大韓機械學會論文集 第12卷 第1號, pp. 173~182, 1988.

<論 文>

個人用 컴퓨터를 利用한 相貫線의 圖示

蔡 熙 昌*

(1987年 11月 2日 接受)

Drawing of Penetrating Lines Using Personal Computer

Hee-Chang Chae

Key Words: Penetrating Line(상관선), 3-D Computer Graphics(3차원 컴퓨터 그래픽), Data Structure(데이터 구조), PROLOG(프로로그), Relational Database(관계 데이터베이스)

Abstract

A program for drawing of penetrating lines was developed in personal computer. PROLOG, a language of Artificial Intelligence, was used and a data structure using relational data base was designed. An algorithm for finding the penetrating lines in the real space was developed. The program can be applied at any types of penetrating problems like curve-surface, surface-surface, curve-object, surface-object, object-object, etc. In developing the program, the following results were obtained.

(1) Relational data base built in PROLOG and the function of backtracking are helpful in Computer Graphics.

(2) In spite of increasing the number of edges, assigning direction to the edges makes it possible to represent the polygon meshes as the non ordered sets of directional half edges.

(3) Topologicaly the penetrating lines of a polygon can be represented as the edge-pairs in the edge list of the polygon,

1. 序 論

두 物體가 交叉하였을 때 생기는 相貫線(line of intersection, penetrating line)을 구하는 문제는 板金 등에 많이 利用될 수 있어 많은 관심을 끌어 왔다. 과거에는 주로 圖學的인 방법을 사용하였으나 재현성이 없고, 形狀이 복잡할 경우 相貫線을 구하는 것이 대단히 어렵다. 요즘에 와서 컴퓨터 기술이 발전함에 따라 과거에 手 作業에 의존하던 일들이 컴퓨터 그래픽에 의하여 그려지고 있으나[1,2], 컴퓨터 그래픽에서도 隱線處理에 어려움이 있기 때문에 隱線處理前에 相貫線을 따라 面을 分割할 필요가 있다[3].

本 硏究에서는 相貫線을 實空間(real space)에서 구하여 이를 임의의 視點에 대하여 그리도록 컴퓨터 프로그램을 개발하였다. 사용 言語는 人功知能語의 一種인 PROLOG로서 言語 自體에 關係데이타베이스를 內藏하고 있고, 백트랙킹(backtracking)기능, 강력한 리스트 데이터구조(list data structure)가 큰 특징이다[4].

* 正會員, 全北大學校 工科大學 機械工學科

사용컴퓨터는 대우 CPC 2600-XT 이며 기억용량은 512 KB 이다.

2. 데이터構造

PROLOG 에서는 데이터베이스를 통해서만이 데이터의 저장과 변경이 가능하다. 모서리, 꼭지점, 多角形 등을 複合領域(compound domain)을 가진 데이터베이스를 설계함으로써 간략하게 표시할 수 있으나, 리스트(list) 演算[4](append, select, delete, member 등)이 모든 데이터베이스 領域에 대해 정의되어야 하고, 修正과 削除가 복잡해지는 단점이 있다. 本 硏究에서는 平面적인 데이타베이스를 사용하되, 해당 세그멘트(segment)와 꼭지점, 모서리, 多角形 번호 등을 포함시켰다. 이러한 構造는 리스트演算을 整數의 領域에 한정시키고 각각의 데이타베이스를 독립적으로 다룰 수 있는 장점이 있다.

多角形은 순서를 고려한 꼭지점의 리스트(ordered list of vertices)로 표시될 수 있으나 속이 빈 多角形 등 복잡한 形狀을 가진 立體를 다루기 어렵고, 반모서리(half edge) 루우프의 리스트[5]로 표시할 수 있으나 루우프내의 순서를 고려하여 자료의 변경을 행하여야 한다.

Baumgart[6]는 모서리에는 인접하는 多角形에 관한 정보를, 꼭지점에는 부수되는 모서리에 관한 정보등을 포함시켰으나, 本 硏究에서는 꼭지점에는 X,Y,Z 座標만 저장하고, 모서리에 方向을 부여한 후 多角形을 모서리(線分) 번호의 리스트로 표시하였다. 링(ring) 또는 루우프 형식으로 저장치 않기 때문에 리스트내에서의 순서는 고려치 않고 단순한 集合으로 볼 수 있다. 모든 자료가 데이터베이스에 저장되므로 多角形의 연결성 등 기타 필요한 정보는 별도의 데이터베이스에 저장하지 않고 관련되는 데이터베이스를 檢索하여 얻을 수 있다.

