특집

3D 게임 그래픽스 엔진에서의 실시간 그림자 처리 기술들

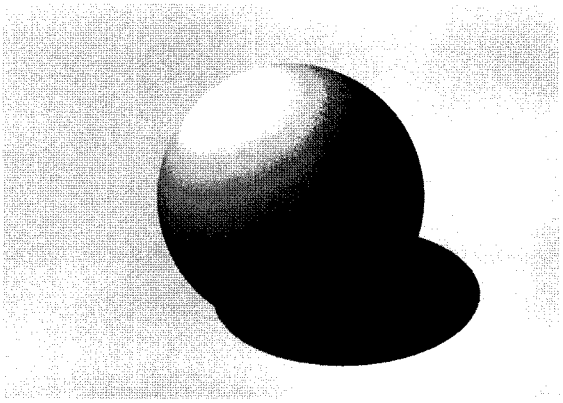
김도형(케이비온라인)

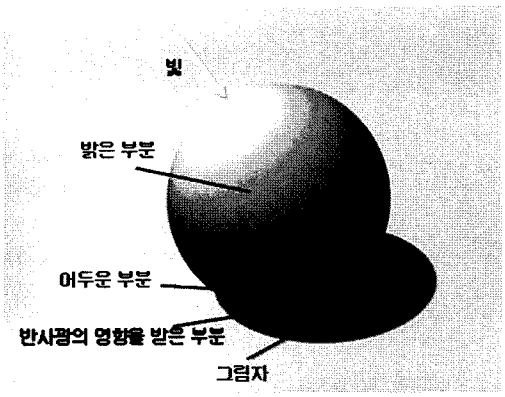
I. 서 론

영화 '친구'에서 어느 조폭 두목이 이런 말을 한다.

"우리는 비록 음지에 살지만 양지를 떠받치고 있다."

조폭이 과연 그런 삶을 사는지에 대해서는 의문이 들지만, 빛과 어두움이라는 주제를 놓고 볼 때 음지가 양지를 떠받친다는 것은 맞는 말이다. 음지가 어두울수록 양지는 더욱 밝게 보이게 된다. 회화나 사진, 그리고 컴퓨터그래픽스까지도 현실 세계를 표현하고자 하는 시도를 하는 분야에서는 언제나 빛과 어두움은 중요한 연구과제이다. 이 글은 이 중에서도 어두움 그리고 그 중에서도 그림자에 대해 3D 게임 그래픽스라는 분야에서 어떻게 다루어지고 있는지 이야기하려 한다. 그림자의 사전적 의미는 "물체가 빛을 가려서 그 물체의 뒷면에 드리워지는 검은 그늘"이다. <그림1>을 보면 하얗고 둥근물체가 하얀 바닥위에 놓여 있다.



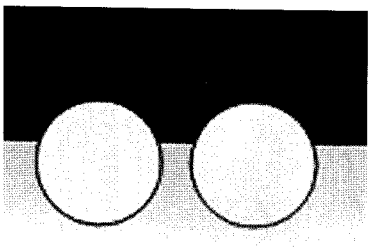


빛은 비스듬히 위쪽에서 비추고 있고 빛을 받는 부분은 밝게, 못 받는 부분은 어둡게 보인다. 어두운 부분을 조금 더 자세히 살펴보면 빛이 바닥에 충돌하여 반사된 반사광의 영향으로 살짝 밝아진 부분도 보인다. 그리고 그 밑에 둥근 물체 때문에 바닥에 빛이 도달하지 못해서 바닥에 둥글게 어두워진 부분도 있는데, 이 부분이 바로 그림자이다. <그림1-1>을 보면 조금 더 이해가 쉬울 것이다. 그림자의 역할은 물체에 존재감을 부여하는데 있다. 물체가 공간안에 존재하며 어느곳에 위치하는지에 대한 정보를 그림자가 나타내어 준다.

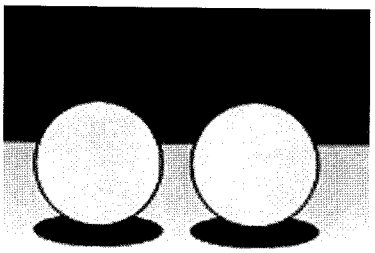
아래의 그림들을 보며 이 같은 특성을 확인해보자.

