시간 일관성을 이용한 효율적인 z-버퍼 알고리즘

(An Efficient z-Buffer Algorithm using Temporal Coherence)

오 경 수 † 신 영 길 †† 신 병 석 †††
(Kyung Su Oh) (Yeong Gil Shin) (Byeong-Seok Shin)

요 약 이 논문에서는 시점이 고정된 경우 프레임간의 시간 일관성을 이용해서 z-버퍼 알고리즘의 렌더링 속도를 향상시키는 방법을 제안한다. z-버퍼 알고리즘은 일단의 다각형들을 렌더링하는 동안 각 화소의 깊이 값을 저장해서 나머지 다각형들을 렌더링할 때 이들의 가시성을 판별하는데 사용한다. 만약 일부 다각형들에 해당하는 색상과 깊이 정보를 렌더링을 하지 않고도 얻어 낼 수 있다면, 나머지 다각형들만을 렌더링해서 동일한 화상을 얻을 수 있다. 화면의 갱신주기가 짧은 경우 연속된 두 프레임에서 움직임이 없는 다각형들의 집합은 상당히 큰 일관성을 가진다. 이러한 시간 일관성을 이용하면 연속한 프레임에서 움직임이 없는 다각형들의 색상과 깊이 정보를 새롭게 계산할 필요가 없다. 이를 위해 고정된 다각형들의 색상과 깊이 정보를 따로 저장해두었다가 재사용하는 방법을 제안한다. 이 방법은 복잡한 자료구조를 사용하거나 기존 z-버퍼 알고리즘 자체를 바꾼 것이 아니므로 구현이 쉽고 하드웨어로 구현하기도 용이하다.

Abstract We present a method that enhances the rendering speed of z-buffer algorithm using temporal coherence between two contiguous frames on fixed viewing conditions. Conventional z-buffer algorithm stores depth value for each pixel on a view plane while rendering some polygons, then it determines the visibility of the remaining polygons based on the stored depth values. If we can get color and depth information for some polygons without rendering, it is possible to generate an image by rendering only the remaining ones. In case of high frame rate, we can find the fact that sets of static polygons of the two contiguous frames are almost the same. This temporal coherence enables us to get the color and depth information of static polygons efficiently. Our algorithm stores color and depth information of static polygons and reuses it for generating the next frame. This method can be easily implemented since it does not require complex data structure and modification for conventional z-buffer algorithm. Also it is adequate for hardware implementation.

1. 서 론

실시간 렌더링(*real-time rendering*)은 사용자에 의해 정의되거나 프로그램에서 생성된 3차원 모델로부터 실시간으로 화상을 생성하는 것이다. 실시간 렌더링은 다양한 시점에서 여러 가지 상황을 사용자의 요구에 따라 만들어볼 수 있는 장점이 있지만, 제한된 시간 내에 렌더링을 해야 하므로 화상의 크기와 화질에 많은 제약이 가해진다. 근래에 들어 기존의 2차원 컴퓨터 게임들이 3차원으로 구현되고, 과거에는 생각지 못했던 복잡한 가상 환경을 구현해야 할 응용 분야가 늘어남에 따라 보다 빠른 렌더링 방법이 필요하게 되었다.

다각형 모델 렌더링에서 성능을 결정하는 가장 중요한 요소는 가시면 결정 방법이다. 대표적인 가시면 결정 방법인 z-버퍼 알고리즘[1]은 구현이 간단하고 적절한 화질을 유지하면서도 속도가 빨라서 실시간 렌더링에 많이 사용된다. z-버퍼 알고리즘에서 렌더링 속도는 다각형의 수에 비례한다. 최근에는 z-버퍼 알고리즘을 하

† 학생회원 : 서울대학교 전산과학과
oks@cglab.snu.ac.kr
†† 종신회원 : 서울대학교 전산과학과 교수
yshin@cglab.snu.ac.kr
††† 정 회 원 : 비트컴퓨터 의료영상연구실 실장
bsshin@cglab.snu.ac.kr
논문접수 : 1998년 12월 24일
심사완료 : 1999년 11월 4일

드웨어로 구현하는 등 짧은 시간에 많은 다각형을 처리하려는 노력이 계속 되고 있지만[2][3][4] 요구되는 데이터의 크기와 복잡도는 그보다 빨리 증가하고 있다.

이러한 복잡한 모델을 실시간으로 렌더링하기 위해서 비가시 다각형 제거법(visibility culling)[5][6][7][8], LOD(level of detail), 택스처링(texturing)[9]등 렌더링 될 다각형의 개수를 줄이는 방법들이 많이 연구되어 왔다. 이 논문에서는 시점(view point)과 시선(viewing direction)이 고정되었을 때 이전 프레임의 렌더링 정보를 활용하여 현재 프레임에서 움직이는 다각형들만을 선택적으로 그리는 고속 렌더링 방법을 제안한다. 이 방법은 시점이나 시선이 변화하는 최악의 경우에도 기존 z-버퍼 알고리즘보다 많은 시간이 소요되지는 않는다.

z-버퍼 알고리즘에서는 색상 정보를 저장하는 색상 버퍼와 깊이 값을 저장하는 동일한 해상도의 z-버퍼가 사용된다. 실시간 렌더링에서는 화면을 갱신하는 매 순간마다 색상 버퍼와 z-버퍼의 내용을 지우고 모든 다각형들을 다시 렌더링 해야 한다. 그런데 시점과 시선이 고정되고 화면의 갱신 주기가 짧은 경우, 두 화상에서 이동한 다각형들이 그려지는 부분만 다르기 때문에 특정 시간에 생성된 화상과 직전에 생성된 화상은 변화가 적다. 이러한 시간 일관성을 이용하여 이동한 다각형만을 렌더링하면 상당한 속도향상을 기대할 수 있다.

2절에서는 z-버퍼 알고리즘을 가속화하기 위한 기존 연구들을 살펴보고, 3절에서 알고리즘의 기본 가정과 사용되는 용어를 정의한다. 4절에서는 이해를 돕기 위해 다각형들의 이동상태가 미리 결정되어 있는 경우의 단순한 렌더링 방법을 설명하고, 5절에서는 다각형들의 이동상태가 수시로 변하는 경우의 렌더링 방법을 설명한다. 6절에서는 실험 결과를 소개하고 결론을 맺는다.

2. 관련 연구

가시면 결정에 시간 일관성을 이용하는 기존의 방법들에는 다음과 같은 것들이 있다.

첫 번째로는 모델은 고정되어 있고 시점만 변하는 상황에서 시점이 움직일 때마다 새로 보이거나 보이지 않게 되는 다각형들을 효율적으로 찾아주는 방법이다. Hubschman은 시점만 움직이는 상황에서 정확한 가시성 판별 방법을 제안하였고[10] Coorg는 보일 가능성이 있는 다각형을 실제보이는 것보다 더 많이 찾는 대신 문제를 간단히 해서 더 간결한 방법으로 해결한다[11]. 이들 방법들은 모두 볼록한 다각형으로 이루어진 모델들에만 적용된다.

