실시간 렌더링을 위한 포괄적 가시성 계산

충북대학교 유관희*

1. 서 론

그림 1과 같이 구성된 3차원 가상 공간에서 주어진 시점으로부터 보이는 부분을 화면에 렌더링(rendering)할 경우, 효율적인 화면 생성을 위한 대부분의 그래픽스 시스템은 모든 객체들 중에 시야(view frustum)에 들어오는 객체를 먼저 추출한 후, 이들 객체에 대해 주어진 시점(viewpoint)으로부터 보이는 부분을 계산한다[5]. 예를 들어 그림 1에서 객체 a와 d 등과 같이 시야에 들어오지 않는 객체를 먼저 제거하고, 나머지 객체에 대해 시점으로부터 보이는 부분을 구한다. 그림에서 객체 c는 객체 b에 의해 가려 시점으로부터 보이지 않는다.

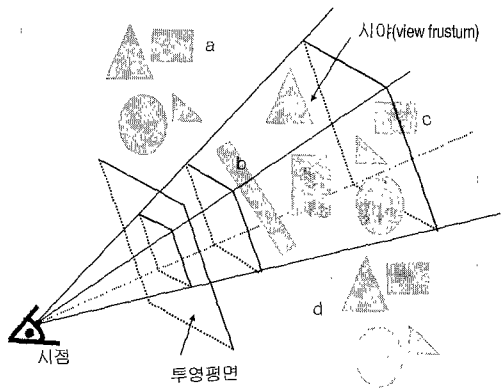


그림 1 그래픽스 시스템에서 렌더링 과정

이를 위한 가장 대표적인 접근 방법이 z-버퍼(z-buffer) 방법으로 시점으로부터 보이는 객체의 표면 깊이 정보를 관리하기 위해 투영평면(projection plane)상에 깊이 버퍼(depth buffer)를 두어 다음과 같이 처리한다. 가상 공간을 구성하는 모든 객체 표면을 하나씩 깊이 버퍼에 투영시켜 대응되는 깊이 버퍼 픽셀(pixel)의 깊이 정보를 비교하여 크면 시점으로부터 보이지 않아 무시되고, 작으면 시점으로부터 보여 깊이 정보 값이 대치된다. 이 작업을 모든 객체의 표면에 대해 실행하여 얻어진 최종 깊이 정보가 시점으로부터 보이는 표면이다[3,5]. 아래 그림 2에서 세 표면 s1, s2, s3에 대해 깊이 버퍼의 (x,y)위치 픽셀에 저장된 최종 깊이 정보는 표면 s1상의 정보이다. 이 방법은 매우 단순하여 대부분의 그래픽 가속기에 구현되어 있다.

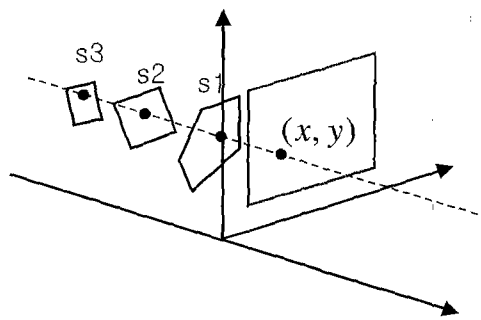


그림 2 Z-버퍼 알고리즘

다음으로 많이 사용되는 방법은 Painter's 알고리즘으로 모든 객체의 표면을 시점으로부터 깊이에 따라 정렬한 후, 맨 뒤의 표면부터 렌더링하는 기법이다[3,5]. 그러나 그림 3 (b)와 같이 모든 표면이 항상 시점으로부터 정렬가능하지가 않을 수 있다. 이 경우 특정한 표면을 잘라 해결하고 있는데, 이 작업의 부담이 너무 크다.

* 종신회원

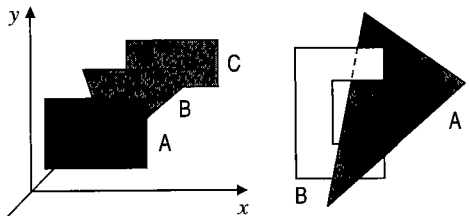


그림 3 Painter's 알고리즘

현재 렌더링의 가속화를 위해 다양한 알고리즘이 그래픽스 하드웨어로 구현되고 있다. 더욱이 더불어 최근 들어 컴퓨터와 그래픽스 하드웨어 기술의 발전과 더불어 사용자들은 3차원 공간상에서 초당 20-30회 이상의 화면 갱신을 요하는 비행 물체 시뮬레이션, 3차원 다자간 네트워크 게임, 가상 공간 walkthrough 등과 같은 고급화된 응용 서비스를 요구하고 있다. 그러나 그래픽 하드웨어 기술의 발달 속도에 비해 사실적인 가상 세계를 요구하는 사용자에 의해 가상 세계를 구성하는 다각형의 수가 몇 십만 개에서 몇 백만 개까지 증가되어, 단지 하드웨어의 기술만으로는 가상세계를 실시간에 렌더링하기란 그리 쉽지 않다. 이를 극복하기 위한 방법으로 모든 객체를 그래픽스 하드웨어에 보내지 않고, 시점으로부터 보일 가능성이 없는 객체의 표면을 최대한 제거한다. 그리고 나머지 객체 표면만을 그래픽스 하드웨어에 보낸다. 이를 Durlach와 Mavor는 다각형 흐름 최소화(polygon flow minimization)이라고 하였다[8].

