論文 89-26-12-10

실시간 처리를 위한 3차원 그래픽스 은폐면 제거

(Hidden-Surface Removal in 3D Graphics for Real Time Animation)

金 應 坤*, 朴 鍾 安**, 宋 鐵***, 金 俊 炫**

(Eung Kon Kim, Jong An Park, Chol Song, and Joon Hyun Kim)

要 約

본 논문에서는 3차원 물체의 실시간 애니메이션을 위한 은폐면(hiddensurface) 제거 알고리듬을 제안한다. 제안된 알고리듬은 물체를 다각형의 집합으로 표현한 다면체에 대하여 처리할 수 있으며, 처리 속도가 빠르고 볼록 다면체 뿐만 아니라 오목 다면체에 대해서도 적용 가능하다. 또한 한 물체가 다른 물체를 통과하는 형태를 갖는 물체의 경우에도 통과하는 부분의 면을 여러 개로 분할하여 표현함으로써 적용 가능하다.

개발된 알고리듬은 IRIS 워크스테이션상에서 C언어를 사용하여 구현되었으며, 항공기를 예로 하여 비행 항적에 따라 비행과정을 시뮬레이션한 결과로써 그 유용성을 보인다.

Abstract

This paper proposes the hidden surface removal algorithm for real time animation. The proposed algorithm can process concave polyhedrons in real time and apply to an object that penetrates others by dividing a surface into multiple surfaces. The algorithm has been implemented on IRIS workstation running Berkeley 4.2 UNIX and the program has been written in C language. The effectiveness of this algorithm is shown by flight simulation of the aircraft with several flight paths.

Ⅰ. 서 론

Flight 시뮬레이션[1,2]을 비롯한 3차원 물체의 실시간 애니메이션을 위해서는 수많은 3차원 좌표 변환을 실시간으로 처리해야 하기 때문에 이에 대한 계산 속도를 올리기 위하여 좌표 변환 하드웨어가 별도로 필요하다.

최근 컴퓨터 그래픽스 하드웨어는 마이크로 프로세서의 출현과 저렴한 고밀도 메모리 소자의 개발로 눈부신 발전을 하였으며, 3차원 그래픽스의 고속처리를 위한 하드웨어인 geometry engine이 개발되어 3차원 물체의 애니메이션 및 실시간 시뮬레이션에 대한 연구가 활발히 진행되고 있다.

3차원 그래픽에서 해결해야 할 커다란 과제중의 하나는 물체를 표현하는데 있어서 실제감을 갖도록 하기 위한 은폐선(hidden line)이나 은폐면(hidden surface) 제거를 실시간으로 처리하는 문제이다. 은폐선이나 은폐면을 제거하는 알고리듬은 많은 연산 시간과 기억 메모리를 필요로 하기 때문이다. 특히 3

*正會員, 國防科學研究所
(Agency for Defence Development)
**正會員, 朝鮮大學校 電子工學科
(Dept. of Elec. Eng., Chosun Univ.)
***正會員, 朝鮮大學校 電算機工學科
(Dept. of Comput. Eng., Chosun Univ.)

接受日字 : 1989年 7月 24日

차원 물체의 실시간 시뮬레이션 및 애니메이션 분야에서는 초당 24회 이상 은폐선이나 은폐면을 제거하여 디스플레이해야 하는데 기존의 알고리듬으로는 복잡한 형태의 물체에 대하여 실시간 처리가 불가능하므로 실시간 처리용 알고리듬이 절실히 요구되고 있다.

물체를 wireframe 모델로 표현한 경우에는 은폐선을 제거해야 하며, solid 모델로 표현한 경우에는 은폐면을 제거해야 하는데 여기서는 볼록형 물체 뿐만 아니라 오목형 물체를 다각형의 집합으로 표현한 solid 모델에 대한 은폐면 제거를 다룬다.

은폐면 제거의 처리 속도는 물체의 표현 방법, 모델링의 정확도, 컴퓨터의 계산 능력 및 적용한 알고리듬에 따라 크게 좌우된다. 기존의 알고리듬 중 처리 속도가 빠른 것은 오목 다면체와 같은 복잡한 물체에 대해서는 적용할 수 없으며, 모든 물체에 대해서 적용할 수 있는 알고리듬은 처리 속도가 너무 느리므로 실시간 처리에 적합하지 않다.

본 논문에서는 오목한 형태를 포함한 물체에 대해서도 실시간 처리가 가능한 은폐면 제거 알고리듬을 제안한다. 항공기를 50개 내외의 면으로 구성하여비행 항적에 따라 비행 과정을 시뮬레이션한 결과로써 그 유용성을 보인다.

Ⅱ. 기존의 은폐면 제거 알고리듬

은폐선이나 은폐면을 제거하는 알고리듬[3,4,5,6]은 크게 두 가지로 구분된다. 즉 물체를 스크린에 투영한 영상을 가지고 은폐면을 가려내는 image space method를 이용한 알고리듬과 투영하기 이전에 실제 물체를 구성하고 있는 다각형과 꼭지점의 좌표 정보를 가지고 은폐면을 가려내는 object space method를 이용하는 알고리듬이 있다. 많은 은폐면 제거 알고리듬들은 image space method를 이용하고 있으나, 처리 속도를 고려할 경우는 object space method를 이용한 방법이 효율적이므로 응용 분야에 따라서 적절한 알고리듬을 적용해야 한다.