두 번째는 렌더링 된 다각형에 의해서 가려지는 다각형들을 조기에 판단하는 방법들이다. 이러한 방법들은 이전 프레임에 가시적인 다각형이 다음 프레임에서도 나타날 확률이 높은 시간 일관성을 이용한다. 즉, 이전 프레임에서 가시적이었던 다각형들을 먼저 처리하는 것이다. 이 방법들에는 계층적 가시성 전처리 방법(visibility preprocessing)[5], z-버퍼 이용법(Hierarchial Zbuffer)[6], 계층적 다각형 나열법(hierarchical polygon tiling)[7], 계층적 폐색도 이용법(hierarchical occlusion map)[8]등의 방법들이 있는데 이들은 복잡한 전처리 과정이 필요하거나 화상 공간의 미리 그려진 다각형들의 위치 정보를 알아내기 위해서 계층적 z-버퍼(hierarchical zbuffer), 적용 영역 마스크(coverage mask), 계층적 폐색도(hierarchical occlusion map)를 사용한다. 이들 방법들은 복잡한 자료구조를 사용하고 화면에 나타나는 다각형들을 모두 렌더링 해야 하므로 가시 다각형들이 많은 경우 속도향상에 한계가 있다.

세 번째는 시점이 고정된 상황에서 광선 투사법을 이용한 가시면 결정방법의 속도를 향상시키기 위한 방법들이다. 이 방법들은 시점이 고정된 상태에서 변화가 없는 물체를 처리하지 않으려는 데서 본 연구와 유사하다. Javans는 모델 공간을 여러 셀(cell)들로 나누고 각 셀마다 자신을 지나가는 광선(ray)들을 저장 한 후 물체가 움직이면 그에 영향을 받는 셀을 지나가는 광선들만 다시 투사하는 방법을 제안하였다[12]. Chapman, Badt는 몇 프레임 건너뛴 화상들을 비교해서 차이가 있다고 판단되는 화소들에 대해서만 중간 프레임들의 화상을 만들 때 광선을 투사한다[13][14]. 중간화상들은 차이가 있어도 양 끝 화상에서는 차이가 없을 수 있으므로 이 방법은 틀린 화상을 만들어 낼 가능성이 있다.

기존의 연구들은 모든 물체가 볼록하고 정적인 경우[10][11], 깊이 복잡도(depth complexity)가 높을 경우[5][6][7]에 시간일관성을 사용하여 좋은 결과를 얻었고 본 연구에서도 시간일관성 이용하여 z-buffer알고리즘을 개선시켰다. 단 시점이 고정된 상태에서만 속도향상이 있고 동적 다각형의 개수가 적은 경우에 큰 속도향상을 기대할 수 있다.

3. 사용 용어 및 기본 가정

동적으로 변하는 가상 환경을 렌더링 할 때, 모델을 구성하는 다각형 $p_1,\ldots,p_n$의 집합을 P라 하자. 특정한 시간 $t-1$과 t에서의 연속된 두 프레임 f_{t-1}, f_t에서 위치가 변하는 다각형을 동적 다각형(dynamic polygon),

위치가 변하지 않는 다각형을 정적 다각형 (static polygon)이라 하고, 이들의 집합을 각각 P_d와 P_s라 하자. 그러면 $P=P_d \cup P_s$, $P_d \cap P_s=\varnothing$이다.

z-버퍼 알고리즘에서는 한 쌍의 색상 버퍼와 z-버퍼가 사용되는 데 이것을 〈C,Z〉라고 표기한다. 이 논문에서는 정적 다각형들의 색상과 깊이 값들을 저장하기 위해 또 하나의 버퍼 쌍을 사용한다. 이들을 구별하기 위해 기존 색상 버퍼와 z-버퍼를 각각 "렌더링 색상버퍼"와 "렌더링 z-버퍼"라 하고 이 둘을 합쳐서 "렌더링 버퍼 쌍(〈RC,RZ〉)"이라 한다. 정적 다각형들의 색상과 깊이 값을 저장하는 버퍼들을 각각 "정적 색상 버퍼"와 "정적 z-버퍼"라 하고 이들을 "정적 버퍼 쌍(〈SC,SZ〉)"이라 한다. 〈RC,RZ〉와〈SC,SZ〉에 저장된 색상과 깊이 값은 각각 〈*RC,RZ*〉와 〈*SC,SZ*〉로 표기한다. 이 알고리즘에서 실제적인 렌더링은 렌더링 버퍼 쌍에만 수행되고 정적 버퍼 쌍은 렌더링 버퍼 쌍과 데이터를 교환하는 데만 사용된다.

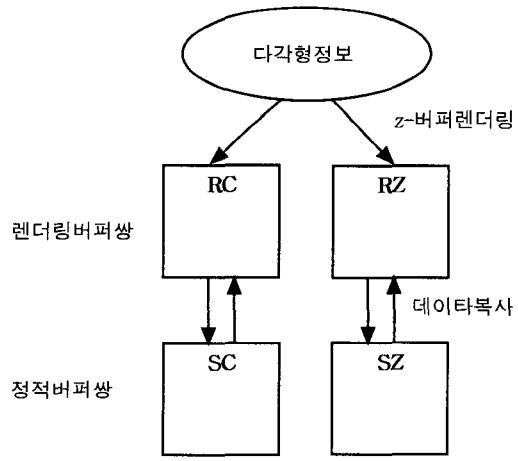

그림 1 렌더링 버퍼 쌍과 정적버퍼 쌍의 관계

각각의 다각형은 스캔 변환되면 화상 공간상에서 일정 영역을 차지하게 된다. 본 논문에서는 이 영역을 근사적으로 나타내기 위해서 영역을 포함하며 변들이 좌표축에 평행한 직사각형을 사용하며 이를 화상영역(extent)라 부른다. 즉, 그림 2에서 보는 바와 같이 화상공간에서 다각형을 둘러싸는 사각형으로 2개의 점으로 표현 할 수 있다. 다각형들의 화상공간상의 교차 검사를 할 때 화상영역의 교차검사를 사용함으로써 근사적이지만 효율적으로 처리 할 수 있다. 화상영역은 z-버퍼 알고리즘을 구현할 때 스캔 변환 단계에서 계산할 수도 있지만, z-버퍼를 지원하는 상당수의 하드웨어나 소프트웨어 라이브러리에서 화상영역 계산 기능을 제공한다[15]. 여기서는 기본적으로 모든 다각형에 자신의 화상영역 정보가 저장된다고 가정한다.