지난 몇 년 동안 다각형 흐름 최소화에 대한 연구가 다양한 측면에서 연구되었으며, 가장 큰 흐름은 시점으로부터 보일 가능성이 있는 최소의 객체를 효율적으로 찾아 이들 객체를 그래픽스 하드웨어에 보내 좀 더 효율적인 렌더링을 제공하고자 하는 것이다. 반대의 측면에서 생각하면 시점에 영향을 미치지 않는, 즉 시점으로부터 비가시한 가상 세계의 객체(invisible objects)를 최대한 파악하여 이를 제거한 후 렌더링을 하는 것이다. 두 분야의 공통적인 사항은 그래픽스 하드웨어에 객체를 보내기 전에 3차원 가상 공간의 객체 정보를 가시성 측면에서 처리하는 것이다. 이를 가시성 계산(visibility computation)이라 하며, 이에 관한 연구는 가상 공간을 구성하는 모형의 단순화에 기반으로 이론적인 측면과 경험적인 측면에서 이루어져 왔다[9].

이론적인 측면에서의 가시성 계산에 관한 연구는 특정 시점에서 보이는 정확한 영역을 구하는 알고리즘을 구하는 연구[17]와 공간상의 모든 시점에 대해 보이는 정확한 3차원 가상 공간의 볼륨을 구하는 알고리즘에 관한 연구가 있다[11]. 그러나 이들 알고리즘은 이론적으로 최적이지만 가상 공간상의 실시간 렌더링에 적용하기가 어렵다.

경험적인 측면에서 가시성 계산은 정확한 가시성을 계산하기보다는 가시성을 보장하는 객체를 구하는 포괄적 가시성(conservative visibility)을 효율적으로 계산하는 방향으로 진행되고 있다[5]. 지난 몇 년간 포괄적 가시성에 관한 연구는 투영 평면상에서 가시성을 처리하는 이미지 공간 접근 방법과 3차원 가상 공간상의 객체들간의 관계를 고려하여 가시성을 처리하는 객체 공간 접근 방법으로 구분된다. 또한 최근 네트워크 속도의 향상과 더불어 네트워크 환경에서의 가상 공간의 효율적인 렌더링 기법이 연구되고 있다.

본 고에서는 이들 세 가지 측면에서 포괄적 가시성의 주요 연구 결과를 각각 제 2장, 3장, 4장에서 살펴보고, 마지막으로 향후 주요 연구 방향을 제시한다.

2. 이미지 공간에서의 접근 방법

이 장에서는 이미지 공간에서 가시성 처리 방법에 대한 대표적인 연구 결과인 Greene 등[12,13,14], Zhang 등[27], HP VISUALIZE fx[21]의 idea를 살펴보고자 한다. 이들 결과의 공통점은 시점에 따른 이미지 생성에 대한 가시성 시험을 별도의 버퍼를 이용하여 시행한다는 것이다.

2.1 Z-피라미드 이용 방법

Greene 등[12]은 가상 공간을 octree로 구성하고, z-버퍼를 계층적으로 분할하여 구성한 Z-피라미드(Z-pyramid)를 이용하여 효율적으로 가시성을 처리할 수 있는 방법을 제안하였다.

Z-피라미드는 그림 4의 왼쪽에 나타난 바와 같이 가장 하단에 기존의 Z-버퍼 정보(NxN)를 저장하고, 이 정보를 (N/2 x N/2)로 샘플링(sampling)하여 바로 상단의 버퍼에 저장한다. 이러한 작업을 버퍼의 크기가 (1x1)까지 반복 처리한다. 그림 4의 오른쪽 아래에 가상 공간이 나타나 있고, 오른쪽

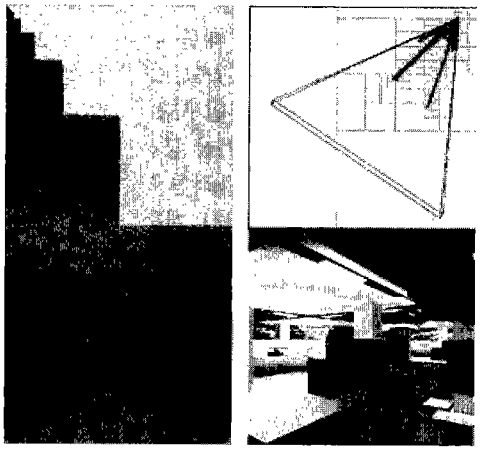


그림 4 구성된 계층적 Z-피라미드

위에 가상 공간에 대응되는 octree가 나타나 있다.

가상공간에 대해 octree가 구성되면, octree의 노드에 대해 front-to-back 순서로 순회하면서 미리 구성된 Z-피라미드에 관해 가시여부를 결정한다. 노드의 bounding box의 front 면이 Z-피라미드에 의해 완전히 가려지면 이 노드는 가시하지 않다. 이런 노드는 더 이상 최종 영상에 영향을 미치지 않으므로 순회를 중단한다. 완전히 가려지지 않는 노드에 대해서는 이 노드에 관련된 객체를 Z-피라미드에 렌더링하고 그의 자식노드에 대해 위의 작업을 반복 적용한다.

Greene이 처음에 제안한 방법[12]은 가림 관계 시험을 위해서만 Z-피라미드를 사용하였고 가시한 octree 노드에 있는 다각형을 렌더링하기 위해서는 기존의 Z-버퍼 기법을 그대로 이용하였다. 그의 자식노드에 대해 위의 작업을 반복 적용한다. 그 후 Greene[14]은 영상 공간에 대해 Z-피라미드를 구성하는 대신 triage coverage mask를 계층적으로 구성하는 방법을 제시하였다. 여기서 triage coverage mask는 영상의 각 점에 대해 다각형의 내부점인지, 외부점인지, 아니면 경계상의 점인지를 나타낸다(그림 5). 제안된 triage coverage mask를 이용하여 다각형의 가림 관계가 결정된다. 이때 가상공간의 다각형들이 BSP(binary space partition) 트리[18]로 구성되어 있으면 깊이 정보를 유지할 필요 없이 front-to-back 순서로 다각형을 순회가능하기 때문에 좀 더 효율적으로 가림 관계가 처리될 수 있다.

