論文
3-2-11

영역별 데이터 추출에의한 효과적인 그림자 표현 (Shadow Generation By Extracted Point data on Subregion)

고 찬* 강 정 호**
(Chan Koh) (Jeong-Ho Kang)

요 약

3차원 공간에 물체를 입체감있게 표현하는 것은 중요한 일이다. 물체를 사실적으로 보이게 하기 위해서는 원근감, 입체감, 물체의 재질, 그림자, 빛의 세기 등 여러 요소가 고려되어야 한다. 그 중 그림자는 물체의 사실적 표현에 중요한 역할을 한다. 그림자를 그리기 위한 방법에는 반복적인 수많은 연산을 해야하고 이것은 많은 처리 시간을 필요로한다. 즉 수많은 점들에 대한 그림자 영역 판단과 빛의 세기 판단 등을 해야하고 이것으로 인해 시간이 많이 걸리게 된다. 본 논문에서는 물체에 대한 사실적인 표현을 어느 정도 유지하는 범위내에서 그림자를 영역별로 분리해 빠르게 계산하여 표현해 주는 방법을 제시한다.

ABSTRACT

To describe an object realistically is the most important thing in the 3-dimentional space in computer graphics. It is not enough with one or two factors. Several factors should be considered such as perspective sense, cubic effect, material of an object, shadow, strength of light, etc. A shadow algorithm plays an important part in the realistic description of an object. There are many methods to describe a shadow, but it means numerous repeated operations to describe a shadow and it needs much time. We seperate the shadow part and calculate the strength of light for numerous points and it takes much time. This thesis presents a new method to describe a shadow quickly by separating categories maintaining a realistic description of an object.

1. 서론

1.1 연구 목적 및 배경

컴퓨터가 우리의 생활 속에 점점 깊이 파고들면서 가상현실이나 광고, 게임, 애니메이션 등 많은 분야에서 컴퓨터 그랙픽스가 응용되고 있다. 오늘날 이처럼 컴퓨터 그래픽스가 많은 분야에서 사용되는 가장 큰 이유는 물체를 모니터 위에 사실감 있게 표현해 내기 위해서이다. 이렇듯 사실감 있는 물체를 모니터 위에 표현해 내는 일은 한 두 가지 요소만 고려하고서는 거의 불가능하고 원근감, 입체감, 물체

* 정회원 : 서울산업대학교 컴퓨터공학과 교수
** 정회원 : 숭실대학교 박사과정
논문접수 : 2002. 2. 4.
심사완료 : 2002. 2. 23.
※ 이 논문은 서울산업대학교 교내 학술연구비 지원에 의하여 연구되었음.

의 재질, 그림자, 빛의 세기 등 많은 요소를 고려해야 함으로써 가능해진다.

앞에서 언급한 요소들 중에 그림자의 표현(Shadow Generation)[1]은 3차원 컴퓨터 그래픽에서 수학적인 계산에 의해 화면에 표시된 물체에 적절한 색깔 및 밝기를 부여하여 물체의 질감 및 입체감을 나타내는 것으로서 많은 시간과 계산을 요하는 작업이다. 또한 Shading의 경우 전적으로 프로그램 내부에서의 계산에 의존하게 되므로 작업자가 Object에 걸맞은 질감과 빛을 잘 파악해서 선택적으로 사용해야만 좋은 결과물을 얻을 수 있다. 그림자의 표현 기법을 알기 위해서는 명암처리(Shading) 알고리즘(Scan - line Algorithm, flat shading, Gouraud shading, Phong shading)과 그림자 생성(Shadow Generation) 알고리즘(Scan - line, 폴리곤 회전, Shadow Volume)에 대해 알아두어야 할 필요가 있다.

본 논문에서는 Scan - line 방법을 바탕으로 그림자를 표현하는데 걸리는 시간을 줄일 수 있는 방법을 소개하고자 한다.

어떤 영상이 광원으로부터의 그림자 생성시 모든 점에 대해 음영처리를 하고 있어 많은 시간이 소요되고 있다. 그래서 시간과 연산을 줄여보고자 그림자 생성시 완전 음영부와 남은 영역[2]으로 구분하고 또 다시 남은 영역을 각각의 트랙으로 구분한 후 각 영역의 대표값만 계산하여 그림자 생성시켜 본다. 그 결과를 현재 사용하고 있는 방법인 모든 점에 대해 처리한 경우와 비교한다. 그리고 본 논문에서 비교 결과 분석에 의해 효과적인 그림자 생성 알고리즘을 제안하고자 한다.