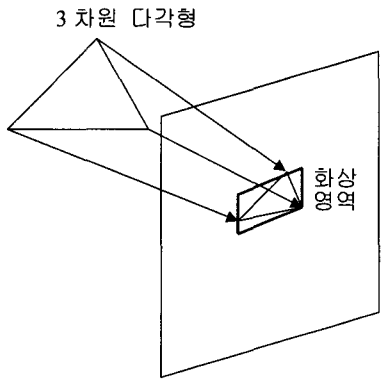

그림 2 다각형의 화상 영역

4. 동적 다각형과 정적 다각형이 결정된 경우의 렌더링 방법

실시간 렌더링을 하기 전에 동적 다각형들과 정적 다각형이 미리 정해져 있고 렌더링 하는 동안 그 상태가 변하지 않는다면 기존의 z-버퍼 알고리즘을 약간 변형하여 렌더링 속도를 향상시킬 수 있다.

z-버퍼 알고리즘은 렌더링 되는 다각형의 순서에 무관하게 화상을 생성하는 특징이 있다. 다각형의 집합 $P=\{p_1, p_2, \ldots, p_n\}$의 한 순열(permutation) $PP=(q_1, q_2, \ldots, q_n)$이 있을 때 이 순서대로 처리한 후의 색상 버퍼와 z-버퍼 내용을 $\langle R_{PP}, Z_{PP}\rangle$라 하자. 그러면 P의 또 다른 순열 PP'에 대해서 $\langle R_{PP}, Z_{PP}\rangle=\langle R_{PP'}, Z_{PP'}\rangle$이 성립한다. $P=P_d \cup P_s$이므로 P_d의 임의의 순열 P_dP과 P_s의 임의의 순열 P_sP가 있을 때 (P_dP, P_sP)도 P의 순열이 된다. 따라서 $\langle R_P, Z_P\rangle=\langle R_{(P_dP, P_sP)}, Z_{(P_dP, P_sP)}\rangle$가 성립한다. 결과적으로 정적 다각형들을 먼저 렌더링 한 후, 동적 다각형들을 렌더링해도 임의의 순서로 렌더링한 경우와 동일한 화상을 얻을 수 있다.

그런데 정적 다각형을 렌더링한 결과인 $\langle R_{P_s}, Z_{P_s}\rangle$는 시점과 시선이 변하지 않는 한 항상 일정하다. 본 논문의 알고리즘에서는 $\langle R_{P_s}, Z_{P_s}\rangle$를 한 번 계산하여 정적 버퍼쌍에 저장한 후 다음 프레임의 화상 생성에 이용한다.

특정 시간 t에서 프레임 f_t의 렌더링을 시작할 때,

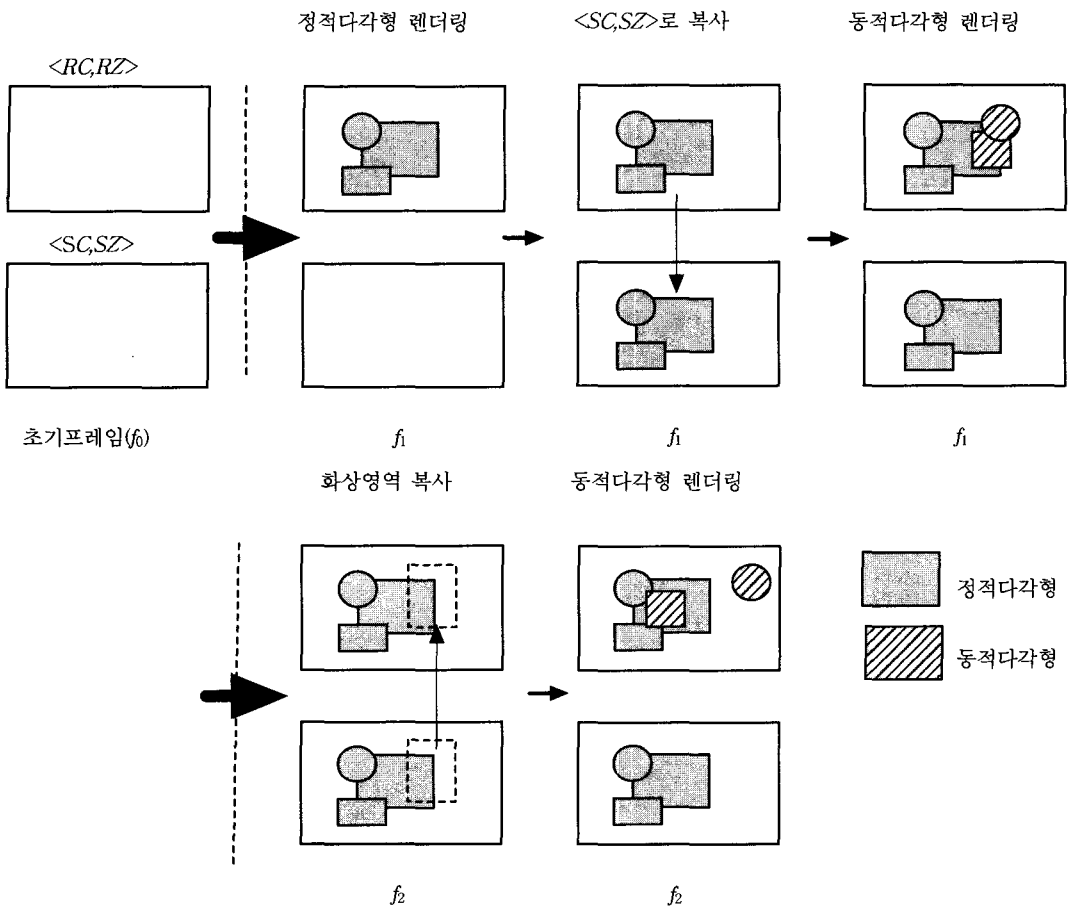

그림 3 다각형들의 이동상태가 결정된 경우의 렌더링 예

렌더링 버퍼쌍에는 f_{t-1}에서 렌더링된 모든 다각형들의 색상과 깊이 값인 $\langle RC_P^{t-1}, RZ_P^{t-1}\rangle$가 저장되어 있다. f_{t-1}에서 동적 다각형들의 화상 영역에 해당하는 부분을 정적 버퍼 쌍으로부터 렌더링 버퍼 쌍으로 복사하면 두 버퍼쌍의 값이 $\langle SC_{P_t}, SZ_{P_t}\rangle$로 같아진다. 이것은 모든 정적 다각형들을 렌더링한 결과인 $\langle RC_{P_t}^t, RZ_{P_t}^t\rangle$에 해당된다. 이처럼 정적 버퍼쌍의 일부만 렌더링 버퍼쌍으로 복사함으로써 정적 다각형들을 새롭게 렌더링하는 오버헤드를 제거할 수 있다. 정적 다각형들에 대한 렌더링이 끝난 후 기존 z-버퍼 알고리즘을 동적 다각형들에 적용해서 렌더링하면 현재 프레임 f_t에서의 모든 다각형에 해당하는 색상과 깊이 정보인 $\langle RC_P^t, RZ_P^t\rangle$를 얻을 수 있다.