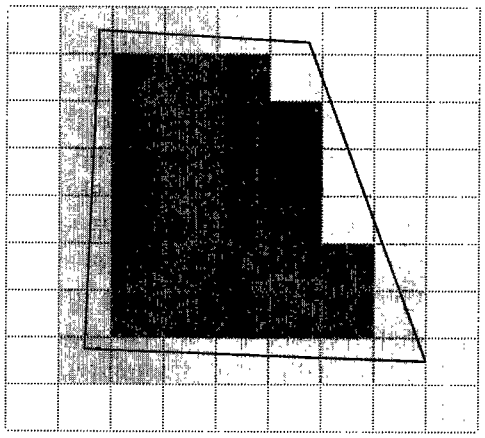


그림 5 Triage coverage mask

2.2 HOM 이용 방법

Zhang[27]에 의해 제시된 HOM(hierarchical occlusion map) 방법에서는 특정 시점으로부터 가장 잘 보이는 다각형이 가상공간에 있는 다른 다각형을 대부분 가릴 거라는 기본 생각을 가지고 먼저 가장 좋은 몇 개의 가리개(occluder)를 선택한다. 여기서 좋은 가리개란 시점으로부터 다른 많은 다각형을 가릴 수 있는 다각형이다. 이렇게 미리 선택된 가리개를 이용하여 실시간 렌더링 과정에서 가림 지도(occlusion map)를 계층적으로 만든다. 각 단계에서의 가림 지도는 그 하위 단계의 가림 지도를 2x2 픽셀 단위로 평균하여 만든다(그림 6).

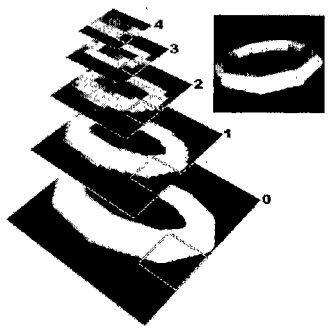


그림 6 HOM 구조

그림 6과 같은 HOM 구조를 이용하여 피가리개(occludee)와의 가림 관계를 시험하고 각 가리개의 깊이를 판단하기 위해 HOM의 각 가림 지도에 깊

이 버퍼를 추가한다.

구성된 HOM을 이용한 영상 생성은 먼저 다각형을 포함하는 가장 작은 사각형 내부 점들이 가림 지도와 겹치는 점의 불투명도(opaque)를 검사한다. 만약 모든 점들이 완전히 불투명하면 더 이상 작업을 진행하지 않는다. 특히 사각형 안의 점들의 불투명도를 검사하다가 불투명도가 너무 낮은 점이 발견되면 가시한 것으로 간주하여 모든 다각형을 렌더링하고 작업을 끝낸다. 그 이외의 경우, 다음 단계 수준의 가림 지도에 대해 위와 같은 방법을 적용한다. 특히 제시된 기법은 구현하기가 쉽고 가리개가 미리 선택되어 있어 그래픽 하드웨어로 구현하기가 매우 쉽다.

2.3 HP VISUALIZE fx

Hewlett-Packard는 VISUALIZE fx 그래픽스 하드웨어에 occlusion culling 기능을 구현하였다[22]. HP사에 구현된 occlusion culling 방법은 객체가 렌더링될 때, 그 객체를 포함하는 bounding box를 투영 평면에 투영하여 얻어진 픽셀의 깊이 정보와 이전에 저장된 Z-버퍼의 깊이 정보의 비교를 통해 가시성이 계산된다. 만약 이들 픽셀이 보여지지 않는 것으로 판단되면 그 객체는 가려지는 것으로 판단하고, 그렇지 않는 객체에 대해 Z-버퍼에 렌더링한다. 이러한 방법을 적용하면 복잡한 객체에 대한 가림 시험을 좀 더 쉽게 할 수 있어 약간의 성능향상을 가져올 수 있다. 하지만 모든 객체가 가려지지 않는 것으로 판단될 경우 추가적인 작업이 요구되므로 성능의 저하를 가져올 수 있다.

3. 객체 공간에서의 접근 방법

이 장에서는 포괄적 가시성을 계산하기 위한 객체 공간에서의 연구 결과를 소개한다. 이들 연구 결과의 공통적인 사항은 가상 공간을 구성하는 객체들간의 관계를 가시성 측면에서 전처리한 후, 특정 시점에서 가시한 부분을 계산하기 위해 전처리 결과를 이용한다는 것이다. 대표적인 연구 결과는 다음과 같다. Teller 등[23,24]은 가상 공간을 셀(cell)로 분할한 후, 셀과 포탈(portal)을 이용하여 가시성을 계산하였다. Coorg 등[6,7]은 객체를 가리개(occluder)와 피가리개(occludee)로 구분한 후, 두 객체간의 가시 여부를 이들 객체의 경계를 지나는 평면에 의해 결정될 수 있다고 보았다. Hudson [10]은 가리개에 대한 shadow frusta를 이용하여 가시성을 계산하였고, 그림자 볼륨을 이용한 가시성 계산의 연구도 있었다[19,25]. Schaufler [22]는 분할된 가상 공간의 볼륨을 투명, 불투명, 경계로 나누어 효율적인 전처리 과정에 이용하였다. 김대승[15]은 가시 계층을 정의하여 좀 더 효율적인 가시성 계산 방법을 제공하였다.