2.1 꼭지점(Vertex)의 표시

$v(S,V,X,Y,Z)$ S : 세그멘트 번호—整數
V : 꼭지점 번호 —整數
X : X 座標 —實數
Y : Y 座標 —實數
Z : Z 座標 —實數

2.2 모서리(Edge)와 多角形(Polygon)의 표시

모든 모서리는 시작점과 끝점으로 표시되며 方向을 가지고 있다. 立體의 경우 多角形과 多角形이 접촉하는 공통 모서리가 존재하므로 시작점과 끝점이 서로 바뀐 2개의 모서리(모서리雙)가 존재한다(Fig. 1). 모서리에 인접한 多角形에 대한 정보를 포함시키지 않기 때문에 단순히 有向線分형태로 저장되므로, 多角形의 list에 有向線分(모서리) 번호를 기록하여야 한다. 有向線分과 모서리雙(線分雙)의 사용에 대한 장단점은 아래와 같다.

(1) 多角形을 모서리의 리스트로 표시할 때 리스트내의 순서를 고려할 필요가 없다.

(2) 他 多角形과의 연결관계를 고려치 않고, 독립적으로 多角形내의 모서리를 切斷, 削除할 수 있다.

(3) 오목형, 속이 빈 多角形 등 일반적인 多角形의 표현이 가능하다.

(4) 內部, 外部 등 위치 판별이 용이하다.

(5) 三角形으로의 分割이 용이하다.

(6) 반면 모서리의 數가 증가하나, 인접한 多角形에 대한 정보를 모서리에 포함시킬 필요가 없다.

(7) 畵面에 圖示할 경우 이중으로 그리는 폐단이 있으므로, 사전에 모서리雙중 1개를 削除하여야 한다.

$e(S,En,V_1,V_2,E_t)$ S : 세그멘트 번호—整數
E : 모서리 번호 —整數
V_1 : 시작점 번호 —整數
V_2 : 끝점 번호 —整數
E_t : 모서리(線分)형태—整數
0 : 補助모서리(補助線分)
(三角形 分割시의 補助모서리, 또는 隱線)
1 : 모서리(實線分)

일반적으로 多角形은 多角形의 法線 벡터와 둘러싼 모서리의 리스트로 표시하되, 모서리의 方向 벡터와 法線 벡터의 外積이 표시하고자 하는 多角形의 外部를

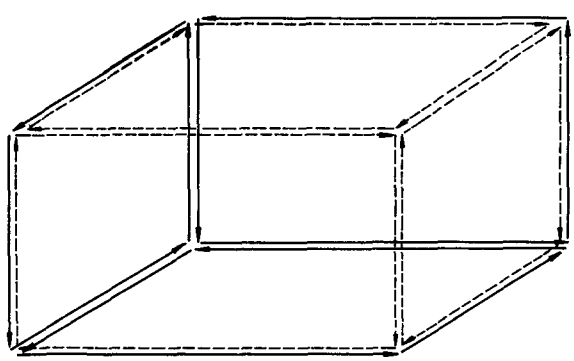


Fig. 1 Edge-pairs in the 3-D object

향하도록 한다. 이 方向은 Fig. 2과 같이 多角形의 法線 벡터의 方向에 따라서 바뀔 수도 있으므로, 多角形의 法線 벡터를 物體 外部를 향하도록 한다.

최초 자료 입력시는 物體 外部에서 보아 반시계 方向의 꼭지점들의 리스트로 표시하여, 인접한 꼭지점들이 모서리가 되도록 하는 것이 편리하다. 中空 圓柱와 같이 중앙이 비어있는 物體의 경우 꼭지점 번호를 陰數로 하여 인접한 꼭지점들이 모서리의 시작점과 끝점이 되지 않음을 표시한다. 또한 法線 벡터는 별도 입력하지 않고 꼭지점 리스트의 앞에서부터 3개의 꼭지점들로 이루어진 2개의 모서리 方向벡터로 부터 外積를 취함으로써 얻는다(Fig. 3).