<그림2>를 보면 둥근물체가 두개 있는데, 이것은 공간안에서 존재감이 전혀 없다. 공중에 떠 있는지 바닥에 붙어있는지 알 수가 없을 뿐 아니라, 종이로 오려놓은 것인지 아니면 야구공처럼 부피가 있는 것인지도 감을 잡을 수가 없다. <그림2-1>은 이것들에게 존재감을 부여하기 위해서 그림자를 넣어놓은 그림이다. 비로소, 야구공처럼 부피가 있는 공 두개가 바닥위에 놓여 있다는 것을 알 수 있다. 그렇다면, 물체는 가만히 있는데 그림자만 옮기면 어떻게 될까? <그림2-2>에서는 두번째 물체의 그림자 위치를 약간 아래로 옮겨 보았다. 그러자 두번째 물체는 조금 더 앞으로 나와 있으며 공중에 떠 있는 것처럼 보인다. 그림자를 옮기자 물체의 존재에 대한 정보가 변경된 것이다. 이처럼 그림자는 물체의 존재감을 부여하기 때문에 장면을 연출하는데 있어서 아주 지대한 영향을 미치는 중요한 요소이다. 가끔 게임 개발자들 중 그림자의 중요성을 망각하는 경우가 많은데, 그림자가 제대로 표현되지 않은 게임들을 보면 물체들이 왠지 불안하고 어수선하며 장면 안에 안착해 있지 못한 모습을 보이게 된다.

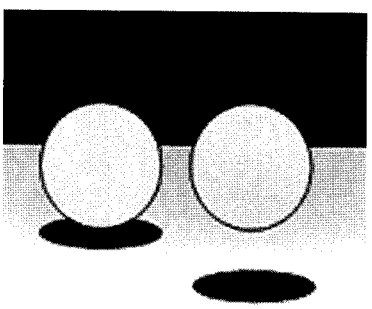
게임에서 그림자에 대한 처리는 그래픽스엔진에서 담당하게 된다. 그러므로 그래픽스엔진의 설계단계에서부터 어떻게 그림자를 처리할지 심사숙고 해야만 할 것이다. 현재 PC 온라인 게임에서 가장 많이 사용되는 그림자 처리 기술은 '투영 그림자(Projected Shadow)' 이고 콘솔(XBOX360, 플레이스테이션3 같은 비디오 게임기를 말함) 및 고사양 PC를 겨냥한 온라인게임에서는 '깊이 그림자(Depth Shadow)' 와 '스텐실 그림자(Stencil shadow)' 를 사용하고 있다. 그림자 생성 기술은 크게



〈그림 2〉



〈그림 2-1〉



〈그림 2-2〉

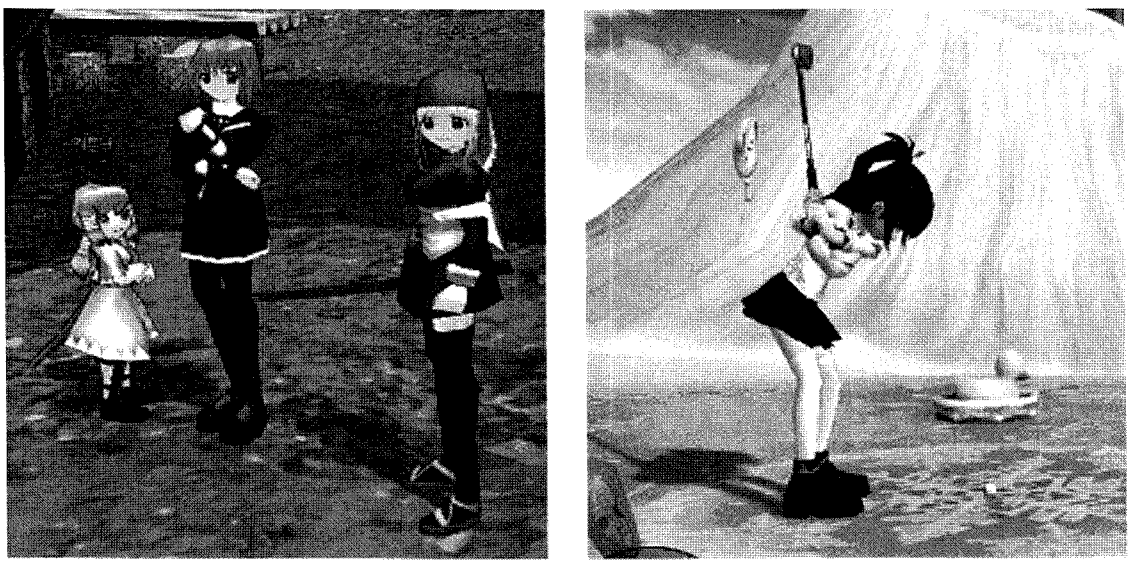
'그림자 맵 (Shadow map)' 과 '그림자 볼륨 (Shadow Volume)' 이렇게 두부류가 있는데, '투영 그림자' 와 '깊이 그림자' 는 '그림자 맵' 부류에 속하며, '스텐실 그림자' 는 '그림자 볼륨' 부류에 속한다. 이 글을 통하여 이 세가지 실시간 그림자 처리 기술에 대한 소개와 2008년도 PC온라인 게임에서 많이 사용하게 될 그림자 기술을 전망해 보도록 하겠다.