2. 그림자 생성 알고리즘

2.1 컴퓨터에서 물체의 표현

2.1.1 Scan - Line Algorithm

Digital Image를 모니터에 그려 주기 위해서 대부분의 Software Packages들은 Scan - Line이라고 알려진 Rendering Algorithm을 많이 사용한다. [2] Digital Image는 아주 가느다란 개별적인 Element나 Pixel들로 구성되어 있는데 이러한 것들이 모여 아주 가느다란 각 열을 이루게 된다. Scan - Line Rendering은 Program이 이러한 열을 이루고 있는 각각의 Pixel을 한 열씩 하나 하나 읽어 들인 후 그 pixel의 color 값을 계산하여 나가는 방식을 말한다. 보통 위에서 아래로 같은 방식으로 한 줄씩 계산하여 내려오면서 그림을 구성하는 방식을 취하게 된다. 극단적으로 시간이 많이 걸리는 Image를 Rendering[3] 할 경우 개별적인 라인을 별도의 기계를 이용하여 Rendering 한 후 합쳐주어 완전한 그림을 만들어 주는 것도 가능하다.

2.1.2 Ray - Casting Algorithm

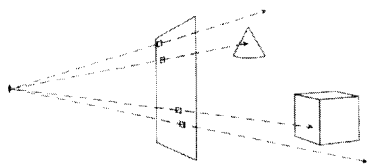
Ray - Casting Algorithm은 물체(Object)를 우리 눈에 보일 수 있도록 물체(Object)나 색깔(Color)을 모니터에 표현해 주는 방법이다. [2][5] 시야(Scene)에서 한 "바라보는 점" 을 떠난 광선(Ray)은 바라보고 있는 시야(Scene)의 다른 지점을 향하여 뻗어 나가게되고 이 광선(Ray)은 물체(Object)에 부딪히게 되면 부딪힌 지점의 색깔을 계산하게 된다. 즉 Pixel의 색깔이 광선(Ray)에 의해서 계산되어지는 것이다. 첫 번째 Pixel에 대한 이러한 계산을 마치게 되면 Ray - Casting Algorithm은 다음 Pixel을 향해 동일한 작업을 반복하게 된다. 이러한 동일한 과정을 거쳐 Image 전체를 계산하게 된다. [그림 1]은 한 개의 광선(Ray)이 각각의 Pixel을 어떻게 통과하는지의 모습을 보여 주고 있다. 제일 상단과 하단에서 뿌려지게 되는 광선(Ray)은 아무 Pixel도 통과할 것이 없으므로 아무런 색깔도 가지지 못한다.

---

1) Shadow Generation은 Shading이란 단어와 혼동하기 쉬우니 유의해야 한다.
- Shadow Generation - 그림자 생성(표현)
- Shading - 명암 처리(표현)

2) 남은 영역은 그림자에서 완전 음영부를 제외한 부분이다.

3) 컴퓨터그래픽, 특히 3차원 그래픽에서 화면에 그린 물체의 각 면에 적절한 색깔을 입히고 여러 가지 효과를 가하여 화상의 실제감을 나타내는 작업이다.

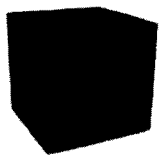


[그림 1] 광선(ray)이 각각의 Pixel을 통과하는 모습
[Fig. 1] Ray penetration each pixel

2.2 빛의 반사와 명암

2.2.1 Faceted Algorithm

Faceted Algorithm은 명암을 만드는 가장 단순한 방식으로 평평한 surface 부분을 모두 같은 색깔로 처리하는 방식이다. [3] 이 방식은 [그림 2]와 [그림 3]에 나타나는 것과 같이 물체(object)의 분할된 면이 한 가지 색깔만을 가지게 된다.



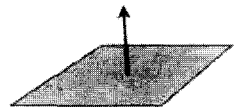
[그림 2] Faceted Algorithm에 의해 표현된 물체 1
[Fig. 2] Represented object 1 by faceted algorithm

[그림 3] Faceted Algorithm에 의해 표현된 물체 2
[Fig. 3] Represented object 2 by faceted algorithm

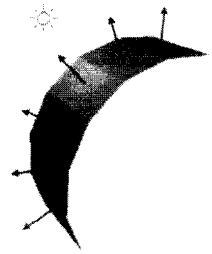
Faceted Shading은 매우 빠른 Rendering이 가능하고 이러한 이유로 많은 3D Package에 적용하고 있고 다른 모든 Shading Algorithm에 중요한 개념으로 자리잡고 있다.