Object space method를 이용하는 대표적인 알고리듬으로는 Back Face Removal 알고리듬이 있으며, image space method를 이용하는 알고리듬으로는 Z-Buffer 알고리듬이 있다. Back Face Removal 알고리듬은 물체를 그리기 전에 관측점에서 물체를 구성하고 있는 각 면이 보이는 면인지 아닌지를 구별하여 보이지 않는 면은 그리지 않도록 한다. 그림 1과 같은 물체에 대하여 이 알고리듬을 적용할 경우 관측방향이 물체의 정면 우측상부일때 면ABCD 전체

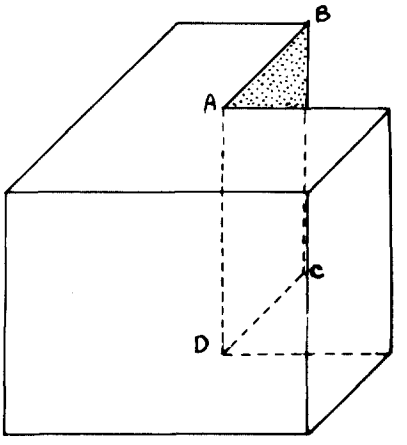


그림 1. 오목형 물체
Fig. 1. Concave object.

를 보이는 면으로 간주하므로 면ABCD 전체가 디스플레이될 수 있다. 따라서 이 알고리듬은 보이는 면과 안보이는 면을 가려낼 뿐이지 그림 1과 같은 오목형 물체에 대해서 면ABCD의 일부만이 보이는 경우 부분적으로 보이지 않는 부분은 처리할 수 없다.

따라서 오목형 물체의 경우에는 다른 알고리듬을 이용해야 한다. 특히 항공기도 오목한 형태를 갖고 있으므로 이 알고리듬을 그대로 적용할 수 없다. 그러나 이 알고리듬의 장점은 처리 속도가 빠르다는 점이다.

Z-Buffer 알고리듬은 매우 간단한 알고리듬이면서 오목한 물체나 볼록한 물체등 모든 물체에 대하여 적용 가능하다. 화면상의 각 화소에 대하여 관측자에게 가장 가까운 면을 찾아 그면의 색깔을 그 화소의 위치에 그리는 알고리듬으로 모든 면의 내부의 각 점(x, y)에 대하여 다음의 (1)~(4) 단계를 수행한다. 우선 스크린상의 각 화소 위치(x, y)에 대하여 화소의 색깔을 저장하는 버퍼인 INTENSITY 배열과 각 화소 위치에서 관측자에게 가장 가까운 점에 대한 Zs(Z좌표의 스크린 좌표)의 값을 갖는 버퍼인 DEPTH 배열이 필요하다.

(1) 각 면의 평면 방정식을 스크린 좌표(Xs, Xs, Zs)의 항으로 구한다.

$$AXs+BYs+CZs+D=0$$

(2) 화소 위치(x, x)에서 면의 깊이 Zs를 구한다.

$$Zs=(-D-AXs-BYs)/C$$

(3) Zs<DEPTH[x, y]를 만족하면 DEPTH[x, y]의 값을 Zs로 하고, INTENSITY[x, y] 값을 그 면의 color 값으로 한다.

(4) 모든 면에 대하여 (1)~(3)의 과정을 수행한 후 INTENSITY 배열을 디스플레이 버퍼로하여 모든 화소를 그린다.

이 알고리듬의 단점은 모든 면의 화소에 대하여 (1)~(4)의 과정을 수행해야 하므로 많은 시간과 메모리가 필요하다는 점이다. 따라서 Z-Buffer 알고리듬은 실시간 처리에는 적용할 수 없다.

두 알고리듬을 적용하여 43개의 면으로 이루어진 F-15기와 64개의 면으로 이루어진 500MD기에 대하여 실험한 결과는 표 1과 같다. 특히 IRIS 워크스테이션에서는 Z-Buffer 알고리듬을 위하여 하드웨어로써 16 bitplane이 제공되고, Z-Buffer 알고리듬과 Back Face Removal 알고리듬을 라이브러리 함수로 내장하고 있으므로 이것을 이용하여 프로그래밍하여 비교하였다.

표 1. 기존의 알고리듬 비교
Table 1. Comparison of two algorithms.

알고리듬	Back Face Removal	Z-Buffer
그리는 속도(F-15)	30 Frames/초	5 Frames/초
그리는 속도(500MD)	29 Frames/초	4 Frames/초
문제점	면의 일부가 다른 면에 가리는 부분은 처리 불가	수행 속도가 너무 느림

[시스템 : IRIS 워크스테이션]

Ⅲ. 실시간 처리를 위한 은폐면 제거 알고리듬

실시간 애니메이션 응용에서 가장 중요한 점은 처리 속도가 빠르면서 처리대상이 되는 물체에 대하여 부분적으로 가리는 면까지 모두 처리하여 그릴 수 있는 알고리듬을 적용해야 한다는 점이다.

본 논문에서는 실시간 애니메이션이 가능하도록 처리 속도를 빠르게 하기 위하여 3차원 물체를 다각형의 집합으로 근사화하여 표현하였으며, 물체를 그릴 때마다 보이는 면과 보이지 않는 면을 계산하지 않고 초기화 과정에서 미리 결정한 면의 그리는 순서에 따라서 그리도록 하였다.

즉 그림 2와 같이 물체의 관측 방향을 공간을 x, y, z의 3개의 축으로 나눈 8개(+++, ++−, +−+, +−−, −++, −+−, −−+, −−−)로 나누어 각 8개의 관측 방향에 대하여 부분적인 은폐면이 제거되어 그려지도록 면들의 그리는 순서를 미리 정한다. A면의 일부가 B면에 의해 가려질 경우 A면을 먼저 그리고 B면을 나중에 그리면 A면의 부분적인 은폐부분은 그려지지 않기 때문이다. 다음 물체의 이동에 따른 관측 방향을 check하여 해당 관측 방향에서 미리 정한 면들의 그리는 순서대로 면들을 그린다.