II. 투영그림자 (Projected Shadow)

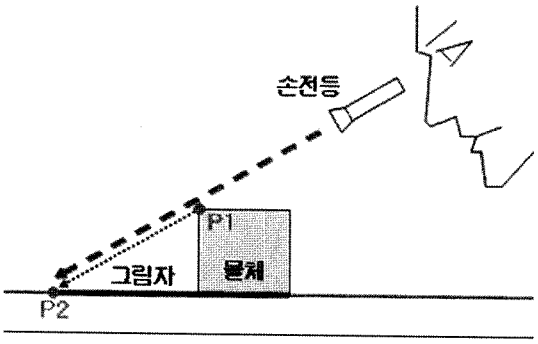
투영그림자는 PC온라인 게임에서 가장 많이 사용되고 있는 실시간 그림자 처리 기술이다. PC온라인 게임은 특성상 저사양 PC에서도 구현이 가능하도록 제작되어야 하는데, 투영 그림자는 단지 '정점 셰이더(Vertex shader) 1.1' 만으로도 구현이 가능하며, 비교적 처리 비용 대비 품질이 좋은 편이어서 현재 가장 많이 사용되고 있다. 참고로, '정점 셰이더 1.1' 만을 사용한다는 말은 거의 모든 3D그래픽카드에서 구현이 가능하다는 뜻이다. <그림3>은 투영그림자를 이용하여 실시간 그림자를 구현한 PC온라인 게임 '마비노기' 와 '팡야' 의 스크린 샷이다.

자! 그럼, 이렇게 멋지게 그림자를 표현해주는 투영 그림자의 원리를 알아보자.

앞에서 그림자란 "물체가 빛을 가려서 그 물체의 뒷면에 드리워지는 검은 그늘" 이라고 했다. 만약 빛이 있는 쪽에서 물체를 바라보면 어떻게 될까? 정확히 빛의 시작점에서 물체를 바라보면 그림자는 보이지 않는다. 왜냐하면 빛의 시작점에서 바라본 화면에서는 빛을 못 받은 부분이 물체의 뒷부분에 숨어 있기 때문이다. 이것은 쉽게 테스트해 볼 수 있다. 어두운 곳에서 손전등을 켜서 눈의 위치에 맞추고



〈그림 3〉 (왼쪽)넥슨 소프트의 마비노기, (오른쪽)엔트리브 소프트의 팡야



〈그림 4〉

바라보면 그림자가 잘 보이지 않는다는 것을 알 수 있을 것이다. <그림4>는 그 상황을 그림으로 표현해 놓은 것이다.

그림을 보면 바라보는 시점에서 물체의 실루엣 끝부분인 P1과 그림자 실루엣의 끝부분인 P2는 손전등이 바라보는 위치에서는 같은 위치로 보이게 된다. 그러므로 그림자는 물체에 가려서 보이지 않게 되는 것이다. 여기서 잘 생각해 보면 광원의 위치에서 바라본 물체의 실루엣에 포함이 되는 부분이 바로 바닥의 그림자 영역이라는 것을 알 수 있다. 투영 그림자라는 것은 바로 이 원리를 이용한 것이다.