3.1 셀과 포탈 이용 방법

건물 내부와 같이 벽과 문을 포함한 많은 방으로 구성된 가상 공간에서 시점이 특정한 방(방번호 11)에 있다고 하자. 좀 더 효율적으로 가시성을 처리를 위해 시점이 움직일 수 있는 공간(viewpoint space)을 여러 개의 작은 시점 공간[이를 셀(cell)이라 함]으로 분할한 후, 셀을 통해 볼 수 있는 공간 정보를 미리 계산한다. 셀을 둘러싼 면들 중 불투명한 면이 벽이고 투명한 면이 포탈(portal-창문과 방문)이다. Airey[2]은 포탈로부터의 가시성을 해결하기 위해 포탈에서 출발하는 몇 개의 광선을 쏘아 만나는 다각형을 보일 가능성이 있는 면(PVS-potentially visible polygons set)에 포함시켰다. 이러한 생각을 확장하여 Teller 등[23]은 가상 공간을 BSP 트리를 이용하여 볼록 셀(convex cell)로 분할한 후, 포탈을 통한 인접 셀을 표현하기 위해 그림 7과 같이 셀간의 인접 그래프를 구성하였다. 인접 그래프의 정점은 셀을 나타내고 에지는 포탈로 연결된 두 셀의 연결을 나타낸다.

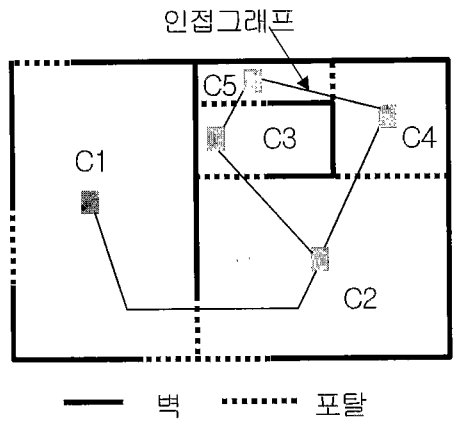


그림 7 가상공간의 셀 분할과 인접그래프

더욱이 각 셀과 교차라는 객체에 대해서도 BSP 트리 혹은 octree를 이용하여 계층화하여 관리한다.

Teller는 두 셀 Ci와 Cj간의 가시성(cell-to-cell visibility)을 구하기 위해 Ci 셀에 대응하는 인접 그래프의 정점을 찾아 이 정점으로부터 깊이 우선 탐색을 통하여 Cj 셀을 찾는다. Ci와 Cj에 대응되는 두 정점을 연결하는 경로가 Ci와 Cj 사이의 순서화된 포탈을 나타낸다(그림 8 참조). 이렇게 구한 순서화된 포탈을 모두 통과하는 직선이 존재하면 Ci 셀과 Cj 셀은 서로 볼 수 있고, 그렇지 않으면 두 셀간은 서로 볼 수 없다. 이것을 해결하는 문제를 Teller는 포탈 stabbing 문제라 하였고, 선형 프로그래밍 기법으로 모델링하여 해결하였다.

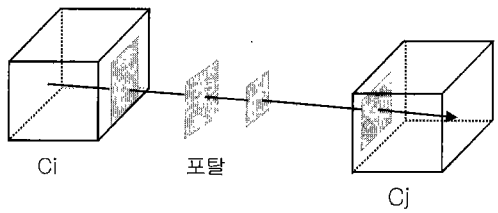


그림 8 포탈 stabbing

전처리 단계에서 cell-to-cell 가시성 검사를 이용하여 임의의 셀에서 가시한 셀(PVC-portentially visible cell)을 미리 구해 놓는다. 이러한 정보를 이용하여 특정 시점에서 가시한 부분을 구하기 위해 먼저 시점이 속하는 셀을 구한 후, 이 셀로부터 가시한 PVC에 속하는 어느 셀들이 시야(view frustum)과 교차하는지를 점검한다(이를 eye-to-cell visibility라고 함). 마지막으로 시점으로부터 가시한 셀에 속하는 다각형에 대해 가시한 영역을 구한다. 최근에는 포탈을 통해 가시한 다각형을 렌더링하기 보다는 포탈을 통해 보여지는 여러 영상을 저장한 후, 가장 적합한 영상을 이용하여 렌더링하는 방법이 제시되었다[1,20].

Lueke[16]는 전처리 작업으로 단순히 가상 공간을 분할하여, 분할된 셀간의 포탈을 통한 인접 그래프만을 구한다. 그리고 실시간에 렌더링시에 포탈을 통한 가시한 면을 구하는 방법을 제시하였다. 시점을 포함하는 셀의 포탈을 화면에 투영하고 투영된 포탈의 bounding box(이를 cull box라 함)를 구한다. 포탈을 통해 가시한 셀에 포함되는 객체를 렌더링할 때, 객체를 포함하는 bounding box와 cull box의 교차여부를 시험하여 교차하는 경우에만 렌더링한다. 교차 볼륨을 다음 단계의 cull box로 지정하고, 시점으로부터 가시한 다음 단계의 셀에 대해 위의 작업을 cull box가 공집합이 될 때까지 반복 적용하여 렌더링한다.

포탈과 셀을 이용하여 렌더링하는 방법의 복잡도는 분명 포탈의 개수에 영향을 받는다. 포탈의 개수가 적게 생성하는 건물 내부와 같은 밀집된 가상 공간(dense virtual space)에 대해서는 적은 수의 cell-to-cell 관계가 생성되어 효율적인 렌더링이 가능하다. 그러나 적은 수의 객체로 구성된 넓은 가상 공간(sparse virtual space)은 많은 수의 포탈이 생성되며, 이에 따라 고려해야될 cell-to-cell의 관계가 기하 급수적으로 늘어난다. 따라서 넓은 가상 공간에 대해 이 방법을 적용하기란 그리 쉽지 않다.