이 알고리듬을 간략하게 흐름도로 표현하면 그림 3과 같으며, 각 단계에서의 처리 내용을 기술하면 다음과 같다.

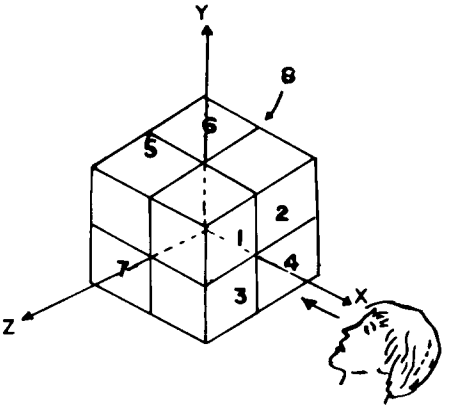


그림 2. 8개의 관측 방향
Fig. 2. 8 viewing directions.

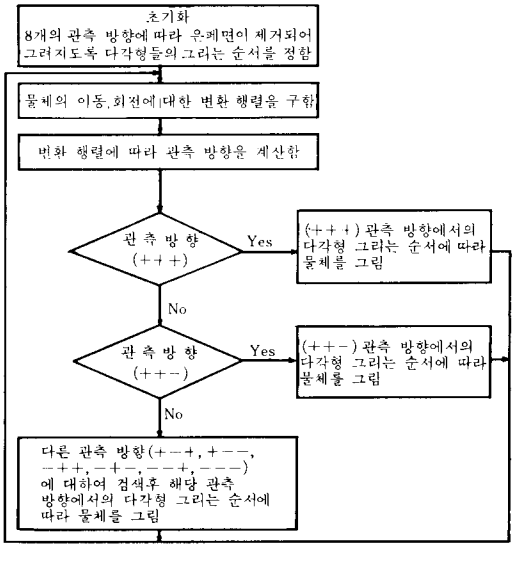


그림 3. 알고리듬 흐름도
Fig. 3. Flow chart of algorithm.

1. 초기화

관측점의 방향을 8개로 나누어 각 관측 방향에서 은폐면이 제거되어 그려지도록 면들의 그리는 순서를 정한다. 볼록형 다면체 뿐만 아니라 오목형 다면체에 대하여 처리가 가능하도록 초기화 과정에서 다음과 같이 미리 각 관측 방향에서 다각형들의 그리는 순서를 정한다.

단계 1) 관측 방향 1(+++)에 대하여 물체를 구성하고 있는 면들 중 보이지 않는 면을 찾아 내기 위하여 가시도(visibility)를 검사한다.[7]

관측 방향 1의 임의의 점 P에서 물체를 바라 보았을 때 각 면의 가시도를 검사하기 위하여 두개의 방향 벡터 즉 면에 대한 법선 벡터 n과 시선 벡터 ℓ을 그림4와 같이 정의한다. 법선 벡터n은 면에 수직이고 물체의 밖을 향하는 벡터이고, 시선 벡터 ℓ은 법선 벡터의 시작점에서 관측점 P를 향하는 벡터로 정의한다. 이 때 두 벡터 n과 ℓ이 이루는 각도 Φ를 계산하여 Φ 값이 0도와 90도 사이이면 보이는 면이며 따라서 디스플레이해야 하고, 90도 보다 크면 보이지 않는 면이므로 디스플레이할 필요가 없다. 임의의 면에 대한 법선 벡터를 구하기 위하여 그림4와 같이 면의 앞 쪽에서 바라 보았을 때 반시계 방향으로 각 꼭지점에 대하여 순서대로 번호를 정한다. 꼭지점 V1에서 V2로 향하는 벡터 u와 꼭지점 V1에서 꼭지점 V3로 향하는 벡터 v를 구하여 두 벡터의 외적 u×v를 구하면 법선 벡터 n이 된다.

$$n = u \times v = (bf - ce,\ cd - af,\ ae - bd) \qquad (1)$$

여기서 $u = (a, b, c)$, $v = (d, e, f)$

또한 시선 벡터 ℓ은 관측점 P(x1, y1, z1)와 법선 벡터의 시작점 V1(v1, v2, v3)으로 부터 다음과 같이 구해진다.

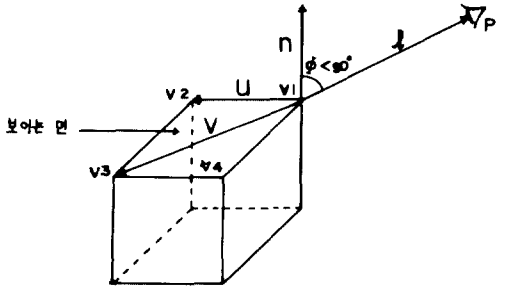


그림 4. 법선 벡터와 시선 벡터
Fig. 4. Normal and line of sight vector.

$$\ell = (x1,\ y1,\ z1,\ ) - (v1,\ v2,\ v3)$$
$$= (x1-v1,\ y1-v2,\ z1-v3) \qquad (2)$$

가시도를 검사하기 위하여 두 벡터가 이루는 각도를 구하면

$$n \cdot \ell = (n1,\ n2,\ n3) \cdot (\ell 1,\ \ell 2,\ \ell 3)$$
$$= n1\ell 1 + n2\ell 2 + n3\ell 3 \qquad (3)$$

$$n \cdot \ell = |n|\ |\ell\ | \cos \Phi$$

$$\Phi = \arccos(n \cdot \ell / (|n|\,|\ell\ |)) \qquad (4)$$

결국 보이는 면은 Φ가 0도에서 90도 사이 이므로 $n \cdot \ell$의 값이 0보다 크고, 보이지 않는 면의 $n \cdot \ell$의 값은 0보다 작게 되기 때문에 각도 Φ를 구할 필요없이 $n \cdot \ell$의 값만을 계산하여 가시도를 검사한다.