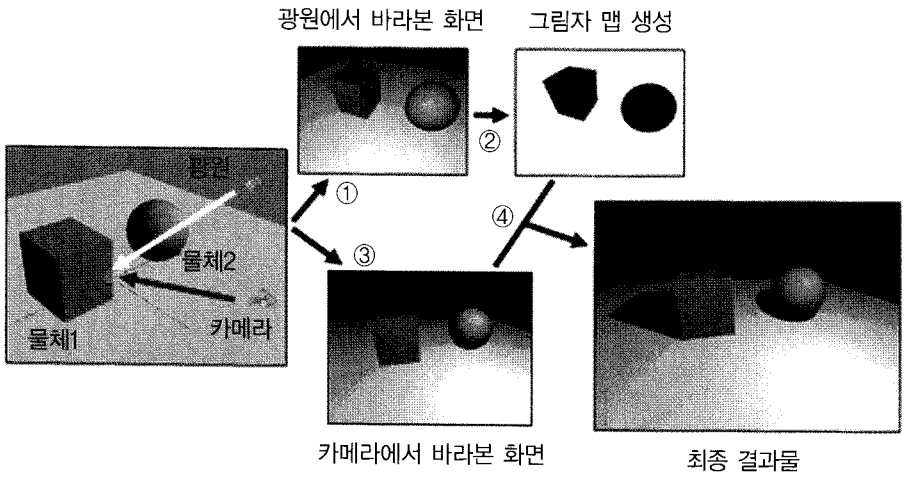
아래 <그림5>는 그래픽스엔진에서 투영 그림자를 처리하는 프로세스를 간단하게 표현해 놓은 그림이다.

그림에서 각 화살표의 번호는 프로세스가 이루어지는 순서를 말하는데 화살표의 번호를 따라가며 어떤 작업이 이루어지는지 정리해 보면 다음과 같다.

① 월드 위치에서 광원의 공간으로 변환을 수행하는 뷰행렬과 프로젝션행렬을 생성한다.

② 생성된 뷰행렬과 프로젝션행렬을 적용하여 그림자를 캐스팅할 물체들을 텍스쳐에 렌더링하여 그림자정보를 저장한다. 이 텍스쳐를 그림자 맵이라고 한다. 이때 바닥은 그림자를 캐스팅하는 물체가 아니므로 렌더링에서 제외한다.

③ 뷰행렬과 프로젝션행렬을 다시 월드 공



〈그림 5〉

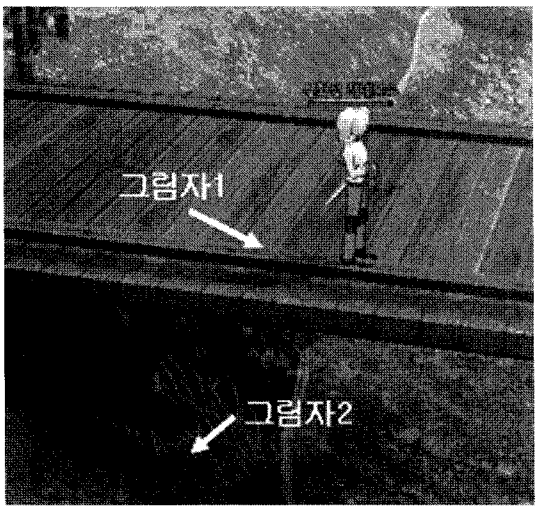
간으로 변환한다.

④ 그림자를 캐스팅한 물체들은 별다른 수행 없이 렌더링하고, 그림자가 드리워져야 하는 바닥은 각 버텍스들이 그림자맵의 어느부분에 해당하는지를 계산하여 그림자맵 좌표를 얻은다음, 그림자맵을 바닥의 디퓨즈맵과 혼합시켜 렌더링 한다. 그러면 바닥에 그림자가 드리워진 최종 결과물을 얻을 수 있다.

다음으로 투영그림자의 단점을 알아보자.