2.2 가리개/피가리개간 관계 평면 이용 방법

Coorg 등[6]은 많은 가상 공간의 다각형이 적은 수의 객체(이를 가리개(occluder)라 함)에 의해 가려질 수 있다는 기본 생각을 가지고 가상 공간의 객체를 가리개와 피가리개로 구분하여 전처리를 실행하는 방법을 제시하였다. 그들은 시점으로부터 가시한 부분이 넓으면 넓을수록, 즉 solid angle value가 크면 클수록 좋은 가리개로 선택하였다. 다음으로 선택된 가리개와 나머지 피가리개 사이의 가시 관계는 두 객체를 서로 다른 반공간에 놓이게 하는 분리 평면(separating plane)과 같은 반공간에 놓이게 하는 접하는 평면(supporting plane)에 의해 분할된 공간에 시점이 어떤 분할 공간에 있는냐에 따라 결정된다. 예를 들어 그림 9의 1 지역에 시점이 놓이면 피가리개 T는 완전히 보이며, 2지역에 시점이 있는 경우 T는 부분적으로 보이며, 3지역에 있는 경우 T는 가리개 A에 의해 완전히 가려진다.

그러나 n 개의 다각형으로 구성된 가상공간에서 분할된 영역의 개수가 $O(n^2)$가 될 수 있어 이를 줄이기 위해 같은 저자들에 의해 동적으로 가리개를 구성하는 방법이 제시되었다[7]. 이를 위해 먼저 가상 공간을 kD-트리로 구성한후, 이를 바탕으로 효율적인 가시성 계산을 위해 공간적 연관성(spatial coherence)와 시간적 연관성(temporal coherence)을 이용하였다.

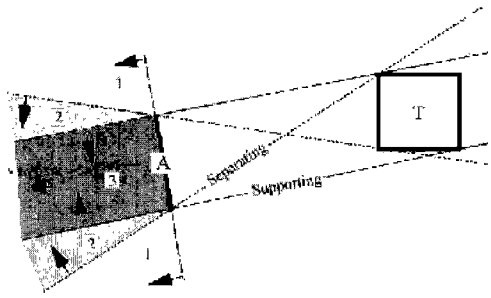


그림 9 가리개와 피가리개

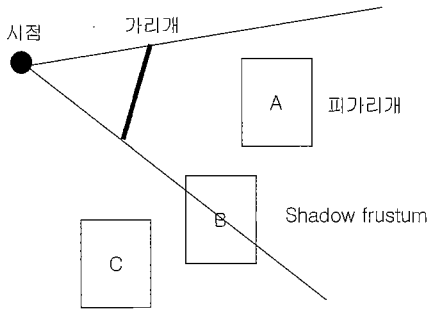


그림 10 Shadow frustum

2.3 Shadow Frusta 이용 방법

Hudson[10]은 Coorg 등[6]에서 논의한 생각과 유사하게 적은 수의 가리개가 가상공간의 다른 많은 객체를 가릴 수 있을 것이라고 생각했다. 그러나 그들은 가리개와 피가리개의 관계를 이용하여 가상공간을 분할하지 않고 미리 가상공간을 일정한 크기의 볼륨으로 분할한 후, 볼륨을 구성하는 각 정점에 대해 Coorg 등에서 제시한 solid angle value로 가리개를 선택한다. 8개의 정점에 대해 선택된 가리개에 대해 기준 값에 따라 몇 십 개의 객체를 볼륨에 대한 가리개로 선택한다. 이러한 작업은 전처리 과정에서 이루어진다. 또한 전처리 작업에서 가상 공간을 octree와 같은 계층 구조로 구성하고, octree의 각 노드에 해당되는 볼륨과 교차하는 다각형을 저장한다. 이렇게 구축된 자료구조를 가지고 실시간 렌더링을 다음과 같이 실행한다. 먼저 시점이 속하는 볼륨을 결정한 후, 이 볼륨에 저장된 몇 십 개의 가리개들 중에 view frustum에 속하지 않은 가리개를 제거하고 나머지 가리개들 중 8-10개 정도를 이 시점에 대한 가리개로 선택한다. 선택기준은 역시 solid angle value이다. 선택된 각 가리개에 대해 그림 10에서와 같이 shadow frustum을 구성한다.

실제 시점으로부터 가시한 다각형의 부분의 효율적 계산을 위해 octree를 깊이 우선 순서로 탐색하면서 octree의 노드에 저장된 볼륨과 shadow frustum과의 교차여부를 시험한다. 교차하지 않지 않으면 이 노드의 자식노드들에 대한 탐색을 더 이상 진행하지 않는다. 만약 볼륨이 shadow frustum 안에 완전히 포함되는 경우 자식 노드에 대한 더 이상의 탐색은 중지하고, 이 노드에 저장된 모든 다각형을 view frustum에 보내 렌더링한다.

볼륨이 완전히 포함되지 않으면 자식 노드들에 대해 위의 작업을 반복 적용하여 모든 다각형을 렌더링한다.

2.4 그림자 볼륨 이용 방법

좀더 실감나는 영상을 생성하기 위해 광원을 영역 혹은 볼륨으로 모형화하는 경우가 있다. 이러한 영역/볼륨 광원이 객체를 비추면 두 유형의 그림자 볼륨(shadow volume)인 완전음영부(umbr)와 반음영부(penumbra)기 형성된다[21,28]. 완전음영부는 빛이 하나도 비치지 않는 볼륨을 의미하며, 반음영부는 완전음영부를 포함하여 일부의 빛이 비쳐지는 볼륨을 의미한다(그림 11).