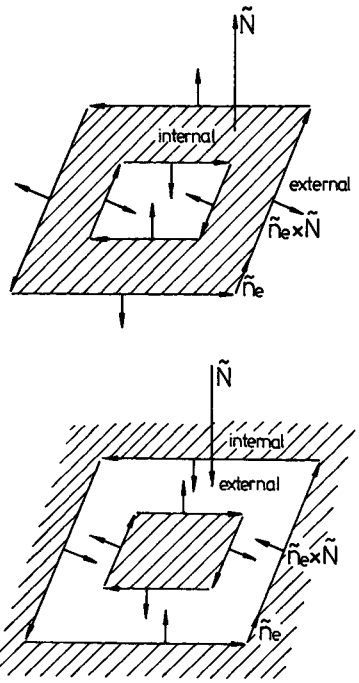
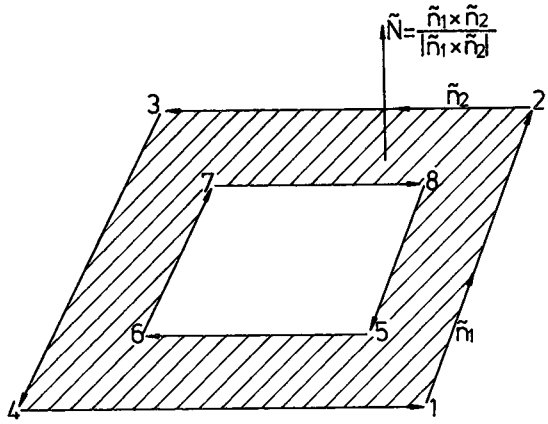


Fig. 2 Representation of polygon by the surface normal and directional edge



P(S, P, [1, 2, 3, 4, 1, -5, 6, 7, 8, 5])

Fig. 3 Representation of hollow polygon by vertex list

모서리雙은 多角形과 多角形의 공통 모서리일 경우에 나타나고 多角形 內에서는 그 多角形이 他 多角形과 交叉하거나 切斷될 경우에 이를 모서리雙으로 나타낼 수 있다.

$p(S,P,[E])$ 또는 $p(S,P,[V])$

S : 세그멘트 번호—整數
P : 多角形 번호 —整數
E : 모서리 번호 —整數
$[E]$: 모서리 번호들의 리스트(실지 演算시 사용, 순서 관련없음)
$[V]$: 꼭지점 번호들의 리스트(최초 자료입력시만 사용)

plane(S,P,N_X,N_Y,N_Z,D)

$[N_X,N_Y,N_Z,D]$: 平面의 方程式$(N_X\cdot X+N_Y\cdot Y+N_Z\cdot Z+D=0)$

(N_X,N_Y,N_Z) : 法線 벡터

2.3 立體와 曲面의 표시

立體와 曲面은 多角形의 리스트로서 표시할 수 있다. 多角形의 데이터베이스가 해당 세그멘트 항목을 포함하고 있고 데이터베이스의 自動檢索技能을 사용하면 해당 세그멘트에 속한 多角形을 쉽게 찾을 수 있기 때문에, 별도로 多角形의 리스트를 데이타베이스에 저장할 필요가 없다.

2.4 曲線의 표시

曲線은 적절한 갯수의 有向線分으로 나누어 표시할 수 있다. 모서리와 같은 有向線分의 형태이나, 多角形의 리스트가 존재하지 않고 별도의 세그멘트를 차지한다.

2.5 三角形

三角形은 多角形을 分割하여 만든다. 逆으로 三角形과 모서리에 관한자료를 가지고 多角形의 모서리 리스트를 구할 수 있다.

$t(S,P,T,V_1,V_2,V_3)$ S : 세그멘트 번호—整數
P : 多角形 번호 —整數
T : 三角形 번호 —整數
V_1, V_2, V_3 : 꼭지점 번호—整數

3. 반올림(Round Off) 誤差와 標準化

컴퓨터가 유한한 크기(4—8 byte)의 메모리로 數를 저장하기 때문에 實數演算을 하는데 있어서 반올림 誤差는 피할 수 없다. 따라서 實數를 서로 비교할 경우 이를 감안하여야 한다. 특히 實數를 비교한 결과가 프로그램의 흐름에 결정적인 역활을 할 경우 일정크기의 범위를 가지고 비교하여야 한다. 반올림 誤差에 영향을 주는 인자는 아래와 같다.

(1) 實數 1개를 저장시키는데 사용한 메모리

(2) 實數의 크기

實數를 대부분 假數(mantissa)와 指數(exponent)로 저장하기 때문에 반올림誤差는 實數의 크기에 비례하는 상대적인 誤差이다.

(3) 演算의 종류

더하기, 빼기의 경우가 가장 작고 그 다음이 곱하기 나누기이며 正弦(sine), 餘弦(cosine), 對數(log), 指數(exponent) 등의 특수함수의 경우가 가장 크다.