Surface(물체 표면)의 색깔을 계산할 때에 Shading Algorithm은 그 Surface(물체 표면)의 각도에 따른 어두움 밝기를 계산하여 준다. 그 계산을 위해 Surface(물체 표면)의 방향을 Surface Normal라고 부른다.

Flat Surface의 경우 하나의 Surface Normal이 Surface(물체 표면)의 방향을 결정짓게 된다 ([그림 4]). [그림 5]에 나타난 것과 같이 여러 개의 Surface(물체 표면)가 연결되어 지고 Surface Normal은 이러한 Surface(물체 표면)에 개별적으로 적용되어 진다. 따라서 이러한 Surface Normal이 빛(Light)과 어떤 각도를 이루느냐에 따라 밝은 정도가 달라지게 된다.



[그림 4] Surface Normal
[Fig. 4] Surface normal



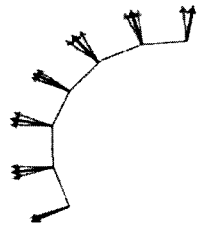
[그림 5] 곡면의 Surface Normal
[Fig. 5] Surface normal of curved surface

평면이 아닌 굴곡이 있는 구 같은 경우는 평면이 없기 때문에 모든 점에 대한 Surface Normal을 계산해야 하는데 이것은 너무나 많은 시간을 필요로 한다. 그래서 Curve Surface를 사용하는 경우 Rendering 전에 Polygon 방식으로 전환하여 계산하여 주는 것이 일반적이다. Curve Surface를 polygon 방식으로 바꾸어 계산하게 되면 Rendering Program은 각각의 Polygon의 하나의 Surface Normal에 대해서만 계산하면 되므로 시간을 절약할 수 있고 실시간으로 보여주는 것도 가능하게 되었다.

이러한 Polygon 방식의 경우 문제를 포함하고 있는데 Polygon으로 나누어졌기 때문에 어찌 되었던 Polygon의 모양이 드러난다는 것이다. ([그림 3]).

2.2.2 Gouraud Shading Algorithm

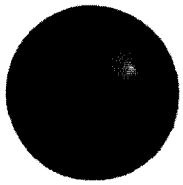
Gouraud Shading Algorithm은 Henri Gouraud에 의해 창안되었고 Polygon의 Surface Normal을 인위적으로 조절하여 Polygon 간의 경계를 부드럽게 보이도록 만들어 준다. [4] [그림 6]은 [그림 5]에서 본 것 같은 Curve Surface를 Polygon 방식으로 옆에서 바라 본 모습이다. 그림에서 보게 되면 각각의 Vertex(점)는 두 개의 Polygon이 만나는 지점에 위치하고 있고 두 개의 Polygon에 의해, 두 개의 Surface Normal이 존재하게 되는 것을 희미한 색의 화살표로 나타내어 주고 있다. 이때 Gouraud Shading Algorithm은 주어진 Vertex(점)에서 Surface Normal의 평균치를 구하고 그 평균치는 검은색 화살표로 그려지게 된다. 그러므로 인해 Faceted Algorithm보다 완만하게 바꾸어져 보이게 하여 준다([그림 7]).



[그림 6] Gouraud Algorithm에서 Surface Normal
[Fig. 6] Surface Normal by Gouraud algorithm

[그림 7] Gouraud Algorithm에 의해 표현된 곡면
[Fig. 7] Curved surface by Gouraud algorithm

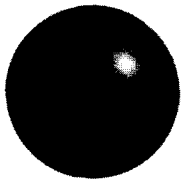
이러한 Grourand Shading 방식은 속도도 빠른 편이지만 [그림 8]과 같이 점광원에 대한 사실적인 표현이 약간 뒤틀려 보이게 되는 경향이 있다.