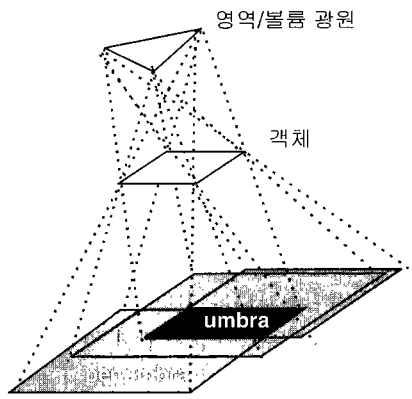


그림 11 그림자 볼륨

이러한 그림자 볼륨을 가시성 계산에 이용하기 위해 영역/볼륨 광원을 가상 공간의 분할된 볼륨에 대응시키고 객체를 그 볼륨에 대해 선택된 가리개로 대응시켜 보자. 그러면 분명 볼륨과 각 가리개

에 의해 형성된 완전 음영부에 속하는 모든 객체는 그 볼륨에 포함되는 모든 시점으로부터 보이지 않음을 알 수 있다[19,26]. 여기서 가리개는 완점음영부의 크기가 크면 클수록 좋은 가리개이다. 최근들어 유관희[25]는 전처리 작업에서 각 볼륨에 대해 가리개를 선택하여 그림자 볼륨을 구한 후, 완전음영부에 속하지 않는 객체에 대해 각 그림자 볼륨에 대해 반음영부와 교차하는 객체를 저장하고, 나머지 객체를 이 볼륨으로부터 가시한 객체로 유지한다. 그가 제시한 방법은 특정 시점에 대한 가리개들을 선택하는 방법은 Hudson[10]과 유사하지만 view frustum에 보내는 객체의 계산의 차이가 있다. Hudson은 모든 shadow frustum에 완전히 포함되지 않은 객체를 view frustum에 보내는 반면 유관희는 미리 구해진 이 볼륨에 대해 가시한 객체와 각 가리개에 저장된 객체들을 즉시 view frustum에 보낸다. 이 방법은 렌더링시 별도의 작업이 필요 없을 뿐만 아니라 많은 객체들이 볼륨 가시성 계산 시 제거될 것이라는 사실 때문에 효율적인 실시간 렌더링을 보장할 수 있다. 반면 각 볼륨에 포괄적 가시성 정보가 저장되어야 함으로 과다한 저장장소가 요구되는 단점이 있다.

2.5 볼륨 가시성 이용 방법

Schaufler[22]가 제시한 방법으로 가상 공간에서 가리개를 선택하지 않고, 불투명한 객체의 내부가 시점으로부터 이 객체의 뒷 공간을 가린다는 기본 생각을 가지고 먼저 가상 공간을 octree로 구성한다. 구성된 octree의 각 볼륨을 불투명, 경계, 투명으로 구분한 후, 불투명 볼륨을 가리개로 사용하였다. 분명 가리개는 크면 클수록 좋으므로 주변 볼륨의 상태를 확인하여 최대한 크게 불투명 볼륨을 만든다. 이러한 전처리 과정 후에 view frustum에 들어오는 가리개를 이용하여 octree의 각 노드를 순회하면서 모든 객체의 가림 상태를 결정한다.

2.6 가시 계층 이용 방법

도시 환경과 같이 보다 일반적인 야외 환경을 다룰 수 있는 방법이 김대승[15]에 의해 연구되었으며, 그는 가상 공간에 대해 효율적으로 가시성 계산을 처리하기 위해 가시 계층(visiblity layers)이라는 자료구조를 제시하였다. 가시 계층을 정의하기 전에 먼저 시점이 움직일 수 있는 볼륨을 설정한다. 모든 객체들 중 시점 볼륨으로부터 완전 가시한 객체를 첫 번째 가시 계층으로 선택한 후, 이들 객체를 제외한 나머지 객체에 대해 위와 똑같은 작업을 수행한다. 그림 12가 위의 방법에 따라 가시계층으로 분리된 객체들을 보여주고 있다.

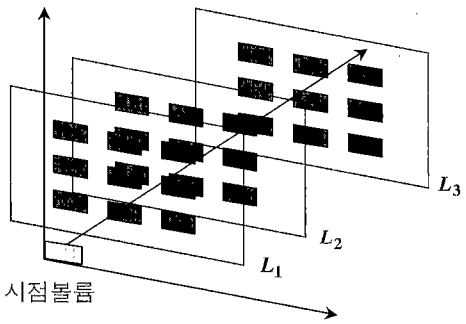


그림 12 가시 계층

가시성 계산을 효율적으로 처리하기 위해 각 가시계층에 속한 객체들에 대해 그림 13와 같이 볼륨으로부터 반음영부를 구하여 합한다. 이때 반음영부의 합집합의 경계가 물체의 경계와 물체들을 연결하여 만들어진 부분으로 구분된다. 가시성 계산을 효율적으로 처리하기 위해 물체의 경계를 가리개로 설정하였고 나머지 생성된 부분을 포탈로 설정하였다. 이러한 가상 공간의 전처리 과정을 거친 후, 볼륨에 속하는 시점으로부터의 가시성 계산은 첫 번째 가시 계층에서 만들어진 반음영부에 대해 그 시점에 대한 view frustum과 교차하는 부분을 구하여 투영 평면에 투영한다. 투영 평면에 포탈이 형성되면 두 번째 가시 계층의 반음영부와 view frustim의 교차영역을 구해 형성된 투영 평면에

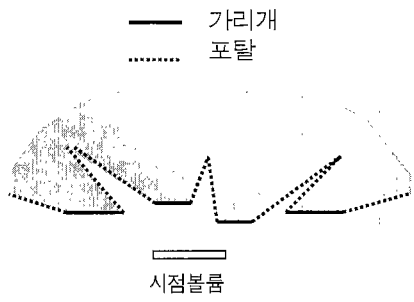


그림 13 가시 계층에서 형성된 반음영부

다시 투영한다. 역시 투영 평면에 포탈이 형성되는지를 확인하여 포탈이 없으면 작업을 멈추고 그렇지 않으면 위의 작업을 모든 가시 계층에 대해 차례대로 실행하여 가시성을 계산한다.