이 방법은 시점과 시선이 고정되어 있을 때 효과적이다. 시점이 변경되는 경우는 관측자를 기준으로하여 모든 다각형이 이동하는 것과 같으므로, 다각형들을 매번 다시 렌더링하고 정적 버퍼쌍의 내용을 수시로 갱신해야 한다. 이 때문에 기존 z-버퍼 알고리즘보다 성능이 떨어질 수 있다. 이러한 문제를 해결하기 위해 정적 버퍼쌍은 시점의 변화가 중단된 순간부터 사용한다. 즉 프로그램이 시작하거나 시점의 변화가 멈추는 순간에 $\langle R_{P_t}, Z_{P_t}\rangle$를 계산하여 다음번 시점이 변화하는 순간까지 사용하고, 시점이나 시선이 변하는 동안은 기존의 z-버퍼 알고리즘을 사용한다.

5. 정적 다각형과 동적 다각형이 결정되지 않은 경우

정적 다각형과 동적 다각형이 미리 결정되고 변하지 않은 경우는 흔하지 않다. 대부분의 응용에서 다각형들은 예측할 수 없는 패턴으로 동적인 상태와 정적인 상태를 반복한다. 그러므로 매 프레임마다 각 다각형의 이동 상태를 판별해서 렌더링해야 한다. 4절에서는 시점이 고정된 동안에는 정적 버퍼쌍의 내용이 변하지 않았지만, 다각형의 움직임이 일정하지 않은 경우는 정적 버퍼쌍의 내용도 시간에 따라 달라질 수 있다.

정적 버퍼쌍을 사용하는 이 논문의 알고리즘에서는 모든 다각형들이 항상 동적인 경우는 기존 z-버퍼에 비해 성능이 나아지지 않는다. 그러나 일정 시간동안 정적인 상태를 유지하는 다각형들이 차지하는 비율이 높을수록 특정시간 t에서의 정적 버퍼 쌍의 내용 $\langle SC^{t}_{P_s}, SZ^{t}_{P_s}\rangle$과 $t-1$에서의 정적 버퍼 쌍의 내용 $\langle SC^{t-1}_{P_s}, SZ^{t-1}_{P_s}\rangle$간의 유사성이 커진다. 여기서는 이러한 시간 일관성을 기반으로 4절의 알고리즘을 확장하여 현재 프레임에서의 화상과 깊이 값을 신속하게 계산하는 방법을 설명한다. 이를 위해 먼저 각 다각형들을 운동성에 따라 네 가지로 분류한다. 4절의 방법에서는 모든 다각형이 P_{dd}나 P_{ss}에 포함되는 경우만을 고려하였지만, 여기서는 각 다각형들이 표 1에서와 같이 4가지 상태를 가지며, 그림4 와 같이 시간이 지남에 따라 이들간에 상태전이가 발생한다. 다각형이 동적(dynamic)이라는 것은 이전 프레임과 현 프레임 사이에 기하정보나 텍스쳐, 색상등 다각형의 외모를 결정하는 요인들이 바뀌는 것을 말한다. 한 다각형이 동적인지의 여부는 다각형에 이전 프레임과 현 프레임 사이에 생성, 파괴, 이동, 텍스쳐 변경, 색상변화 등의 연산들이 가해질 때 표시를 하고 현 프레임이 렌더링 되기 직전에 검사하는 것으로 쉽게 알아 낼 수 있다.

표 1 시간 변화에 따라 다각형들이 가질 수 있는 4가지 상태

	$t-1$ 에서의 상태	t 에서의 상태
P_{dd}	dynamic	dynamic
P_{ds}	dynamic	static
P_{sd}	static	dynamic
P_{ss}	static	static

기본적인 렌더링 방법은 4절의 방법과 유사하다. 즉, 이전 프레임에서 계산된 정적 버퍼쌍의 값을 이용하여 현재 프레임의 정적 다각형들을 렌더링한 결과를 얻어내고, 여기에 동적 다각형들을 렌더링하여 최종 결과를 얻는다. 단지 4절에서는 정적 버퍼쌍의 값이 시점과 시선이 고정된 동안 불변이지만, 여기서는 매 프레임마다 갱신해주어야 한다는 점이 다르다.

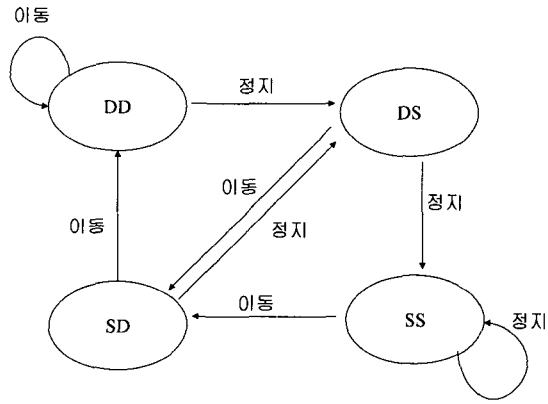

그림 4 다각형의 상태 전이도

5.1 정적 다각형의 색상과 깊이 정보로 렌더링 버퍼쌍을 갱신하는 단계

현재 프레임 f_t에서의 렌더링을 시작하기 전에 렌더링 버퍼 쌍에는 프레임 f_{t-1}에 가시적인 모든 다각형들의 색상과 깊이 정보인 $\langle RC^{t-1}_{P}, RZ^{t-1}_{P}\rangle$가 저장되어있고, 정적 버퍼 쌍에는 f_{t-1}에서 정적이었던 다각형들만 렌더링된 결과인 $\langle SC^{t-1}_{P_{sd}\cup P_{ss}}, RZ^{t-1}_{P_{sd}\cup P_{ss}}\rangle$가 저장되어 있다. 이를 바탕으로 렌더링 버퍼 쌍의 내용을 f_t에서 정적인 다각형만 렌더링한 결과인 $\langle RC^{t}_{P_{sd}\cup P_{ss}}, RZ^{t}_{P_{sd}\cup P_{ss}}\rangle$로 만들어 주어야 한다. 이를 위해 다음 네 가지 과정을 거친다.

첫째로 4절의 알고리즘에서와 같이 정적 버퍼쌍에서 P_{dd} 다각형들의 화상 영역에 해당하는 부분을 렌더링 버퍼 쌍으로 복사한다. P_{dd} 다각형들은 f_{t-1}에서 정적 버퍼 쌍에 그려져 있지 않고, f_t에서의 정적 버퍼 쌍에도 그려지지 말아야 한다. 그런데 f_{t-1}에서의 렌더링 버퍼 쌍 값인 $\langle RC^{t-1}_{P}, RZ^{t-1}_{P}\rangle$에는 P_{dd} 다각형까지 이미 그려져 있는 상태이므로 정적 버퍼 쌍으로부터 렌더링 버퍼 쌍으로 P_{dd} 다각형들의 화상 영역에 해당되는 부분을 복사하면 렌더링 버퍼 쌍의 내용이 P_{dd}다각형을 렌더링하기 전 상태로 된다.