단계 2) 단계 1)에서 보이는 면으로 판정된 면들에 대하여 그리는 순서를 정하기 위하여 임의의 두 면을 선택하여 각 면을 구성하는 3D 좌표(X, Y, Z)에 대한 스크린 좌표(Xs, Ys, Zs)의 Zs 값에 대하여 다음 조건(5)와 (6)을 check한다. 그림 5에서 조건(5)와 (6)을 동시에 만족하면 A면이 B면보다 관측점에 가깝기 때문에 B면을 먼저 그리고 A면을 나중에 그리면 B면의 은폐 부분이 화면에 나타나지 않게 된다. ((7)과 (8)을 만족하면 반대)

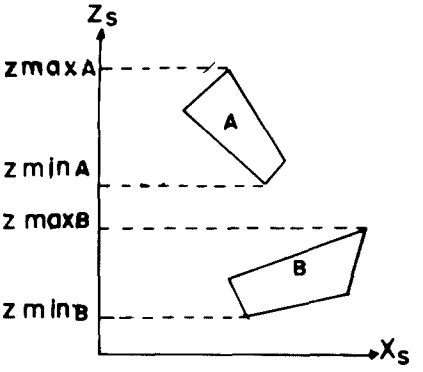


그림 5. 두 면의 깊이 비교
Fig. 5. Depth comparison of two surfaces.

$$ZmaxA > ZmaxB \qquad (5)$$

$$ZminA > ZminB \qquad (6)$$

$$ZmaxB > ZmaxA \qquad (7)$$

$$ZminB > ZminB \qquad (8)$$

단계 3) 단계2)에서 두 면 A, B가 조건 (5)와 (6), (7)과 (8)을 각각 만족하지 않으면 xy 평면상에서 두 면의 overlap을 검사한다. 두 면이 xy 평면에서 overlap하지 않으면 두 면의 그리는 순서는 어느 면을 먼저 그려도 무방하다. 그림 6에서 다음 조건 (9)~(12) 중 어느 하나를 만족하면 두 면은 반드시 overlap 하지 않는다.

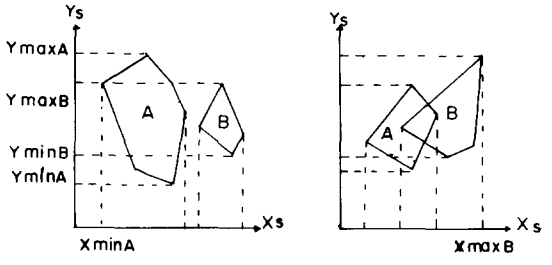


그림 6. 두 면의 x, y값 비교
Fig. 6. Comparison of the two values X and Y for a pair of surface.

$$XmaxA < XminB \quad (9)$$

$$XmaxB < XminA \quad (10)$$

$$YmaxA < YminB \quad (11)$$

$$YmaxB < YminA \quad (12)$$

조건 (9)~(12) 중 어느 하나도 만족하지 않으면 단계 (4)로 간다.

단계 4) 단계 3)에서 조건 (9)~(12) 중 어느 하나도 만족하지 않으면 두 면은 overlap하거나 그림 7·a와 같이 한 면이 다른 면을 포함할 경우이거나 그림 7·b와 같이 전혀 overlap하지 않을 경우가 있다. 그림 7·a의 경우는 단계 2)에서 두 면의 Zs 값을 비교하는 과정에서 고려가 되었으므로 여기서는 overlap 하는 경우와 그림 7·b의 경우를 처리하면 된다. 두 면이 overlap하는 경우는 두 면을 구성하는 선분들의 교차점을 구하여 그 점에서의 두 면에 대한 Zs 값을 비교하여 Zs 값이 큰 면이 관측점에 가까우므로 나중에 그리면 된다. 또한 두 면의 선분이 교차하지 않을 경우는 그림 7·b의 경우이므로 그리는 순서에는 무관하다. 두 면이 overlap하는 지의 검사와 검사후의 면들의 그리는 순서를 정하는 과정은 다음(가)-(라)와 같다.

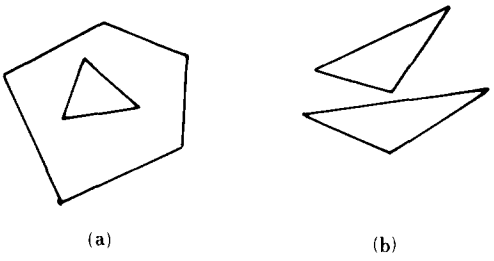


그림 7. 조건 (9)-(12)를 만족하지 않는 경우
Fig. 7. Examples that do not satisfy the conditions (9)-(12).

(가) A, B 두 면을 구성하는 선분중 임의의 선분을 각각 하나씩 선택하여 직선 방정식을 구한다.
(나) 두 직선이 일치하거나 평행하면 (가)로 간다.
(다) 두 선분의 교차점 I (Xi, Yi)를 구한다.
(라) 그림 8에서 다음 조건 (13)~(20) 중 어느 하나를 만족하면 두 선분은 점I에서 교차하며, 교차점에서의 두 면의 Zs 값을 구하기 위하여 (마)를 수행한다.