가시 계층을 이용한 가시성 계산은 가상 공간을 구성한 객체들이 시점 볼륨으로부터 잘 정돈되어 있는 경우에 효율적으로 적용될 수 있다. 다시말해 가시계층의 개수가 적으면 적을수록 잘 작동된다. 그러나 일반적으로 가상 공간을 구성하는 시점 볼륨으로부터 서로 가림 관계에 있는 객체들이 수없이 많이 존재할 수가 있다. 이러한 문제점을 제거하기 위해 객체를 분할해야 하는데, 이 경우 가시계층의 개수가 늘어날 수 있다.

4. 네트워크 환경에서의 포괄적 가시성

최근 네트워크 기술의 발전과 더불어 client-server 모형에서 서버가 3차원 가상 공간을 client에 제공하여 다양한 형태의 응용이 서비스되고 있다. 이러한 네트워크 환경에서 실시간 상호 작용이 가능한 그래픽스 시스템을 구성하기 위한 가장 장애 요인은 network bandwidth와 transmission latency이다. 이를 극복하기 위해 기존의 occlusion culling 기술을 이용하여 서버가 client의 현 시점에서 보일 가능성이 있는 객체만을 client에게 제공해야 한다. 더욱이 약간의 시점 변화가 일어나는 경우 서버는 새로 이 시점으로부터 가시한 객체를 계산하여 client에게 전송해야 한다. 이때 가장 치명적인 문제는 latency이다. 이러한 이유로 인해 off-line에서 연구된 다양한 결과들을 직접적으로 네트워크 환경에 이용하기란 그리 쉽지 않다.

이 문제를 해결하기 위해 Cohen 등[4]은 주어진 시점으로부터 e만큼 떨어진 볼륨을 e-neighborhood로 정의하였고, 이 볼륨으로부터 가시한 모든 객체의 e-superset으로 정의하였다. 그들은 서버가 client에게 미리 e-superset을 보내 시점이 e-neighborhood내의 시점으로 이동하는 경우 더 이상의 자료 전송이 일어나지 않게 함으로써 transmission latency 문제를 해결하고자 하였다. 2차원에서의 e-neighborhood를 점검하는 방법을 고려해 보자. 그림 14에서 e2가 선분 L로부터 보이지 않을 때, e2에 인접한 시점 e1가 e2의 e-neighborhood에 속하는지를 점검하기 위한 방법으로 e1 시점에 대해 선분 L 객체의 양 끝점을 연결하는 두 직선을 구한다. 두 직선과 교차하는 객체 집합을 각각 구한 후, 두 집합의 공통 객체가 있는지를 점검한다. 공통 객체가 존재하면 분명 e1은 선분 L로부터 보이지 않는다. 따라서 e1은 e2의 e-neighborhood이다. 두 시점이 다른 한 점의 e-neighborhood인지를 점검하기 위해서는 선분 L의 양 두 끝점과 두 시점이 연결하는 직선을 구한 후, 각각의 직선과 교차하는 각 객체 집합에 대해 공통 객체가 존재해야 한다.

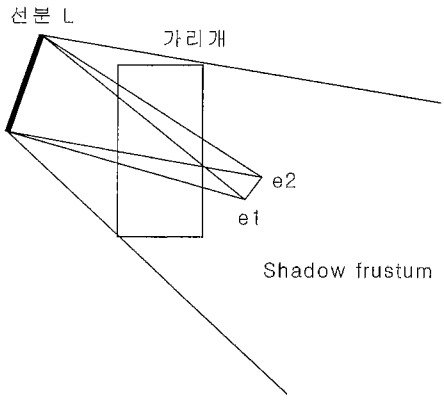


그림 14 두 시점의 e-neighborhood

그들은 이러한 2차원상의 방법을 3차원으로 확장하기 위해 선분은 삼각형으로 고려하였고, e-neighborhood를 네 점에 의해서 형성된 사면체로 정의하였다.

5. 결 론

본 고에서는 가상 공간의 실시간 렌더링을 위해 요구되는 포괄적 가시성을 이미지 공간, 객체 공간, 네트워크 환경으로 분류하여 정리하였다.

이미지 공간에서의 접근 방법은 선택된 가리개 정보를 이용하여 투영 평면상에 계층적 구조를 구성하고, 전체 가상 공간에 대해 octree을 구성한다. 구성된 두 계층 구조를 이용하여 객체의 겹침 검사, 객체가 view frustum에 놓여지는지 아닌지를 효과적으로 처리할 수 있다. 이처럼 이미지 공간에서는 어떻게 하면 시점으로부터 이미지를 생성할 때 최대한 보이지 않은 면을 효율적으로 제거할 수 있는가에 초점을 맞춰 연구되고 있다.

객체 공간상의 가시성 처리에 관한 연구는 가상

공간을 시점을 포함하는 볼륨으로 분할한 후, 이들 볼륨에 대한 가시성을 어떻게 하면 효율적으로 처리할 수 있을지에 초점을 맞추어 연구되고 있다.