둘째로 P_{sd} 다각형들의 화상 영역에 해당되는 부분을 무효화 시켜준다. 무효화 (invalidation)란 버퍼쌍의 일부를 배경 색과 배경 깊이 값으로 채우는 연산을 가리

킨다. P_{sd} 다각형들은 f_{t-1}에서의 정적 버퍼 쌍에 그려져 있어야 하고 f_t에는 정적 버퍼 쌍에 그려지지 말아야 한다. 그런데 f_{t-1}에서의 정적 버퍼 쌍과 렌더링 버퍼 쌍 모두에 P_{sd} 다각형들이 그려진 상태이므로 이들을 복사하는 것은 의미가 없다. 따라서 P_{sd} 다각형들의 화상 영역을 무효화 해주고, 이 화상 영역들과 겹치는 정적 다각형들을 세 번째와 네 번째 단계에서 다시 렌더링 해야 한다.

세번째로 P_{ss} 다각형들의 영역을 그린다. P_{ss}는 렌더링 버퍼 쌍과 정적 버퍼 쌍에 모두 렌더링 되어있다. 또한 현재 프레임 f_t의 정적 버퍼 쌍에 렌더링되어야 한다. 그러므로 P_{ss} 다각형만 고려한다면 이 단계에서 렌더링 버퍼의 내용을 변경할 필요가 없다. 그러나 이전 단계에서 P_{sd} 다각형들의 영역을 무효화시키면서 이들이 P_{ss} 다각형과 겹치는 부분은 함께 무효화 될 수 있다. 따라서 P_{ss} 다각형들 중에서 그 화상 영역이 P_{sd} 다각형의 화상 영역과 겹치는 것이 있다면 그 다각형을 다시 렌더링 해주어야 한다. P_{ss}가 P_{dd}와 겹치는 부분은 첫단계에서 정적 버퍼 쌍으로부터 올바른 정보를 복사하므로 다시 렌더링할 필요가 없다.

마지막으로 f_{t-1}에서 동적이었지만 f_t에서는 정적인 P_{ds}다각형들을 처리한다. f_t에서 정적이므로 P_{ds}인 다각형들만 고려한다면 그대로 두면 된다. 하지만 화상영역이 P_{sd}나 P_{dd} 과 겹치는 다각형의 경우 P_{sd}의 화상 영역을 무효화하거나 P_{dd}의 화상 영역에 해당하는 정적 버퍼쌍의 내용을 복사하는 과정에서 함께 무효화되거나 정적 버퍼쌍의 내용으로 덮어 씌어지게 된다. 그러므로 P_{ds}중에서 P_{dd}나 P_{sd}와 겹치는 다각형들은 다시 렌더링 해야한다.

5.2 렌더링 버퍼 쌍의 내용으로 정적 버퍼 쌍의 내용을 갱신하는 단계

5.1 단계를 마치면 렌더링 버퍼 쌍에는 현재 프레임에 정적인 다각형들이 렌더링된 정보인 $\langle RC^t_{P_{sd}\cup P_{ss}}, RZ^t_{P_{sd}\cup P_{ss}}\rangle$가 저장된다. 정적 버퍼 쌍에는 이전 프레임에서 정적인 다각형들의 정보가 있으므로 둘 사이에는 P_{ds} 와 P_{sd} 다각형들의 화상 영역 내에서만 다르다. 따라서 렌더링 버퍼쌍의 모든 값을 복사하는 대신에 P_{ds} 와 P_{sd} 화상 영역만 정적 버퍼 쌍으로 복사하면 정적 버퍼쌍에는 현재 프레임에서 정적인 모든 다각형들의 정보가 저장되고, 이것은 다음 프레임에서 다시 사용된다.

5.3 동적 다각형들의 렌단계

렌더링 버퍼 쌍에는 현재 프레임에서 정적인 다각형을 z-버퍼 알고리즘으로 렌더링한깊이 정보가 들어있다. 여기에 동적인 다각형들을 가지고 z-버퍼 알고리즘을 적용하면 최종 화상을 얻을 수 있다. 그림 5는 다각형들의 이동상태가 결정되지 않은 경우의 가속화된 알고리즘을 설명하고 있다.

* 동적 다각형과 정적 다각형이 결정된 경우와 결정되지 않은 경우의 비교

4절에 설명된 알고리즘은 SS와 DD인 다각형들만 존재하고 각 다각형에는 SS인지 DD인지에 대한 정보가 저장되어 있어야 사용될 수 있다. SS와 DD인 다각형들로만 이루어진 모델에서 5절에서 설명될 방법을 적용해도 4절의 알고리즘과 같은 수의 렌더링과 복사연산이 이루어진다. 즉 $P_{ds}=P_{sd}=\phi$ 이므로 5.1의 둘째, 셋째, 넷째 단계가 필요 없어진다. 또한 5.2의 복사연산도 필요없어진다. 즉 5.1의 동적 다각형들의 화상영역의 복사와 5.3의 동적 다각형의 렌더링만 수행하게 되어 4절에서 하는 일과 같아진다.

```
Realtime_Render(POLYGON P)
{
    StaticBufferValid = false;
    while(ContinueToRender){
        UpdateWorldInfo(); // 시점과 각 다각형의 기하, 색상
                           정보변화가 일어난다.
                         // 기하, 색상 정보가 변하는 다각
                           형은 동적, 그렇지 않은
                         // 다각형은 정적으로 표시된다.

        if(시점이나 시선이 변경){
        // 시점과 시선이 계속 변경하는 경우는 기존 z-버퍼
           알고리즘과 동일하다.
           StaticBufferValid = false;
           ZBuffer( Pss∪Psd∪Pds∪Pdd);
        }
        else { // 시점과 시선이 고정
            if(StaticBufferValid){      // 시점과 시선이 계속
                                           고정된 경우
              CopyInvalidExtent( Pdd);
              // 정적 버퍼쌍에서 동적 다각형의 화상영역에
                 해당하는 부분을
              // 렌더링 버퍼쌍으로 복사
              InvalidateExtent( Psd); // Psd다각형의 화상
                                         영역를 무효화
              ZBuffer(Intersect( Pss, Psd));
              // Psd와 겹치는 Pss를 렌더링
              ZBuffer(Intersect( Pds, Pdd) ∪ Intersect
                                          ( Pds, Psd));
              // Psd나 Pdd와 겹치는 Pds를 렌더링
              CopyToStaticBuffer( Psd, Pds);
              // 렌더링 버퍼쌍의 내용중 Psd, Pds에 해당하
                 는 부분을
              // 정적 버퍼쌍으로 복사
            }
            else{       // 시점과 시선이 고정된 직후
```

```
            ZBuffer( Pss∪Pds );     // 정적 다각형만 렌더링
            CopyToStaticBuffer();// 렌더링 버퍼쌍의 내
                                    용을 정적 버퍼쌍으
                                    로 복사
            StaticBufferValid = true;
        }
        ZBuffer( Psd∪Pdd );     // 동적 다각형만 렌더링
    }
    ShowImage();      // 화상 출력
  }
}
```