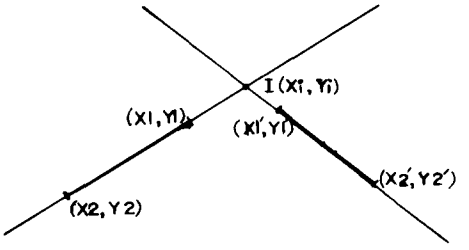


그림 8. 두 선분의 교차 검사
Fig. 8. Edge intersection test.

$$Xi < A1 \quad (13)$$

$$Xi > B1 \quad (14)$$

$$Xi < C1 \quad (15)$$

$$Xi > D1 \quad (16)$$

$$Yi < A2 \quad (17)$$

$$Yi > B2 \quad (18)$$

$$Yi < C2 \quad (19)$$

$$Yi > D2 \quad (20)$$

여기서

$A1=\min(X1, X2)$; $B1=\max(X1, X2)$
$C1=\min(X1', X2')$; $D1=\max(X1', X2')$
$A2=\min(Y1, Y2)$; $B2=\max(Y1, Y2)$
$C2=\min(Y1', Y2')$; $D2=\max(Y1', Y2')$

(마) (라)의 결과 두 선분이 점I에서 교차하면 두 면은 overlap하므로 점 I에서 두 면의 Zs 값을 구하여 Zs 값이 작은 면을 먼저 그리고 큰 면을 나중에 그린다. 교차점 I에서의 두 면의 Zs 값을 구하기 위하여 우선 두 면의 평면 방정식을 구한다. 평면 방정식 $aXs+bYs+cZs+d=0$의 a, b, c, d 값을 구하기 위하여 그림 9에서 평면을 구성하는 꼭지점으로 부터 두 벡터 u, v를 다음과 같이 구하여

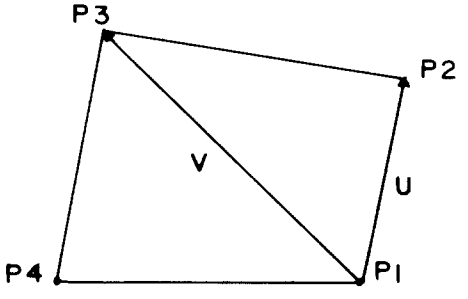


그림 9. 평면 방정식을 구하기 위한 그림
Fig. 9. An example for the plane equation.

$$u=p1p2=(X2, Y2, Z2)-(X1, Y1, Z1)$$
$$=(X2-X1, Y2-Y1, Z2-Z1) \quad (21)$$

$$v=P1P3=(X3, Y3, Z3)-(X1, Y1, Z1)$$
$$=(X3-X1, Y3-Y1, Z3-Z1) \quad (22)$$

두 벡터의 외적 $u\times v$로 부터 a, b, c, d를 다음과 같이 구한다.

$$(a, b, c)=u\times v \quad (23)$$

$$a=Y1(Z2-Z3)+Y2(Z3-Z1)+Y3(Z1-Z2) \quad (24)$$

$$b=Z1(X2-X3)+Z2(X3-X1)+Z3(X1-X2) \quad (25)$$

$$c=X1(Y2-Y3)+X2(Y3-Y1)+X3(Y1-Y2) \quad (26)$$

$$d=-aX1-bY1-cZ1 \quad (27)$$

A, B면의 평면 방정식을 각 각

$$a1Xs+b1Ys+c1Zs+d1=0 \quad (28)$$

$$a2Xs+b2Ys+c2Zs+d2=0 \quad (29)$$

라 하면 교차점 I에서의 Zs 값은 각각 식 (28), (29)에 $Xs=Xi$, $Ys=Yi$를 대입하여 구하면된다.

(바) 면을 구성하는 다른 선분에 대하여 (가)-(마)의 과정을 반복하며, 모든 선분에 대하여 (라)에서의 조건을 만족하지 않으면 두 면은 overlap하지 않으므로 두면의 그리는 순서에는 무관하다.

단계 5) 단계 1)에서 보이는 면으로 판명된 면들 중 단계 2)에서 선택한 두 면을 제외한 다른 두면을 선택하여 단계 2)~4)를 수행한다.

단계 6) 8개의 모든 관측 방향에 대하여 단계 1)-5)를 수행한다.

이 알고리듬을 이용하여 각 관측 방향에 따른 다각형의 그리는 순서를 정하는 예로써 그림10과 같은 오목형 물체에 대하여 적용하였다.

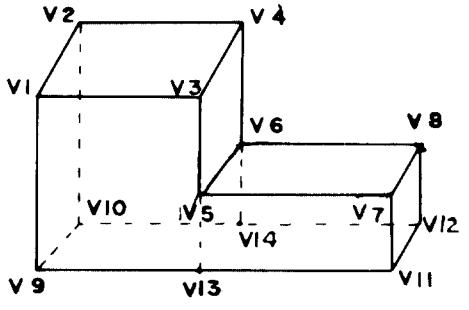


그림10. 예제의 물체
Fig. 10. Example's object.

물체를 표현하는 데이타 구조로써 다각형을 꼭지점 리스트에 대한 포인터로 표현하는 방법을 적용하여 다음과 같이 구성하였다.

Objects = {Object 1, Object 2, ···}
= {F-15, 500MD, ···}
Object = {Polygons}
Polygon = {Color, Vertices}
Vertex = {Coordinates}
= {(x1, y1, z1), (x2, y2, z2), ···}

표 2와 3은 각 각 꼭지점, 다각형의 정보를 나타낸 것이며 표 4는 위의 알고리듬 수행 결과를 나타낸 것이다.

표 2. 꼭지점 테이블
Table 2. Vertex table.