[그림 8] Gouraud Algorithm에 의해 표현된 물체
[Fig. 8] Represented object by Gouraud algorithm

2.2.3 Phong Shading Algorithm

Gouraud Shading Algorithm의 위와 같은 한계를 극복한 것이 Bui Tuong Phong에 의해 개발된 Phong Shading Algorithm이다. [4][5][7] Phong Shading은 Gouraud Shading이 평균을 구하는 것과는 달리 Vertex의 Surface Normal을 계산하여 이 Surface Normal로 색깔을 직접 계산한다. 이러한 방식은 보다 정확한 색깔과 highlight의 표현이 가능하고 Polygon 간의 구분 같은 경계도 보이지 않게 해 준다([그림 9]). 하지만 질이 좋아지게 되면 Rendering 시간은 더 늘어나게 된다.



[그림 9] Phong Algorithm에 의해 표현된 물체
[Fig. 9] Represented object by Phong algorithm

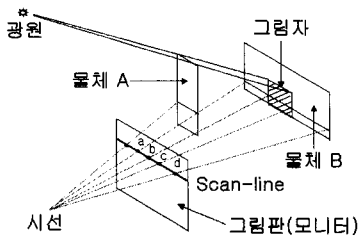
2.3 그림자 영역의 판단

2.3.1 Scan - line

Appel(1968), Bouknight & Kelly(1970)에 의해 개발되었으며 위에서 설명한 Scan - line Algorithm과 한 열씩 하나 하나 읽어 들여서 처리한다는 면에서 유사하다. [5][6]

먼저 읽어들인 line에 대해 Scene에서의 경계인 Vertex들을 찾고 그 Vertex 사이 영역에 그려주어야 할 것이 Polygon인지 그림자인지를 판단하는 방법이다.

[그림 10]에서는 어떻게 경계의 Vertex를 찾고 그림자 여부를 판단하는지를 보여주고 있다.

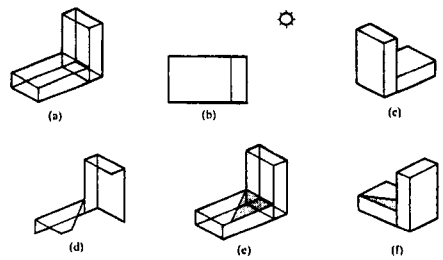


[그림 10] Scan - Line에 의한 그림자 영역 판단
[Fig. 10] Decision of shadow part by scan-line

① 영역 a에서 polygon A는 보이므로 그려준다.
② 영역 b에서 polygon B는 보이므로 그려준다.
③ 영역 c에서 polygon A의 그림자에 의해 가려는 명암을 주어서 그려준다.
④ 영역 d에서 polygon B는 보이므로 그려준다.

2.3.2 Polygon 회전에 의한 그림자 유도

Atherton et al.(1978)에 의해 개발되었으며 다음과 같은 알고리즘에 의해 수행된다. [6]

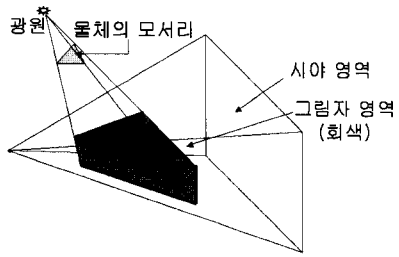


[그림 11] Polygon 회전에 의한 그림자 영역 판단
[Fig. 11] Decision of shadow part by rotation of polygon

(a) 원래의 상태
(b) 바라보는 시선의 좌표계(a,b,c)를 빛이 비추는 좌표(x,y,z)로 이동
(c) 보이지 않는 면을 제거
(d) 시선의 좌표(x,y,z)를 다시 원래의 좌표(a,b,c)로 이동
(e) 제거 되었던 면을 다시 복원
(f) 그림자 영역 생성

2.3.3 Shadow Volume

1977년 Franklin Crow에 의해 창안되었으며 우리가 모니터에 표현할 수 있는 범위인 View Volume 안에서 그림자 영역인 Shadow Volume을 계산해내서 그 영역을 그림자 영역으로 처리를 하는 방법이다. [6]



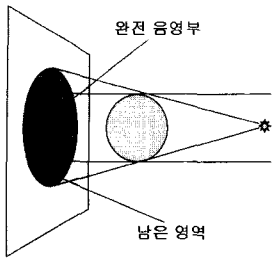
[그림 12] Shadow Volume에 의한 그림자 영역 판단
[Fig. 12] Decision of shadow part by shadow volume

3. 영역 추출점에 의한 그림자 생성 알고리즘

본 논문에서는 그림자 생성시 완전 음영부와 남은 영역으로 구분하고 남은 영역을 각각의 트랙으로 구분한 후, 각 영역에 대표값만을 계산하여 그림자를 생성시켜 본다. 그리고 나서 그 결과를 현재 사용되고 있는 방법인 Scan - Line 알고리즘과 논문에서 새로이 제시된 방법을 비교한다. 비교 결과 분석에 의해 효과적인 그림자 생성 알고리즘을 제안하고자 한다.

