing and Interactive Global Illumination, Ph.d thesis, Sarrland University, 2004.

[7] Woo A. and Amantides J., "Voxel Occlusion Testing: A Shadow Determination Accelerator for Ray Tracing," Proceedings of Graphics Interface ‘90, pp. 213 - 226, May 1990.

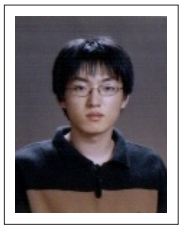
[8] Fernandez., Bala K., and Greenberg D.P, "Local Illumination Environments for Direct Lighting Acceleration," Proceedings of 13th Eurographics Workshop on Rendering, pp. 7 - 14, Jun. 2002.

[9] Walter, B., Fernandez, S., Arbree, A., Bala, K., Donikian, M., and Greenberg, D.P., “Lightcuts: A Scalable Approach to Illumination," ACM Transactions on Graphics, vol. 24, no. 3, pp. 1098-1107, Aug. 2005.

[10] Hunt W., Mark W.R., and Fussel D.S., "Fast and Lazy Build of Acceleration Structures from Scene Hierarchies," Proceedings of 2007 IEEE/EG symposium on interactive ray tracing, pp. 47-54, Sep. 2007.

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

[12] Dammertz H. and Keller A., "The Edge Volume Heuristic - Robust Triangle Subdivision for Improved BVH Performance," Proceedings of 2008 IEEE/EG symposium on interactive ray tracing, pp. 155-157, Aug. 2008.



나 재 호(Jae-Ho Nah)

2005년 연세대학교 컴퓨터과학과 학사
2007년 연세대학교 컴퓨터과학과 박사
2007년~현재 연세대학교 컴퓨터과학과 박사과정

관심분야 : 렌더링 알고리즘, 그래픽스 하드웨어

박 우 찬(Woo-Chan Park)

1993년 연세대학교 전산과학과 학사
1995년 연세대학교 전산과학과 석사
2000년 연세대학교 컴퓨터과학과 박사
2001년~2003년 연세대학교 연구교수
2003년~현재 세종대학교 컴퓨터공학과 교수

관심분야 : 그래픽스 하드웨어, 렌더링 프로세서

한 탁 돈(Tack-Don Han)

1978년 연세대학교 전자공학과 학사
1983년 Wayne 주립대학교 컴퓨터공학과 석사
1987년 매사추세츠대학교 컴퓨터공학과 박사
1989년~현재 연세대학교 컴퓨터과학과 교수

관심분야 : 고성능 컴퓨터 구조, 미디어 시스템 구조, HCI, 유비쿼터스 컴퓨팅