그림 5 다각형들의 이동상태가 동적으로 결정되는 경우의 효율적인 렌더링 알고리즘

6. 실 험

6.1 이론적 성능 평가

6.1.1 처리 속도

다각형 하나를 렌더링하는데 소요되는 평균 시간을 t_r, 렌더링 버퍼쌍 내의 한 화상 영역을 무효화하는데 소요되는 평균 시간을 t_i, 렌더링 버퍼 쌍과 정적 버퍼쌍 사이에 한 영역을 복사하는데 걸리는 평균 시간을 t_c, 버퍼쌍영역 전체를 무효화 하는데 소요되는 시간을 t_w라 하면 기존 방법의 실행 시간 t_{old}과 제안된 방법의 수행 시간 t_{new}는 다음과 같다.

$$t_{old} = t_w + t_r \times (|P_{dd}| + |P_{sd}| + |P_{ds}| + |P_{ss}|) \quad (1)$$

$$\begin{aligned} t_{new} &= t_c \times |P_{dd}| + t_i \times |P_{sd}| + t_r \times |(P_{ss} \otimes P_{sd})| \\ &\quad + t_r \times |(P_{ds} \otimes (P_{sd} \cup P_{dd}))| + t_c \times (|P_{sd}| + |P_{ds}|) + t_r \times |P_{dd}| \\ &= t_c \times (|P_{dd}| + |P_{sd}| + |P_{ds}|) + t_i \times |P_{sd}| \\ &\quad + t_r \times (|(P_{ss} \otimes P_{sd})| + |(P_{ds} \otimes (P_{sd} \cup P_{dd}))| + |P_{dd}|) \end{aligned}$$

식1 에서 $|P|$는 다각형의 집합 P에 포함되는 다각형의 수이고, $P_A \otimes P_B$는 P_A의 원소들 중에서 P_B의 원

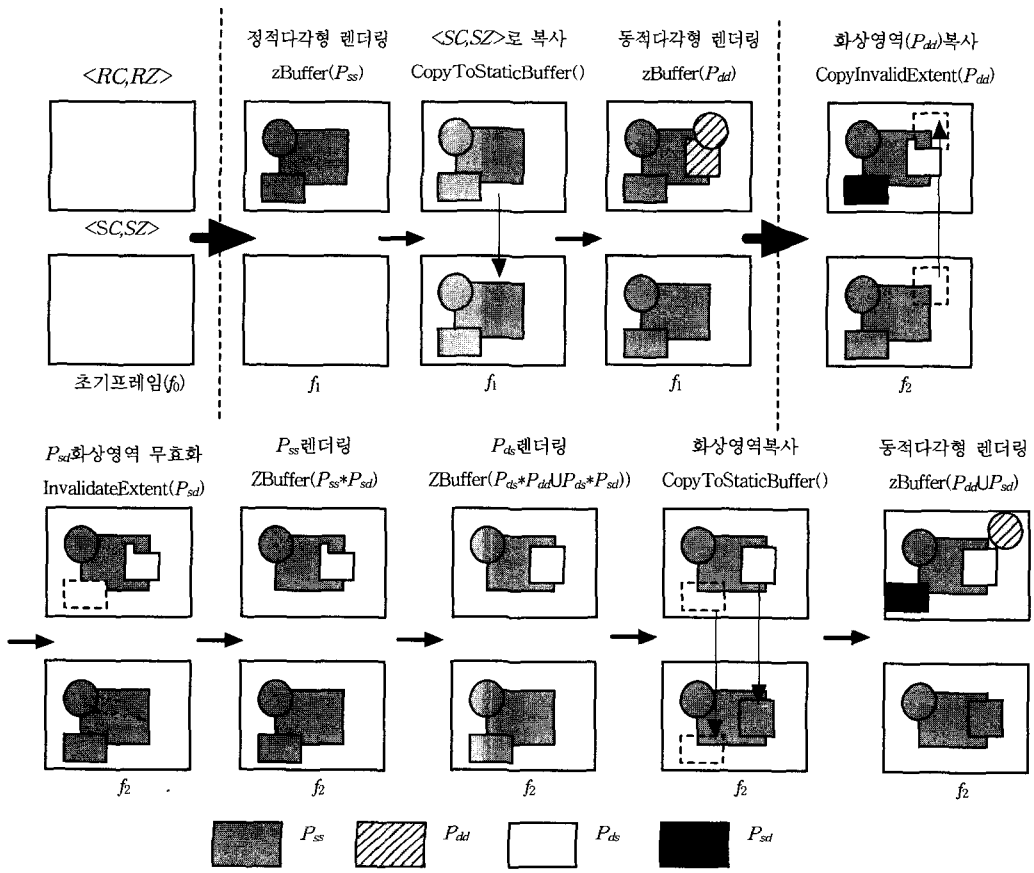

그림 6 다각형들의 이동상태가 동적으로 결정되는 경우의 렌더링 예

소들과 화상영역이 교차하는 원소들의 집합을 얻어내는 연산자이다. 예를 들어 $P=P_{ss}\otimes P_{sd}$ 일 때, P는 P_{ss}다각형들 중에서 그 화상영역이 P_{sd}다각형들과 교차하는 것들의 집합이다. 이 집합을 계산하는데 드는 시간은 특별한 방법을 쓰지 않고 모든 다각형들 사이의 화상영역을 검사한다고 가정할 때 $t_{intersec}*|P_A|*|P_B|$이다. 여기서 $t_{intersec}$는 한 쌍의 다각형들의 화상영역이 겹치는지 검사하는데 걸리는 시간으로 두 점으로 정의된 사각형 두 개 사이의 교차성 검사에 걸리는 시간과 같다.

그러므로 제안된 방법의 수행시간에서 ⊗연산의 계산시간까지 고려한다면 다음과 같다.

$$\begin{aligned} t_{new} &= t_c\times(|P_{dd}|+|P_{sd}|+|P_{ds}|)+t_i\times|P_{sd}| \\ &+ t_r\times(|(P_{ss}\otimes P_{sd})|+|(P_{ds}\otimes(P_{sd}\cup P_{dd}))|+|P_{dd}|) \\ &+ t_{intersec}*|P_{ss}|*|P_{sd}|+t_{intersec}*|P_{ds}|*|P_{sd}\cup P_{dd}| \end{aligned} \quad (2)$$

위의 두 식에서 $t_{intersec}, t_i, t_c \ll t_r$ 이므로 t_r의 계수가 성능을 좌우한다. 기존 방법에서 t_r의 계수는 $|P_{dd}|+|P_{sd}|+|P_{ds}|+|P_{ss}|$이고, 제안된 방법에서는 $|P_{ss}\otimes P_{sd}|+|P_{ds}\otimes(P_{sd}\cup P_{dd})|+|P_{dd}|$이다. $|P_{ss}|\geq|P_{ss}\otimes P_{sd}|$이고 $|P_{ds}|\geq|P_{ds}\otimes(P_{sd}\cup P_{dd})|$ 이므로 제안 된 방법이 기존 방법에 비해서 속도가 빨라진다. 특히 전체 다각형수에 대한 정적 다각형 수의 비가 클수록 보다 큰 속도향상을 기대할 수 있다.