3.1 영역 구분 방법 제안

그림자 영역을 빛의 세기를 계산할 필요가 없는 완전 음영부와 그 이외의 남은 영역부분으로 구분하도록 방법을 제안한다. [12] [13]



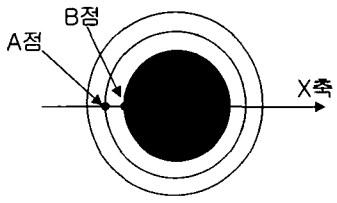
[그림 13] 완전 음영부와 남은 영역
[Fig. 13] Shading part and remain part

3.2 효과적인 그림자 영역 분할 방법 제안

남은 영역을 5트랙, 10트랙, 20트랙, 40트랙 등 여러 경우의 트랙으로 나누고 각각의 그림을 비교하여 어느 정도의 분할이 최적의 영역 분할로 가장 적당한지를 결정한다.

3.3 트랙별 추출점 선정, 트랙별 추출점의 계산

트랙별 추출점은 영역의 X축 상의 시작 반지름과 끝 반지름의 중간 값을 계산하고 그 좌표 값을 추출점으로 선정한다. 이렇게 선정된 영영별 추출점들의 빛의 세기를 계산하고 해당 영역 전체에 적용하는 방법을 제안한다.



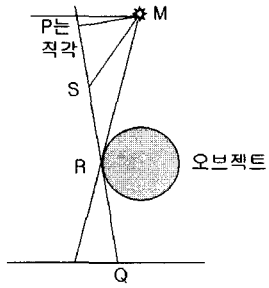
[그림 14] 영역별 추출점을 선정하는 방법
[Fig. 14] Decision of feature extracted point by each part

3.3.1 영역별 빛의 세기 계산

영역내의 선정된 추출점의 빛의 세기를 다음의 공식의 의해 계산하고 적용한다. [14]

빛의 세기 = $M / (c^2+k^2)(a+b-k)^2$

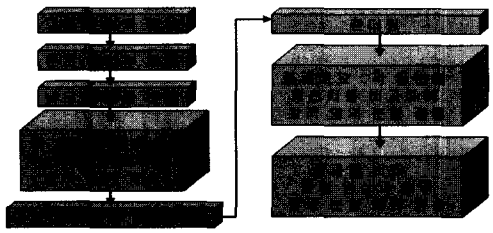
- a : P와 R까지의 거리
- b : R과 Q까지의 거리
- c : M과 P까지의 거리



[그림 15] 빛의 세기 계산
[Fig. 15] Calculation of strength of ray

3.4 그림자 생성 알고리즘 제안

위에서 제안한 것을 가지고 shadow generation 알고리즘을 제안한다.



[그림 16] 영역 추출점에 의한 효과적인 그림자 생성 알고리즘
[Fig. 16] Shadow generation algorithm by extracted points

4. 실험 및 결과분석

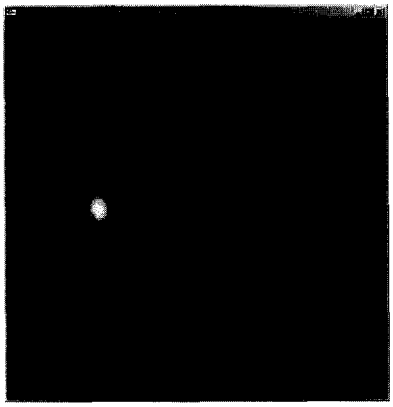
4.1 구현결과

4.1.1 최적의 영역 분할 선택

[그림 17]에서 [그림 20]은 그림자의 남은 영역을

각각 5등분, 10등분, 20등분, 40등분하여 그림자를 그려준 결과로, 적은 분할 영역 내에서 그림자를 효과적으로 표현한 분할은 20등분임을 알 수 있다.

[그림 17]은 남은 영역을 5개의 트랙으로 나누었을 때의 그림이다. 그림에서 보이듯이 나누어진 영역을 쉽게 알아볼 수 있을 정도로 그림자의 효과가 떨어져서 그림자 처리로 적합하지 않았다.