네트워크 환경에서의 가시성은 서버에서 client에 보내는 가시한 객체를 최대한 적게 보내기 위해 방법으로 연구가 진행되고 있다.

위에서 논의된 방향으로 세 분야에 대한 연구는 진행될 것으로 판단되며, 앞으로 동적으로 변화하는 가상 공간에서 효율적인 가시성 계산을 행할 수 있는 연구가 이루어질 것으로 예측된다.

참고문헌

[1] D.G. Aliaga and A.A. Lastra, "Architectural walkthroughs using portal textures," *IEEE Visualization '97*, pp.355-362, 1997.

[2] J. Airey, J. Rohlf and F. Brooks, "Towards image realism with interactive update rates in complex virtual building environments," *Proc. of ACM Symposium on Interactive 3D Graphics*, pp.41-50, 1990.

[3] E. Angel, *Interactive computer graphics: a top down approach with OpenGL*, 2nd Edition, Addision Wesley, 2000.

[4] D. Cohen-Or and E. Zadicario, "Visibility streaming for network-based walkthroughs," 1996.

[5] Cohen and Wallac, *Radiosity and realistic image synthesis,* Academic Press Professional, 1993.

[6] S. Coorg and S. Teller, "Temporally coherent conservative visibility," *Proc. of 12th ACM Symposium on Computational Geometry*, 1996.

[7] S. Coorg and S. Teller, "Real-Time occlusion culling for models with large occluders," *Proc. of ACM Symposium on Interactive 3D Graphics*, 1997.

[8] N.I. Durlach and A.S. Mavor, *Virtual reality: scientific and technological challenges,* National Academy Process, Washington, D.C., 1995.

[9] J. Heo, "A survey on culling for virtual reality," Tech Memo 98-36, VR Group, Dept. of Computer Science, KAIST, 1998.

[10] T.Hudson, D. Manocha, J. Cohen, M. Lin, K. Hoff and H. Zhang, "Accerlated Occlusion Culling using Shadow Frusta," *Proc. of 13th ACM Symposium on Computational Geometry*, pp.1-10, 1997.

[11] Z.Gigus, J. Canny and R. Seidel, "Efficiently computing and representing aspect graphs of polyhedral objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 13, No 6, pp 542-551, 1991.

[12] N. Greene, M. Kass and G. Miller, "Hierarchical z-buffer visibility," *Proc. of ACM Siggraph*, pp.231-238, 1993.

[13] N. Greene, *Hierarchical rendering of complex environments*, PhD Dissertation, University of California Santa Cruz, 1995.

[14] N. Greene, "Hierarchical polygon tiling with coverage masks," ACM SIGGRAPH'96, pp.65-74, 1996.

[15] D. Kim, *Visibility layers and their applications to rendering,* PhD Dissertation, Dept. of Computer Science, KAIST, 1997.

[16] D. Luebke and C. Georges, "Portals and mirrors: simple, fast evaluation of potentially visible sets," *ACM SIGGRAPH Special Issue on 1995 Symposium on Interactive 3D Graphics,* pp.155-162, 1995.

[17] M. McKenna, "Worst case optimal hidden-surface removal," *ACM Trans. Graphics*, Vol.6 pp.19-28, 1987.

[18] B. Naylor, "Interactive solid geometry via partitioning trees," *Proc. of Graphics Interface*, pp.11-18, 1992.

[19] T. Nishita and E. Nakamae, "Half-tone represetation of 3-D objects illuminated by area sources or polyhedron sources," *Proc. of IEEE COMPSAC*, pp.237-242,

1983.

[20] V. Popescu, A. Lastra, D.G. Aliaga and M.O. Neto, "Efficient warping for architectural walkthroughs using layered depth images," *IEEE Visualization '98*, pp.18-24, 1999.

[21] N. Scott, D. Olsen and E. Gannet, An overview of the visualize fx graphics accelerator hardware, *The HP Journal*, May:28-34, 1998.

[22] G. Schaufler, J. Dorsey, X. Decoret and F.X. Sillion, "Conservative volumetirc visibility with occluder fusion," *Proc. of ACM SIGGRAPH*, 2000.

[23] S.Teller and C.H. Sequin, "Visibility preprocessing for interactive walkthrough," *Proc. of ACM SIGGRAPH*, pp. 61-69, 1991.

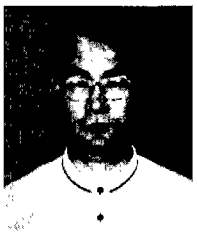
[24] S.Teller, *Visibility Computations in Densely Occluded Polyhedral Environments*, PhD Dissertation, University of Berkeley, 1992.

[25] K.H. Yoo, *An efficient visibility computation for real-time walkthrough*, Research Report, Chungbuk National Unversity, 2000.

[26] K.H. Yoo, *3D visibility and its applications*. PhD Dissertation, Dept. of Computer Science, KAIST, 1995.

[27] H. Zhang, D. Manocha, T. Hudson and K. Hoff, "Visibility culling using hierarchical occlusion maps," *Proc. of 14th ACM SIGGRAPH*, 1997.

유 관 희



1985 전북대학교 전산통계학과(학사)
1988 한국과학기술원 전산학과(석사)
1995 한국과학기술원 전산학과(박사)
1988.1~1997.8 (주)데이콤 종합연구소 선임연구원
1997.8~현재 충북대학교 컴퓨터교육과 교수
2000.2~현재 (주)덴탈그래픽 개발이사
관심분야:컴퓨터 그래픽스, 실시간 렌더링, 3D 치과 응용(보철,교정)
E-mail:khyoo@cgs.chungbuk.ac.kr