꼭지점	x	y	z
V1	−100	40	50
V2	−100	40	−50
V3	00	40	50
V4	00	40	−50
V5	0	0	50
V6	0	0	−50
V7	100	0	50
V8	100	0	−50
V9	−100	−40	50
V10	−100	−40	−50
V11	100	−40	50
V12	100	−40	−50
V13	0	−40	50
V14	0	−40	−50

표 3. 다각형 테이블
Table 3. Polygon table.

다각형	꼭지점
P1	V1 V3 V4 V2
P2	V3 V5 V6 V4
P3	V5 V7 V8 V6
P4	V7 V11 V12 V8
P5	V1 V2 V10 V9
P6	V9 V10 V12 V11
P7	V1 V9 V13 V3
P8	V5 V13 V11 V7
P9	V2 V4 V14 V10
P10	V6 V8 V12 V14

표 4. 알리고듬 적용 결과
Table 4. Result of algorithm execution.

관측 방향	다각형 그리는 순서
+++	P1, P7, P2, P8, P3, P4
++−	P1, P9, P2, P10, P3, P4
+−+	P2, P7, P8, P6, P4
+−−	P2, P9, P10, P6, P4
−++	P8, P3, P7, P1, P5
−+−	P10, P3, P9, P1, P5
−−+	P8, P7, P6, P5
−−−	P10, P9, P6, P5

2. 관측 방향의 계산

앞의 절에서는 관측 방향을 8개로 나누어 가 방향에서 물체를 구성하는 다각형들의 그리는 순서를 정하였다. 3차원 물체의 애니메이션을 위해서는 물체의 움직임을 초당 24회 이상 그려야 하므로 그럴 때마다 임의의 관측점에 대하여 앞 절의 단계 1)-5)를 수행해야하나 본 논문에서는 앞 절의 단계 1)-6)에서 모든 관측점을 8방향으로 나누어 각 방향에 대하여 면들을 그리는 순서를 미리 정하여 놓았으므로 물체가 이동하였을 때 관측점의 방향만을 계산하여 해당 관측 방향에서의 면들의 그리는 순서에 따라 그리면 된다. 물체의 애니메이션 전시 방법을 2가지로 생각할 수 있다. 첫째는 물체를 원점에 놓아 두고 관측점의 위치를 바꾸어 가며 그때의 물체의 모습을 그리는 방법과 둘째는 관측점의 위치를 원점에 놓아 두고 물체의 자세 및 위치를 변화시켰을 때 원점에서 물체를 바라본 모습을 그리는 방법이 있다. 두 가지 경우에 대한 관측 방향 계산은 다음과 같이 하며, 본 논문에서는 두 번째 방법을 이용하였다.

가. 관측점의 이동

그림 11과 같이 애니메이션하고자 하는 물체를 원점에 놓아두고 관측점의 위치 P를 움직이는 경우이다. 이 때는 관측점의 3D 좌표 x, y, z 값의 부호 만을 check하여 관측점의 방향을 정하면 된다.

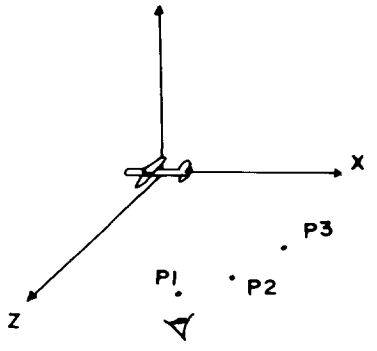


그림11. 애니메이션을 위한 관측점의 이동
Fig. 11 Viewpoint movement for animation.

나. 물체의 이동

그림 12와 같이 관측점의 위치를 원점에 놓아 두고 물체를 회전 또는 선형 이동시키는 경우이다. 이 때는 물체의 이동에 따른 상대적인 관측방향을 계산해야 한다.

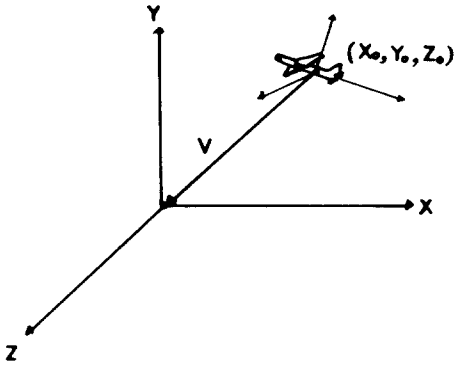


그림12. 애니메이션을 위한 물체의 이동
Fig. 12. Object movement for animation.

물체가(Xo, Yo, Zo) 위치로 translate, x축에 대하여 θ도, y축에 대하여 Φ도, z축에 대하여 γ도 만큼 회전하였을때의 관측 방향은 항공기의 원점에서 관측점(원점)을 향하는 벡터 V를 구하여 그 벡터의 성분 부호를 check하면 된다.

x축에 대하여 θ도, y축에 대하여 Φ도, z축에 대하여 γ도 만큼 회전하였을 때 변환 행렬 $R(\theta)x$, $R(\theta)y$, $R(\gamma)z$는 각각 다음과 같다.

$$R(\theta)x=\begin{bmatrix}1&0&0&0\\0&\cos\theta&\sin\theta&0\\0&-\sin\theta&\cos\theta&0\\0&0&0&1\end{bmatrix} \quad (30)$$

$$R(\Phi)y=\begin{bmatrix}\cos\Phi&0&-\sin\Phi&0\\0&1&0&0\\\sin\Phi&0&\cos\Phi&0\\0&0&0&1\end{bmatrix} \quad (31)$$

$$R(\gamma)z=\begin{bmatrix}\cos\gamma&\sin\gamma&0&0\\-\sin\gamma&\cos\gamma&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} \quad (32)$$