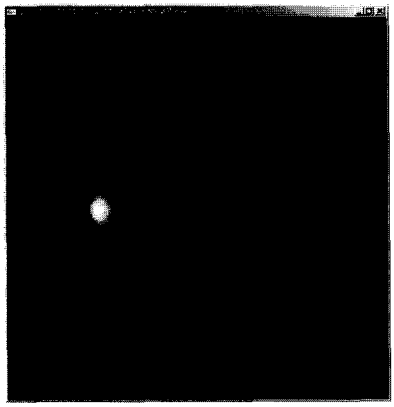


[그림 17] 5트랙으로 분할
[Fig. 17] Divided by 5 tracks

10개의 트랙으로 나누었을 때는 [그림 17]보다는 그림자 효과가 좋아졌지만 그래도 나누어진 영역을 알아볼 수 있을 정도의 그림자 효과가 나타나서 10트랩도 적합하지 않음을 알 수 있다.

[그림 18] 10트랙으로 분할
[Fig. 18] Divided by 10 tracks

20개의 트랙으로 나누었을 때는 [그림 19]에서 보이듯이 트랙으로 나눈 형태가 거의 보이지 않고, 그림자 표현이 아주 효과적으로 그려짐을 볼 수 있다.



[그림 19] 20트랙으로 분할
[Fig. 19 Divided by 20 tracks]

40개의 트랙으로 나눈 [그림 20]은 [그림 19]에서 20개의 트랙으로 나눈 것과 거의 차이가 없는 것을 볼 수 있다.

[그림 20] 40트랙으로 분할
[Fig. 20] Divided by 40 tracks

4.1.2 기존의 방법과의 비교

그림자를 그려주는데 필요한 연산은 빛의 세기 계산에 필요한 연산과 점을 화면에 그려주는데 필요한 연산 두 가지로 나눌 수 있다. <표 1>의 데이터

는 하나의 점을 화면에 표현하기 위해 빛의 세기 계산에 필요한 연산들을 하나로 보고, 빛의 세기가 계산된 점을 화면에 그려주는데 필요한 연산들을 하나로 보았을 때 수행되는 횟수를 비교한 것이다.

<표 1> 기존 방법과 결과 비교

<Table 1> Comparision of result

	빛의 세기 계산에 필요한 연산의 수	점을 화면에 그려주는데 필요한 연산의 수	비교
기존 알고리즘 (Scan -Line)	약 360,000회	약 360,000회	모든 점에 대해 각각 계산
본 논문에서 제시한 알고리즘 (20 트랙 분할시)	20회	약 360,000회	분할한 영역의 대표값을 계산하여 전체에 적용

<표 1>의 결과에 따르면 그림자의 크기가 크면 클수록 연산의 횟수는 더 줄어들게 된다.

본 논문에서 제시한 알고리즘은 <표 1>에서 알 수 있듯이 빛의 세기 계산에 필요한 연산을 획기적으로 줄임으로 인해 그림자를 생성하는데 소요되는 시간을 단축시킬 수 있다. 이러한 소요시간의 단축 알고리즘은 영화나 애니메이션처럼 상대적으로 그림자의 사실감이 사진처럼 보일 필요가 없고 빠른 그림자 계산을 필요로 하는 분야에 활용 가능할 것이다.

5. 결론

그림자를 그리기 위해서는 많은 요소를 고려해야 하고, 많은 연산을 필요로 하는 것이 당연하다. 하지만 그림자를 화면에 표현하는 일은 적용분야에 따라 사실감이나 정밀감을 요구하는 정도의 차이가 있다. 즉 가상현실이나 사진에서는 매우 높은 사실감 정도를 요구할 것이고, 애니메이션이나 영화에서는 사진과 같은 사실감보다는 좀 더 빠른 그림자생성이 필요할 것이다. 그림자에서 사실감보다 빠른 그림자 생성이 더 비중이 있다면 많은 연산을 계속 반복할 것이 아니라 그림자 영역에서 비슷한 부분을 추출하여 동일한 처리를 해도 될 것이다.

그래서 본 논문에서는 그림자를 그릴 때 모든 영역의 점들을 가지고 빛의 세기를 계산한 것이 아니라 완전 음영부 영역과 남은 영역을 분할한 영역에서 각각 임의의 점을 추출하여 빛의 세기에 대한 계산을 함으로써 컴퓨터의 연산을 획기적으로 줄여 보았다.