앞에 <그림3>을 다시 자세히 보면 알 수 있는 것이 있는데, 그림자가 바닥에는 생기지만 목덜미에 생겨야할 얼굴그림자나, 겨드랑이에 생겨야 할 팔그림자, 다리에 생겨야할 치마그림자가 생기지 않는다는 것을 알 수 있다. 자기 자신에 의해 생긴 그림자가 자기 자신에게 드리워지는 것을 소위 '셀프섀도우(Self Shadow)'라고 하는데, 투영 그림자에서는 '셀프섀도우'가 지원되지 않는다. 그 이유는 투영그림자에서 만들어지는 그림자맵은 그림자의 실루엣 정보만 있을 뿐, 물체보다 그림자가 앞에 있는지, 또는 뒤에 있는지를 결정할 수 있는 정보가 없어서, 그림자를 캐스팅한 물체 스스로에게 그림자 맵을 적용하면 온통 검게 표시되어 버리기 때문이다. 그래서 그림자를 캐스팅한 물체는 스스로에게 그림자를 드리울 수가 없다.

투영그림자의 단점은 셀프 섀도우가 지원되지 않는다는 점 외에도 물체를 관통하여 그림자가 생기는 현상도 있다[1]. <그림6>은 '마비노기'의 스크린 샷이다. 그림자가 다리위에 생겼는데도 불구하고 다리를 관통하여 지면에서 또 하나의 그림자가 생기는 것을 확인할 수 있다. 이 때문에 투영그림자를 사용할 때는



〈그림 6〉

지형과 지형위에 있는 물체들의 배치에 신경을 많이 써야 한다.

III. 깊이 그림자(Depth Shadow)

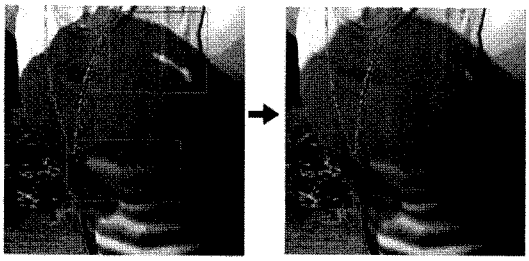
'깊이그림자'는 '투영그림자'의 단점을 보완한 좀 더 발전된 형태로 볼 수 있다.

깊이 그림자의 전반적인 프로세스는 투영그림자와 같지만 그림자맵을 생성할때 실루엣 정보와 함께 추가적으로 광원으로부터의 거리 즉, 다시 말해 광원의 시점에서 볼때 물체의 깊이 값을 저장한다는 점이 다른점이다. 렌더링시 그림자를 생성할 때 이 깊이 값을 비교하여 그림자와 물체간에 앞뒤 순서를 알아낼 수 있어서 셀프 섀도우를 구현할 수 있고, 그림자가 물체를 관통하여 생기는 문제도 사라지게 된다. 깊이 그림자를 구현하기 위해서는 세이더모델2 이상을 지원하는 그래픽카드가 필요한데, ATI Radeon 9500, Nvidia

〈그림 7〉 SEGA사의 버추얼파이터5

Geforce FX 5200 이상이면 구현이 가능하다. <그림7>은 깊이 그림자를 이용해 그림자를 구현한 플레이스테이션3용 게임인 버추얼파이터5 이다. 이 그림에서 보면 어깨와 가슴에 각각 자신의 얼굴과 팔의 그림자 드리워진 것을 확인할 수 있다.[2]

그림자맵을 이용해서 그림자를 생성하는 모든 기술들에서 일어나는 공통적인 문제점이 있는데, 그것은 일명 그림자맵 알리어싱(Shadow Map Aliasing)이라고 불리는 그림자의 계단현상문제다.[3] 이 문제가 생기는 원인은 그림자맵을 생성하는 텍스쳐의 해상도가