(4) 演算回數

(5) 자료변환시 발생

十進法의 數値(外部 file)을 二進法(內部 메모리)으로 변환하거나, 역으로 二進法의 자료를 十進法으로 변환할 경우

(1), (3), (4), (5)의 경우는 불가피 하나, (2)의 경우는 자료를 標準化하여 입력자료의 크기에 대한 영향을 배제하여야 한다. X,Y,Z 座標의 경우 최대절대값으로 이를 나누어 -1에서 1의 범위를 갖도록 함으로써 이후의 반올림 誤差를 같은 기준하에서 처리할 수 있다. 實數를 8 byte에 저장하고 특수함수계산을 한 경우, 實數 비교시 $10^{-9}\sim10^{-11}$의 誤差의 한계(ε)을 사용하는 것이 적합하다.

$K<F\longrightarrow K>F+\varepsilon$

$K=F\longrightarrow |K-F|\leq\varepsilon$

$K<F\longrightarrow K<F-\varepsilon$

$K\geq F\longrightarrow K\geq F-\varepsilon$

$K\leq F\longrightarrow K\leq F+\varepsilon$

4. 多角形의 三角形으로의 分割

相貫線의 결정, 斷面에 의한 分割, 隱線處理에 있어서 多角形을 직접 사용하기가 어려우므로 多角形을 幾何學적으로 가장 간단한 형태인 三角形으로 分割한다. 本 論文에서 사용한 三角形 分割 알고리즘은 오목형이거나, 속이 비어있는 형태이거나, 內部에 相貫線을 포함하거나 관계없이 일반적으로 사용할 수 있는 것으로 다음과 같다(Fig. 4).

PROLOG의 특성을 최대로 살려 어떤 과정이 실패하면 자동적으로 同級(—로 표시, 3—1, 3—2, 3—3)의 문장이거나, 同級의 문장이 없는 경우에는 상위 複數解가 발생하는 곳으로 백트랙킹 한다. *로 표시한 곳은 複數解가 존재하는 곳을 나타낸다.

*1. 데이터베이스로부터 多角形 1개를 선택하고 多角形의 모서리 리스트를 버퍼 $pp(Es)$에 저장한다.

*2. $pp(Es)$에서 1개의 모서리를 선택하여 시작점과 끝점을 각각 V_1, V_2로 한다.

*3. 시작점이 V_2이거나 끝점이 V_1이거나 또는 다른 임의의 모서리를 선택하여 그 시작점(또는 끝점)을 V_3로 정한다.

4. 三角形 $V_1-V_2-V_3$가 三角形 분리조건을 만족하는지 여부를 검정한다.

(1) 三角形 분리조건 I (orientation)

三角形 $V_1-V_2-V_3$의 法線 벡터(N_X, N_Y, N_Z)가 多角形의 法線 벡터와 일치한다(부호가 반대가 되면 V_1, V_2, V_3로 분리 불가능하다).

(2) 三角形 분리조건 II (includeness of other edges)

V_1-V_2, V_2-V_3, V_3-V_1를 제외한 Es의 어떠한 모서리도 三角形 $V_1-V_2-V_3$에 포함되는 부분이 없어야 한다.

(3) 모서리의 削除와 追加

Es에 V_2-V_3, V_3-V_1 모서리의 존재여부를 확인하여 V_1-V_2와 해당 모서리를 버퍼에서 削除한다.

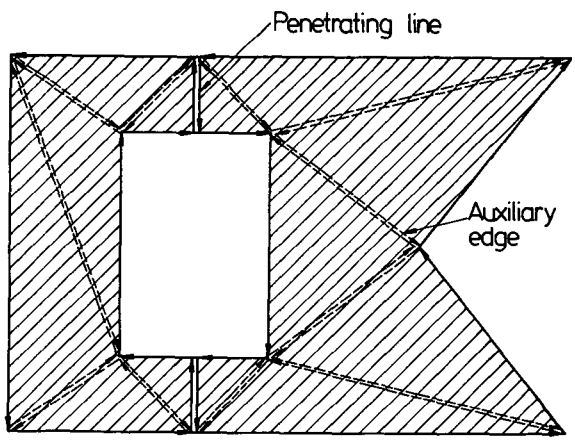


Fig. 4 Decomposition of polygon into triangles

(3—1) $V_2—V_3$, $V_3—V_1$ 모두 존재할 경우

버퍼로 부터 $V_1—V_2$, $V_2—V_3$, $V_3—V_1$의 3개의 모서리를 削除한다(4.의 과정 성공).