6.1.2 기억공간

기존 z-버퍼에서는 렌더링 버퍼쌍만을 사용하지만, 제안된 방법에서는 렌더링 버퍼쌍 외에 동일한 크기의 정적 버퍼쌍이 사용되기 때문에 기억공간은 기존 방법의 2배가 소요된다.

6.2 실험적 성능평가

이론적인 성능향상이 실제 응용에서도 발생하는지 확인하기 위하여 다수의 다각형들로 이루어진 가상 환경을 구축하여 실험하였다. 적용 대상은 3차원 도미노 게임 프로그램이며, 실험에는 두 개의 모델이 사용되었다. 첫 번째 다각형 모델은 그림 7과 같으며 804개의 다각형으로 이루어져 있고, 이중 55%정도만이 정적인 상태를 유지한다. 두 번째 다각형 모델은 그림 8과 같으며 1404개의 다각형들로 이루어져 있으며 쓰러지는 도미노를 구성하는 다각형들을 제외하고 대부분의 다각형들이 정적 상태를 유지한다. 실험은 Pentium II 266MHz CPU 와 64MB의 메모리를 장착한 개인용 컴퓨터 상에서 실시하였고 별도의 그래픽 하드웨어는 사용하지 않았다. 실험에서는 본 논문에서 제안하는 방법이 기존 방법보다 상당한 속도향상이 발생하는지의 여부와, 정적 다각형의 비율이 높을수록 렌더링 속도가 실제로 더 빨라지는지를 알아보았다.

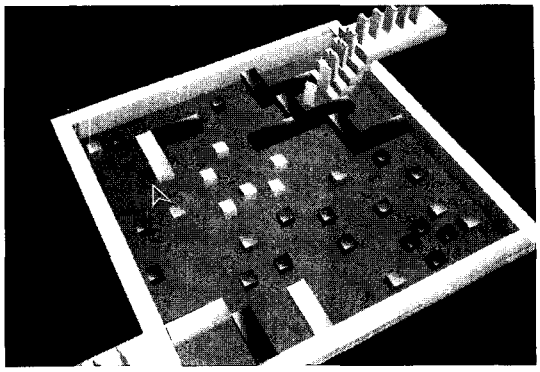

그림 7 실험 대상 모델 I (다각형 수 = 804)

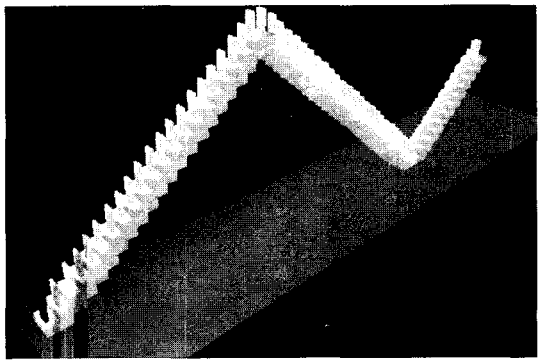

그림 8 실험 대상 모델 II (다각형 수 = 1404)

그림 9은 실험 대상 모델 I을 연속된 50개의 프레임에서 렌더링한 후 각 프레임을 렌더링하는데 소요된 시간을 측정한 것이다. 모델 I에서 전체 다각형에 대한 정적 다각형(P_{ss})의 프레임당 평균 비율은 55.44%이고, 동적 다각형 (P_{ds}, P_{sd}, P_{dd})은 44.56%이다. 실험 결과에서 볼 수 있듯이 기존 z버퍼 알고리즘을 사용하는 경우 프레임당 평균 렌더링 시간은 316msec인데 반해, 가속화된 방법을 사용할 경우는 170msec가 소요되어, 결과적으로 속도가 두 배정도 빨라지는 것을 확인할 수 있었다.

그림 10은 제안된 방법에서 다각형들의 상태와 렌더링 시간과의 관계를 나타낸 것이다. 결과를 보면 동적 다각형이 증가 (정적 다각형이 감소) 할수록 렌더링 시간이 다소 증가되는 것을 확인 할 수 있다.

그림 11는 모델 II을 연속된 50개의 프레임에서 렌더링한 후 각 프레임을 렌더링하는데 소요된 시간을 측정한 것이다. 모델 II에서는 전체 다각형에 대한 정적 다

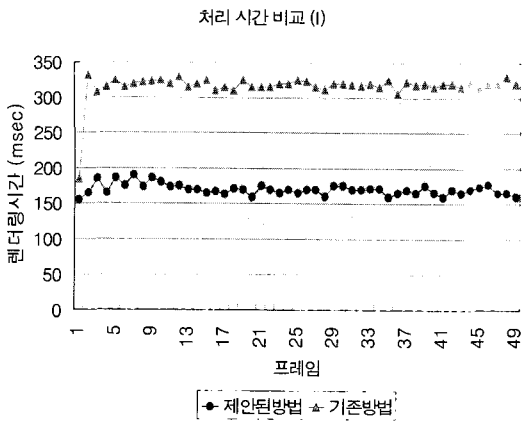

그림 9 모델 I에서의 프레임별 렌더링 시간 비교

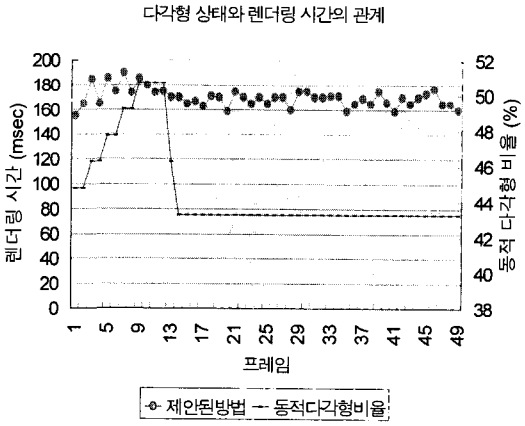

그림 10 다각형의 상태와 렌더링 시간과의 관계

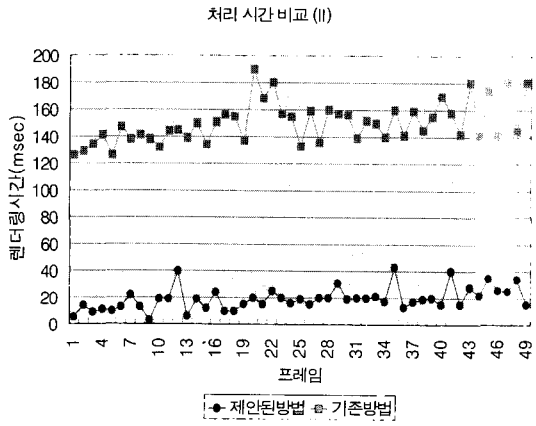

그림 11 모델 II에서의 프레임별 렌더링 시간 비교

각형(P_{ss})의 프레임당 평균 비율이 98.66%로서 거의 대부분의 다각형들이 고정되어 있다. 실험 결과에서 기존 z버퍼 알고리즘을 사용하는 경우 프레임당 평균 렌더링 시간은 158msec인데 반해, 가속화된 방법을 사용할 경우는 19msec가 소요되어 기존 방법에비해 8배이상 속도 빨라진다.