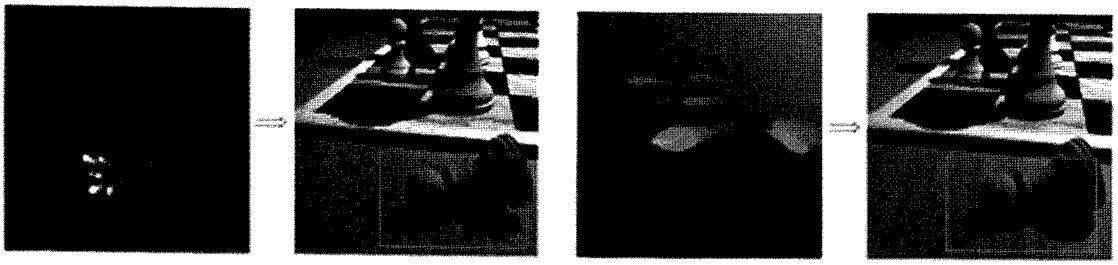


계단현상이 일어난 화면 Blur 처리로 계단현상을 줄인 화면

〈그림 8〉

유한하기 때문에 생기는데, 해상도가 큰 텍스쳐를 사용할 경우 이 현상은 약간 줄일 수 있지만 완전한 해결책은 아니다. 알리어싱을 완화하는 방법을 크게 두가지로 생각해 볼 수 있는데, 하나는 그림자를 생성할 때 뭉개지도록 Blur 필터링을 해주는 것인데, 이 때 사용되는 필터링 기술로는 PCF(Percentage closer filtering)라는 것이 유명하다.[4] <그림9>를 보면 버추얼파이터5에서 알리어싱 문제를 완화시키기 위해서 그림자를 생성할때 Blur처리를 하고 있는 것을 볼 수 있다.

알리어싱 문제의 또 다른 완화 방법으로 '원근 그림자 맵(PSM : Perspective Shadow Map)'이라는 기술이 있다.[5] 카메라로부터 가까이 있는 물체 일수록 그림자가 세밀해야 보기 좋



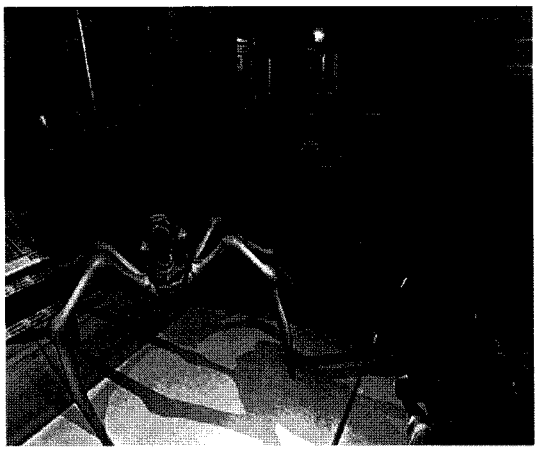
표준 그림자 맵 원근 그림자 맵

〈그림 9〉

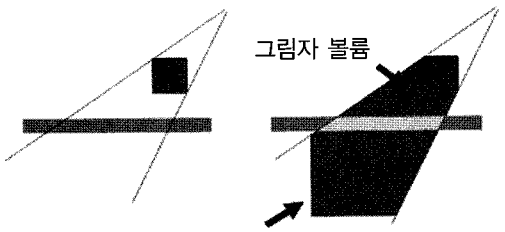
다. 반면 멀리 떨어져 있는 물체는 비교적 그림자가 세밀하지 않아도 눈에 크게 거슬리지 않는다. 이 점을 착안하면 만들어진 것이 '원근 그림자맵' 기술이다. 그림자 맵을 생성할 때 카메라 시점의 원근을 적용해서 가까이 있는 물체의 그림자가 더 세밀해지도록 하는 기술이다. 이 밖에도 '라이트 공간 원근 그림자맵(LSPSM : Light Space Perspective Shadow Map)'[6], '대수계 그림자 맵(LOGSM : Logarithmic Shadow Map)'[7] 등 그림자맵에서 알리어싱 문제를 해결하기 위한 새로운 기술들이 계속 연구되고 있지만, 그림자맵에서 알리어싱이 전혀 없는 완전히 깨끗한 그림자를 얻을 수 있는 방법은 아직까지 없다.

IV. 스텐실그림자(Stencil shadow)

이 기술은 스텐실버퍼라는 메모리 공간을 이용하기 때문에 흔히 스텐실 그림자라고 불리어 진다. 스텐실 그림자는 Id Soft사의 둠3에서 본격적으로 사용하기 시작하면서 많이 알려지게 된 그림자 기술이다. 깊이 그림자와 마찬가지로 셀프 섀도우를 지원한다. 그러나 그림자 맵 기술 대신 그림자 볼륨이라는 기술을 사용하는데, 이 기술의 장점은 그림자 맵에서 해결할 수 없었던 알리어싱 문제를 해결할 수 있다는 점이다.