(3—2) $V_2—V_3$가 존재하고 $V_3—V_1$, $V_1—V_3$의 모서리가 존재하지 않는 경우

버퍼로부터 $V_1—V_2$, $V_2—V_3$의 2개의 모서리를 削除하고, $V_1—V_3$의 補助모서리를 追加한다(4.의 과정 성공).

(3—3) $V_3—V_1$가 존재하고 $V_2—V_3$, $V_3—V_2$의 모서리가 존재하지 않는 경우

버퍼로부터 $V_1—V_2$, $V_3—V_1$의 2개의 모서리를 削除하고, $V_3—V_2$의 補助모서리를 追加한다(4.의 과정 성공).

(3—4) $V_2—V_3$, $V_3—V_2$, $V_3—V_1$, $V_1—V_3$의 모서리 모두 존재하지 않는 경우

버퍼로부터 $V_1—V_2$를 削除하고 $V_3—V_2$, $V_1—V_3$의 補助모서리를 追加한다(4.의 과정 성공).

(3—5) (3—1)—(3—4)의 과정이 실패할 경우 : *3.의 과정으로 백트랙킹.

5. 三角刑 $V_1—V_2—V_3$를 데이터베이스에 追加한다.

6-1. Es≒ [ ] : 2의 과정으로 백트랙킹.

6-2. Es=[ ] : 補助모서리를 모두 削除한 후 1의 과정으로 백트랙킹.

5. 相 貫 線

5.1 曲線과 面 및 立體의 相貫

曲線 또는 直線이 面 또는 立體와 交叉할 경우 曲線에 속한 모든 有向線分과 面 또는 立體에 속한 三角形의 交點을 조사하여 交點이 존재할 경우 이점을 기준으로 分割한다.

5.2 相貫線

서로 다른 세그멘트에 있는 多角形이 접촉할 때 相貫線이 생긴다. 多角形끼리의 相貫線은 직접구하기 어려우므로 多角形을 幾何學적으로 가장 간단한 형태인 三角形으로 분해한 후 三角形끼리의 相貫線을 구한 다음(Fig. 5) 이것을 해당 多角形의 리스트에 반영한다. 多角形의 內部에 相貫線이 존재할 경우 이선을 기준으로 三角形이 分割된다고 볼 수 있으므로 모서리雙으로 표시되어야 한다. 결국 相貫線을 따라서 칼질을 한다고 볼 수 있다. 경우에 따라서는 기존의 모서리가 相貫線에 의하여 분리될 수도 있고 기존의 모서리에 포함되어 아무 효과도 없는 경우도 있을 수도 있다.

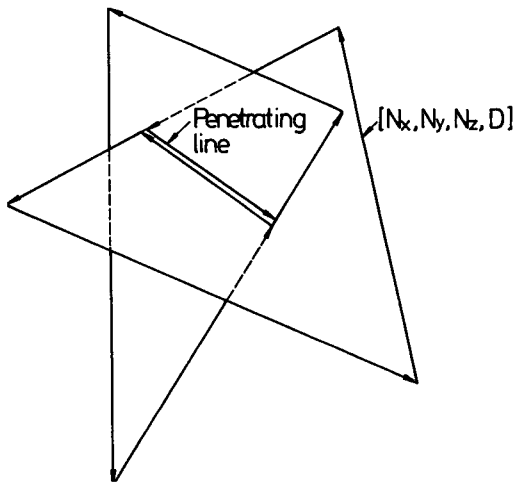


Fig. 5 Penetrating line

相貫線을 항상 모서리雙으로 표시하기 때문에 원래의 多角形의 位相的 一貫性(topological consistency)에 아무런 영향을 미치지 않고 多角形 內部에 相貫線이 있음을 기록할 수가 있다. 추후 이선을 기준으로 하여 三角形으로 分割한 후, 상대편 세그멘트 內部에 속한 三角形을 削除하고, 三角形에 속한 모서리를 削除한다면 서로 交叉하는 物體의 합침(union)등에 利用할 수 있다. 또한 相貫線을 實空間(real space)에서 구하였으므로 전개도의 작성에도 사용할 수 있다.

*1. 데이터베이스로 부터 서로 다른 두 개의 세그멘트의 각각 分割된 三角形을 선택한다.

2. 첫번째 三角形이 소속한 平面의 方程式으로 부터 나머지 한개의 三角形의 3변이 그 平面과 交叉하는 點들을 구한다.