본 논문은 주로 고정된 물체의 그림자 생성 속도 향상에 대해서 중점적으로 연구되어졌다. 본 연구의 결론을 내리는 시점에서 아쉬웠던 점으로는 대략 두 가지로 간략지울 수 있겠다. 첫째로는 물체가 움직일 때나 크기변화가 있을 때에 대한 연구가 미흡했던 점이고, 둘째로는 두 개 이상의 광원이나 여러 개의 물체에 대한 그림자의 생성 연구가 미흡했다는 점이다.

※ 참고문헌

[1] 고 찬 "그래픽스 소프트웨어", 정익사, 1997, pp.16~25 pp.113~127

[2] URL : http://3dwizard.co.kr/

[3] URL : http://www.be3d.net

[4] URL : http://topcad.co.kr/

[5] Jim Leisy "Computer Graphics", FRANKLIN, BEEDLE, 1994, pp.465~494

[6] Alan Watt, Mark Watt "Advanced Animation and Rendering Techniques", ADDISON WESLEY, 1992, pp.153~172

[7] R. J. Wolfe "3DGraphics", OXFORD, 2000, pp.45~55

[8] URL : http://www.opengl.org/

[9] URL : http://www.hallem.com/

[10] URL : http://dip2k.coco.st/

[11] URL : http://www.xmission.com/%7Enate/index.html

[12] 유관희, "블록 이차 광원으로부터 완전음영부를 구하는 최적 알고리즘", 정보통신진흥원, 1999

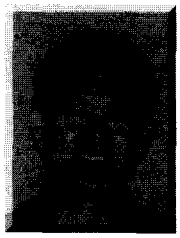
[13] 정은종, 윤경현 "분산광선추적법을 이용한 물체의 움직임 표현", 기술과학연구소, 1995

[14] 남윤영, 이기동 "사실적인 그림자 생성 알고리즘", 한국 정보 과학회(HCI) 2000 학술대회

[15] Edward Angel "Interactive COMPUTER GRAPHICS - A top-down approach with Open GL", Addison-Wesley, 1997, pp.235~245 pp.456~460

[16] Charles Petzold "Programming Windows - Fifth Edition", COMPEOPLE, 2000.7

고 찬



경희대학교 공학사(기계공학)
연세대학교 공학석사
(전산학/컴퓨터그래픽스 전공)
한국과학기술원 박사과정수학
(경영공학/MIS 전공)
서울대학교 경제학박사과정수료
(기술경제및정책 전공)
경희대학교 공학박사
(전자공학/그래픽스영상처리 전공)
미국 North Carolina State University, Post Doc.
해군대학 컴퓨터 교관
(해군장교 복무, 전쟁게임시뮬레이션 담당)
(주)대우엔지니어링 전산실 근무,
System Analyst
미국 Stanford University,
(ISL, CDR연구소) 객원 교수
미국 San Jose State University,
CISE학과 초빙 교수(CISE학과 개설 위원회 위원 및, 교육과정개발 참여)
미국 University of California,Davis
교환교수(전자계산학과, CIPIC)
미국 University of California, Berkeley
연구교수(SIMS)
미국, Global CyberVenture,LLC.,
San Jose, CA, 공동설립자 및 Senior Technical Consultant(현)
미국 ATUNE International,
LLC.,San Jose, CA, 공동설립자 및 Vice President(현)
(사)한국정보통신기술사협회 이사,
부회장(현)
(사)한국컴퓨터게임학회 총무이사,
부회장,(현)
(재)게임종합지원센터 기술자문위원,
기술평가 위원(현)
한국산업인력공단 기술사 시험 출제,
평가위원(현)
삼성SDS, "e-Test 인증 시험"
기술 자문위원(현)
서울산업대학교 컴퓨터공학과 교수(현)
자 격 :
1987. 정보처리기술사
(정보관리 분야, 과학기술처) 자격취득
1999. 공인정보시스템감리인
(한국전산원) 자격취득
전공 및 연구분야 : 영상정보처리,
S/W개발방법론, 정보기술및관리,
정보시스템감리, 정보보호,
정보경제및정책,
http://computer.snut.ac.kr/~chankoh

강 정 호

2000년 서울산업대학교 졸업
(학사)
2002년 서울산업대학교 졸업
(석사)
2002년 숭실대학교 박사 과정
전공분야: 컴퓨터그래픽스,
영상네트워크