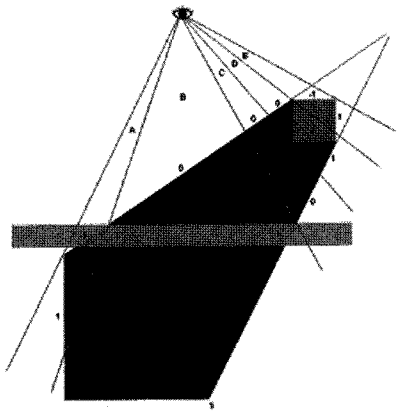


〈그림 10〉 Id Soft사의 둠3



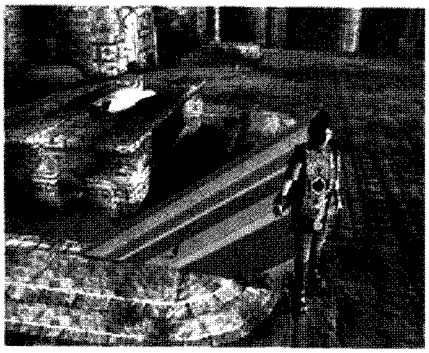
광원방향으로 늘어난 폴리곤

〈그림 11〉

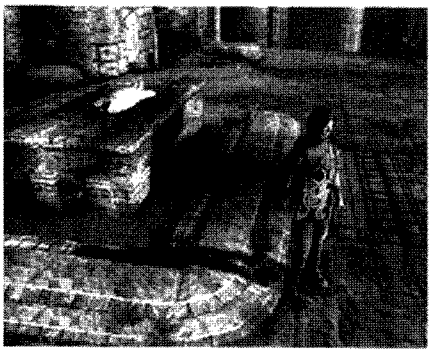


〈그림 12〉

그림자 볼륨의 구현 방법은 렌더링 될 물체의 모서리에 가상의 폴리곤을 끼워 넣은 다음 광원의 빛을 받지 못하는 부분의 가상 폴리곤을 광원 방향으로 길게 늘어뜨린다. 이때 가상 폴리곤에 의해서 생성되는 공간을 그림자 볼륨이라고 한다. 그림자 볼륨을 스텐실 버퍼에



그림자 볼륨을 표시한 화면

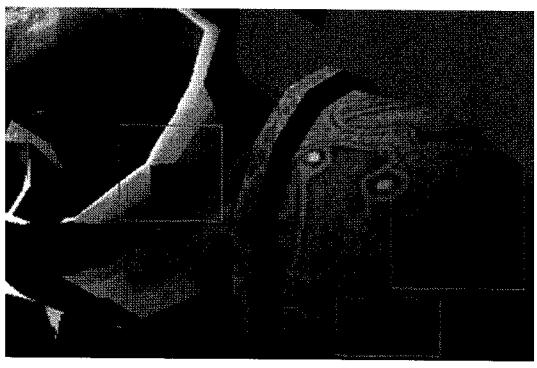


스텐실그림자를 이용해 그림자를 생성한 화면

〈그림 13〉 소니 엔터테인먼트의 완다와 거상

렌더링하면서 카메라의 '앞쪽을 향하는 폴리곤은 +1,뒤쪽을 향하는 폴리곤은 -1'을 해준다. 그러면 그림자가 그려져야 할 영역은 뒤쪽 폴리곤 렌더링시 깊이버퍼 체크 실패로 그리기가 실패해 -1을 해줄 수 없게 되어 스텐실버퍼에 1로 남게 된다. 이렇게 생성된 스텐실 버퍼에 1로 마킹된 영역만 어둡게 처리해 주면 그림자가 생성되게 된다. <그림12>는 이 과정을 도식화한 것인데 그림에서는 A, B, D 영역에 그림자가 그려지게 된다.

스텐실 그림자를 그릴 때 그림자 볼륨을 생성할 가상의 폴리곤 때문에 원래 물체의 폴리곤보다 약 5배 정도 렌더링 할 폴리곤이 늘어나게 되어 심각한 부하가 발생된다. 그래서 스텐실 그림자는 저사양 PC에서 구현하기에는 무리가 있는 기술이다. 그러나 이러한 성능부하를 조금 줄일 수 있는 트릭이 있는데 그것은 바로, 폴리곤수가 적은 그림자용 메시를 별도로 사용하는 방법이다.