물체가 t초시(Xo, Yo, Zo)위치로 translate 하고, x, y, z축으로 각각 θ, Φ, γ도 만큼 회전하였을 때 관측 방향이 되는 벡터 V(x, y, z)는 식 (33)으로 부터 구할 수 있으며, 그 결과 벡터의 성분은 식 (34), (35), (36)과 같다.

$$\begin{bmatrix}x\\x\\z\\1\end{bmatrix}=-[R(\gamma)z]\cdot[R(\Phi)y]\cdot[R(\theta)x]\cdot\begin{bmatrix}Xo\\Yo\\Zo\\1\end{bmatrix} \quad (33)$$

$$x=-Xo\cdot\cos\gamma\cdot\cos\Phi-Yo\cdot(\sin\gamma\cdot\cos\theta+\cos\gamma\cdot\sin\Phi\cdot\sin\theta)-Zo*(\mathrm{stn}\gamma\cdot\sin\theta-\cos\gamma\cdot\sin\Phi\cdot\cos\theta) \quad (34)$$

$$y=Xo\cdot\sin\gamma\cdot\cos\Phi-Yo\cdot(\cos\gamma\cdot\cos\theta-\sin\gamma\cdot\sin\Phi\cdot\sin\theta)-Zo\cdot(\cos\gamma\cdot\sin\theta+\sin\gamma\cdot\sin\Phi\cdot\cos\theta) \quad (35)$$

$$z=-Xo\cdot\sin\Phi+Yo\cdot\cos\Phi\cdot\sin\theta-Zo\cdot\cos\Phi\cdot\cos\theta \quad (36)$$

따라서 x, y, z 값의 부호를 check하여 관측 방향을 정하면 된다.

Ⅳ. 실험 및 고찰

본 논문에서는 실시간 처리용 은폐면 제거 알고리듬을 다면체로 표현한 항공기(F-15, 500MD)에 대하여 비행 항적에 따라 애니메이션하여 적용하였다.

개발 환경은 floating point 계산, transformation, clipping 등을 12개의 geometry engine으로 수행하는 IRIS 워크스테이션이며, UNIX OS 상에서 C 언어로 실현하였다.

F-15기와 500MD기를 표5와 같이 50개 내외의 면으로 모델링하여 표1에서와 같이 기존의 알고리듬을 적용하여 비교한 결과 Back Face Removal 알고리듬은 항공기를 초당 29-30 frame 그렸으나 은폐면이 완전히 제거되어 그려지지 않았으며, Z-Buffer 알고리듬은 은폐면을 완전히 제거하여 그렸으나 항공기를 초당 4~5 frame 밖에 그리지 못하였고, 본 논문에서 제안한 알고리듬을 적용한 결과 은폐면을 완전히 제거하면서 100 frame 그리는데 3.3~3.5초가 소요되어 초당 약 28 frame 이상을 그리므로 실시간 처리에 매우 효율적임을 알 수 있다.

표6은 각 비행 항적에 대하여 기종별로 은폐면을 제거하여 비행 장면을 시뮬레이션한 결과의 소요 시간을 나타낸 것이다. 각 비행 항적은 미육군 물자 체계 분석 연구소에서 입수한 것으로 직진(straight), 교란 침투(jinking), 돌출 강하(pop up & dive), 회전(turn)등이 있다. 표6의 실험 결과 F-15기의 경우 직진 항적에 따라 60개의 면으로 구성된 배경과 항공기를 800 frame 그리는 데 소요된 시간이 27.5초였으며, 그림13은 이 장면의 화면을 촬영한 것이다.

표 5. 실험 결과 I
Table 5. Experimental Result I.

항공기 기종	F-15	500MD
꼭지점의 수	63	111
면의 수	43	64
100 frame 그리는 속도	3.3초	3.5초

표 6. 실험 결과 II
Table 6. Experimental Result II.

항 적	직 진	교란침투	돌출강화	회 전
Frame 수	800	700	730	580
F-15 그리는 속도	27.5초	24초	25초	19.5초
500MC 그리는 속도	29초	25.5초	27초	20.5초

본 논문의 알고리듬으로 항공기와 같이 복잡한 물체에 대해서도 실시간 처리가 가능하였으나, 그림14과와 같이 한 물체가 다른 물체를 통과하는 경우에는 물체의 표현 방법에 따라 은폐면이 정확히 제거될 수도 있거나 그렇지 않은 경우도 있다. 이와 같은 경우 하나의 면을 여러 개의 면으로 분할하여 정의함으로써 본 논문의 알고리듬을 적용할 수 있으리라 판단된다. 그림14의 예에서는 사각형 HIJK는 사각형 HDEK와 IJED와 삼각형 HDI와 KEJ로, 삼각형 ABC는 삼각형 DBE와 사각형 ADEC로 분할하면 된다.

또한 관측방향을 8개로 분할하여 본 알고리듬을 적

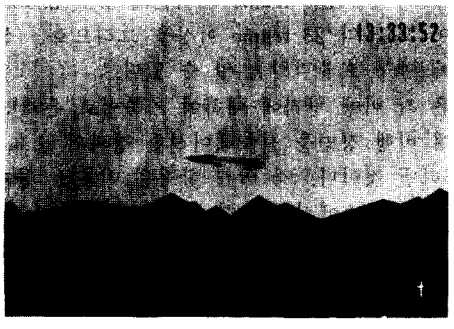


그림13. 비행 시뮬레이션 장면
Fig. 13. Flight simulation scene.