위의 실험 결과에서 새로 제안된 방법을 사용할 경우 렌더링 시간이 기존 방법을 사용할 경우보다 상당히 단축되는 것을 확인할 수 있었다. 또한 이러한 성능향상은 전체 다각형 수에대한 정적 다각형 수의 비율이 증가될수록 커진다는 사실을 확인하였다.

7. 결 론

z버퍼 알고리즘의 속도를 개선하기 위해서 정적 다각형의 깊이, 색상 정보를 렌더링하지 않고 효율적으로 얻어내는 방법을 제안하였다. 이 방법은 최악의 경우에도 기존 방법과 같은 수행속도를 가지며, 시점이 고정된 경우 수행 속도가 동적 다각형의 개수에만 비례한다. 따라서 시점이나 시선이 자주 변하지 않고 전체 다각형의 개수에 비해 동적 다각형의 개수가 적은 응용에 적합하다.

향후 연구는 시점의 변화를 제한적으로 허용하면서도 빠른 렌더링 속도를 유지할 수 있는 방법을 고안하는데 초점을 맞출 것이다. 이 논문에서 제안하는 방법을 화상 기반 렌더링 (image based rendering) 기술[16]과 결합하면 시점 이동을 허용하면서도 동적 다각형만을 실시간에 렌더링하도록 할 수 있다. 화상 기반 렌더링은 기하 모델 없이 화상만으로 렌더링을 가능하게 해주어서 여러 가지 장점을 갖는 반면 변화하는 모델을 처리할 수 없는 단점이 있다. 변화하는 모델을 화상 기반 렌더링으로 렌더링 하려면 렌더링의 입력이 되는 화상들을 매 프레임에 계속 만들어 주면 된다. 그러나 일반적으로 렌더링에 입력되는 화상을 만드는데 더 오랜 시간이 걸린다. 본 논문에서 소개한 알고리즘은 모델 전체는 복잡하더라도 동적인 다각형들의 복잡도가 작은 경우 많은 속도향상을 보이므로 동적 다각형이 적은 상황에서 화상 기반 렌더링에 입력이 되는 화상들을 실시간에 만들어 내는 것을 가능하게 한다.

또 한가지는 하드웨어를 사용하여 속도를 개선하는 것이다. 하드웨어로 구현할 경우 z-버퍼 기능만 제공하는 하드웨어를 사용하는 경우에 비해 CPU와 그래픽 칩의 상호작용을 덜 수 있게 되어 효율적인 렌더링이 가능하다.

참 고 문 헌

[1] Foley, J. D., Dam, A. V., Feiner S. K. and Hughes, J. F., Computer Graphics principles and practice, pp 668, Addison-Wesley, 1995

[2] Rohlf, J. and Helman, J., "IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics," *Proc. of ACM SIGGRAPH94*, pp. 381-394. 1994.

[3] Akeley, K., "Reality Engine graphics," *Proc. of ACM SIGGRAP93*, pp. 109-106, 1993.

[4] Pineda, J. "A Parallel Algorithm for Polygon Rasterization," *Computer Graphics*, Vol. 22, No. 4, pp. 17-20, 1988.

[5] Teller, S. J., "Visibility Computations in Densely Occluded Polyhedral Environments," PhD thesis in UC Berkeley, 1992.

[6] Green, N., Kass, M. and Miller, G., "Hierarchical Z-Buffer Visibility," *Proc. of ACM SIGGRAPH93*, pp. 231-238, 1993.

[7] Greene, N., "Hierarchical polygon tiling with coverage masks," Proc. of SIGGRAPH96, 65-74, 1996

[8] Zhang, H., Manocha, D., Hudson, T. and Hoff, K. E. III, "Visibility Culling using Hierarchical Occlusion Maps," *Proc. of ACM SIGGRAPH97*, pp. 77-88, 1997.

[9] Catmull, E., "A Subdivision Algorithm for Computer Display of Curved Surfaces," Ph.D. Thesis, Computer Science Department, University of Utah, 1974.

[10] Hubschman, H. and Zucker, S. W., "Frame to frame coherence and the hidden surface computation constraints for a convex world," ACM Transactions on Graphics, 1(2):129-162, 1982

[11] Coorg, S. and Teller, S., "Temporally coherent conservative visibility," Proc. 12th ACM symposium on Computational Geometry, 1996

[12] Jevans, D., "Object space temporal coherence for ray tracing," Proc. Graphics Interface '92, 176-183, 1992

[13] Chapman, J., Calvert, T.W. and Dill, J. "Spatial-temporal coherence in ray tracing," Proc. Graphics Interface '90, 196-204, 1990

[14] Badt, S. J., "Two algorithms for taking advantage of temporal coherence in ray tracing," The Visual Computer, 4:123-132, 1988

[15] Microsoft, DirectX6.0 API Specification, Microsoft Corporation, Redmond WA, 1997.

[16] McMillan, L. and Bishop, G., "Plenoptic Modeling: An Image Based Rendering system," *Proc. of SIGGRAPH95*, pp. 39-46, 1995.

오 경 수

1994년 서울대학교 계산통계학과 졸업. 1996년 서울대학교 전산과학과 졸업(이학석사). 1996년 ~ 현재 서울대학교 전산과학과 박사과정. 관심분야는 실시간 렌더링, 컴퓨터 게임, 화상기반 렌더링(image based rendering), VRML, 가상현실

신 영 길

1982년 서울대학교 계산통계학과 졸업(학사). 1984년 서울대학교 계산통계학과 석사학위 취득. 1989년 University of Southern California 박사학위 취득. 1990년 ~ 1992년 경북대학교 전임강사. 1992년 ~ 현재 서울대학교 전산과학과 조교수. 관심분야는 컴퓨터그래픽스, CAD, 인공지능, 멀티미디어 등 임.

신 병 석

1990년 2월 서울대학교 컴퓨터공학과 졸업(공학사). 1992년 2월 서울대학교 컴퓨터공학과 졸업(공학석사). 1997년 2월 서울대학교 컴퓨터공학과 졸업(공학박사). 1997년 ~ 현재 주식회사 비트컴퓨터 의료영상 연구실장으로 재직중. 관심분야는 컴퓨터 그래픽스, CAD