〈그림 14〉

플레이스테이션2용 게임인 '완다와 거상'이 이러한 트릭을 이용하여 열악한 실행환경에서도 스텐실 그림자를 구현해 냈다.[8] 스텐실 그림자는 알리어싱 문제는 없지만 또 다른 문제를 하나 가지고 있다. 그것은 폴리곤을 이용하여 그림자를 만들기 때문에 물체의 어둠이 시작되는 부분에 날카로운 그림자 간섭이 일어난다는 점이다.[9] <그림4>를 보면 그런 현상이 일어난 부분들이 빨간색 박스로 체크되어 있다.

이 현상은 렌더링 될 물체의 폴리곤수가 적으면 적을수록 두드러지게 나타나게 된다.

'완다와 거상'의 경우 적은 폴리곤수에 따른 이러한 현상을 극복하기 위하여 그림자의 농도를 아주 약하게 하고 Blur처리를 함으로써 해결하고 있다.

스텐실 그림자는 이러한 단점에도 불구하고 그래도 현재까지 나온 기술 중에 가장 좋은 그림자 품질을 얻을 수 있는 기술이다. 시스템의 성능만 받쳐준다면 말이다.

V. 2008년도 PC온라인 게임에서 유행할 그림자 처리 기술에 대한 전망

2008년은 PC온라인 게임에서의 주력 그림자 처리 기술이 대대적으로 바뀌는 전환점이 될 것으로 필자는 생각한다. 투영그림자는 그동안 대부분의 3D그래픽카드에서 구현이 가능하다는 장점 덕분에 많은 사랑을 받아 왔지만, 이제는 유저들의 PC사양도 많이 향상되어서 셰이더모델2에서 구현 가능한 조금 더 높은 품질의 그림자도 소화해 낼 수 있을 것으로 생각된다. 참고로 셰이더모델2를 지원하는 그래픽카드는 2003년도에 처음 선보여 벌써 5년이라는 시간이 흘렀다. 그렇다면 투영 그림자를 교체할 다음 기술로 깊이 그림자와 스텐실 그림자를 생각해 볼 수 있는데, 그래도 아직까지 스텐실 그림자는 PC온라인 시장에서는 시기상조인 듯하다.

예측해 보건데, 2008년도에는 '원근 그림자맵'을 사용하는 '깊이 그림자'로 많은 게임들이 개발되지 않을까 생각해 본다. 이 방식은 셰이더모델2를 지원하는 가장 처리부하 대비 효과가 뛰어난 기술이기 때문이다. 중국이나 필리핀 같이 PC사양이 우리나라보다 떨어지는 나라까지도 시장으로 생각한다면 투영 그림자 와 깊이 그림자를 둘 다 지원하는 것도 좋은 선택일 것 같다.

2008년에 좀 더 향상된 그림자 품질로 개발되어질 PC온라인 게임들을 모두 함께 즐거운 마음으로 기다려 보도록 하자.

참고문헌

[1] Takashi Imagire, DirectX9 셰이더 프로그래밍, 한빛미디어, 352 page, 7월, 2004년

[2] http://www.watch.impress.co.jp/game/docs/20061025/3dvf5.htm

[3] DirectX9 SDK 'Shadow Map' Reference

[4] Wolfgang Engel, ShaderX2 DirectX9 셰이더 프로그래밍, 정보문화사, 207page, 9월, 2004년

[5] www.watch.impress.co.jp/game/docs/20041021/3dmark1.htm

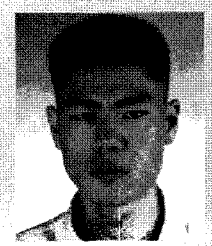
[6] www.watch.impress.co.jp/game/docs/20050929/3dinis.htm

[7] http://gamma.cs.unc.edu/logsm/

[8] http://www.watch.impress.co.jp/game/docs/20051207/3dwa.htm

[9] DirectX9 SDK 'Shadow Volume' Reference

저자소개



김 도 형

2004년 2월 부천대학 전산학과 졸업 (전문학사)
2001년 10월-2005년 7월 (주)NariSoft. Project Manager
2005년 3월-현재 한국폴리텍 서울강서대학 컴퓨터게임과 외래교수
2006년 2월-2007년 3월 (주)ActozSoft. 클라이언트 프로그램 개발 팀장
2007년 4월-현재 (주)KBOnline. 엔진 개발 팀장

주관심 분야 : 컴퓨터 그래픽스