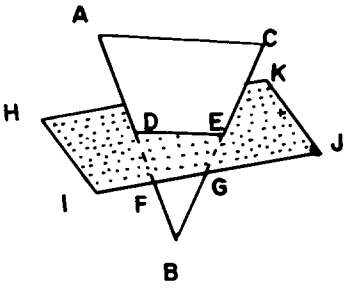


그림14. 한물체가 다른 물체를 통과하는 예
Fig. 14. Example of an object that penetrates other.

용하여 처리하였을 때 은폐면이 완전히 제거되지 않는 복잡한 형태의 물체에 대해서는 은폐면이 완전히 제거될때 까지 관측방향을 더 많이 분할하여 이들 관측방향에 대하여 본 알고리듬을 각각 적용하여 면들의 그리는 순서를 정함으로써 해결할 수 있으리라 판단된다.

V. 결 론

본 논문에서는 3차원 물체의 실시간 애니메이션을 위한 실시간 처리용 은폐면 제거 알고리듬을 제안하고 이를 IRIS 워크스테이션에서 C언어로 실현하였다.

기존의 은폐면 제거 알고리듬 중 처리 속도가 빠른 알고리듬은 볼록형 다면체와 같은 제한된 물체에 대해서만 적용 가능하거나, 모든 물체에 적용가능한 알고리듬은 처리 속도가 너무 느리므로 본 논문에서는 처리 속도가 빠르면서 항공기와 같이 오목한 형태를 갖는 물체에 대해서도 적용 가능한 알고리듬을 개발하였는데 항공기를 예로하여 비행 항적에 따라 시뮬레이션한 결과 초당 28 frame 이상 디스플레이함으로써 본 알고리듬의 유용성을 보였다.

본 알고리듬은 물체를 다각형의 집합으로 근사화한 다면체에 대하여 적용 가능하며 한 물체가 다른 물체를 통과하는 형태의 물체에 대해서도 처리할 수 있다.

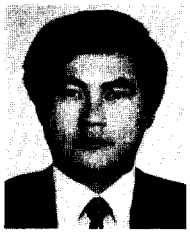
본 연구 결과는 Flight 시뮬레이션을 비롯한 실시간 시뮬레이션 및 애니메이션 분야에 적용할 수 있다.

參 考 文 獻

[1] Johnson K. Yan, "Advances in computer generated imagery for flight simulation,"

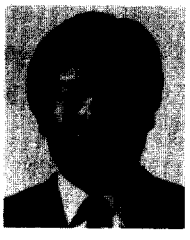
IEEE Computer Graphics & Applications, pp. 37-51, Aug. 1985.
[2] Michael J. Zyda and Robert B. McGhee, "Flight simulation for under $100,000," *IEEE Computer Graphics & Applications,* pp. 19-27, Jan. 1988.
[3] Donald hern and M. Pauline Baker, *Computer Graphics,* Prentice-Hall, 1986.
[4] Griffiths, J.G., "Bibliography of hidden line and hidden surface algorithms," *Computer Aided Design,* vol. 10, no. 3, May 1978.
[5] Sutherland, I.E., Sproull, R.F., and Schumacher, R.A., "A characterization of ten hidden-surface algorithms," *Computing Surveys,* vol. 6, no. 1, pp. 1-55, Mar. 1974.
[6] David F. Rogers, *Procedural Elements for Computer Graphics,* McGraw-Hill Book Company, 1985.
[7] Chan S. Park, *Interactive Microcomputer Graphics,* Addison-Wesley Publishing Company, 1985.
[8] *IRIS User's Guide Volume I Graphics Programming,* Silicon Graphics Inc., 1986.
[9] Fundamentals of Graphics Programming, Silicon Graphics Inc., 1988.

著 者 紹 介



金 應 坤(正會員)
1957年 7月 11日生. 1980年 2月 조선대학교 전자공학과 졸업. 1987年 2月 한양대학교대학원 전자공학과 공학석사 학위 취득. 1987年 3月～현재 조선대학교 대학원 전기공학과 박사과정 재학. 1981年 4月～1984年 7月 해군장교 근무. 1984年 8月～1985年 8月 금성반도체(주)연구소 연구원. 1987年 2月～현재 국방과학연구소 연구원. 주관심분야는 컴퓨터 그래픽스, 영상처리, VLSI CAD 등임.

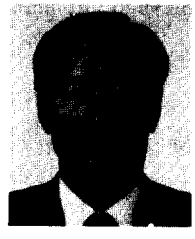
●



朴 鍾 安(正會員)
1952年 7月 7日生. 1975年 2月 조선대학교 전자공학과 졸업. 1986年 2月 조선대학교대학원 공학박사 학위 취득. 1983年 8月～1984年 8月 미국 매샤추세츠 대학 객원 교수. 현재 조선대학교 전자공학과 부교수. 주관심분야는 디지틀 신호처리, A/D-D/A Converter, 컴퓨터 그래픽스 등임.

宋 鐵(正會員)
1938年 2月 28日生. 1961年 3月 조선대학교 전기공학과 졸업. 1977年 2月 조선대학교대학원 전기공학과 공학석사 학위 취득. 1985年 2月 전북대학교 대학원 전기공학과 공학박사 학위 취득. 1973年 3月～현재 조선대학교 전산기공학과 교수. 주관심분야는 신호처리, 컴퓨터 그래픽스 등임.

●



金 俊 炫(正會員)
1933年 3月 1日生. 1956年 3月조선대학교 전기공학과 졸업. 1959年 3月 조선대학교 대학원 공학석사 학위 취득. 1986年 2月숭실대학교 대학원 공학박사 학위 취득. 1989年 현재 조선대학교 전자공학과 교수. 대한전자 공학회 전남, 광주지부 지부장, 평의원. 주관심분야는 제어계측, 컴퓨터 그래픽스 등임.