*3. 交叉點들(최대 3點)로 부터 2點씩을 연결하는 線分이 平面의 方程式을 구한 三角刑 內部에 존재하는 부분(相貫線)을 구한다.

4. 相貫線을 첫번째 三角形이 소속한 多角形에 합치되, 당시 多角刑에 존재하는 모든 모서리와 포함關係, 交叉關係를 고려한다(Fig. 6)

(1) 相貫線을 補助線分의 형태로 저장한다.

*(2) 데이터베이스로 부터 1개의 補助線分을 선택한다(처음 시작은 1개 뿐임).

(3—1) 多角形의 어떤 모서리와 포함關係나 交叉關係가 성립할 경우.

*1) 多角形의 모서리 리스트로부터 1개의 모서리를 선택한다.

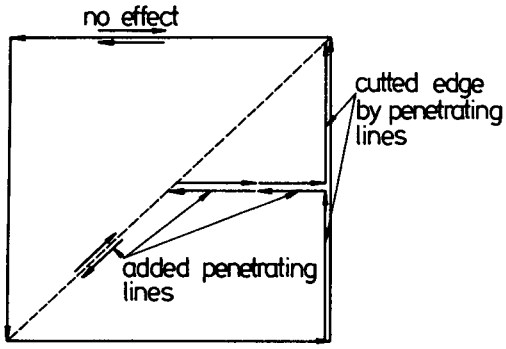


Fig. 6 Addition of penetrating line and cutting edges by penetrating line

2) 포함關係, 交叉關係 검증

2-1) 모서리 속에 포함할 경우

補助 線分을 削除하고, 포함되지 않는 부분의 補助線分을 追加한다(3)의 과정 성공.

2-2) 모서리와 交叉할 경우

交叉點을 기준으로 補助線分과 해당 모서리를 각각 切斷한다. (3)의 과정 성공.

(3—2) 어떤 모서리와도 포함거나 交叉하지 않을 경우

補助 線分을 削除하고 相貫線의 모서리雙을 多角刑에 합친다.

(4—1) 補助 線分이 존재할 경우

(2)로 백트랙킹

(4—2) 補助 線分이 존재하지 않을 경우

4의 과정 성공.

5. 4와 동일하게 두번째 多角形의 리스트에 相貫線을 합침.

6. 3 또는 1으로 백트랙킹.

5.3 一直線 分割 모서리의 합침

相貫線을 구하는데 있어 多角形과 多角刑의 相貫線을 구하지 않고, 分割된 三角形끼리의 相貫線을 구하기 때문에 相貫線도 分割되어 나타난다. 어차피 相貫線을 고려하여 다시 三角形 分割을 시작하여야 하는바 相貫線이 分割되면 그만큼 三角形數가 증가하는 폐단이 있다. 三角形數를 줄이기 위하여 중간에서 다른 모서리와 交叉하지 않으면서 一直線이 되는 모서리의 경우는 이를 합쳐야 한다.

6. 座標변환과 투시도의 작도

6.1 視座標系

直交座標系상의 임의점 (X, Y, Z)를 球面座標系 ρ, θ, ϕ로 표시된 視點에서 보았을 때 視座標는 다음과 같다[7].

$$X_e = -X\cdot\sin\theta + Y\cdot\cos\theta$$
$$Y_e = -X\cdot\cos\theta\cos\phi - Y\cdot\sin\theta\cdot\cos\phi + Z\cdot\sin\phi$$
$$Z_e = -X\cdot\sin\phi\cdot\cos\theta - Y\cdot\sin\theta\cdot\sin\phi - Z\cdot\cos\phi + \rho$$

6.2 畫面座標系

視座標系에서 畫面座標系로 座標변환은 투상의 종류에 따라 달라진다. 투상시 주의 할 것은 선 또는 平面상에 있는 2 또는 3點의 畫面座標(X_S, Y_S)와 각점의 깊이(Z_S)가 주어졌을 때 그외 동일直線 또는 동일平面상에 있는 임의의 點에 대한 깊이를 선형적으로 구할 수 있도록 하는 일이다[8]. 크립핑(clipping)은 하지 않고 畫面상에 가능한 최대 크기로 그리도록 자동적으로 스케일링(scaling)하도록 한다.

(1) 평행투상(parallel projection)

평행투상의 경우 視點과 원점과의 거리 ρ는 무의미하다. 視座標(X_e, Y_e, Z_e)를 직접 사용할 수 있으나 반올림誤差를 제어 하기 위하여 깊이의 경우 Z_e에서 ρ를 뺀값으로 정한다. 사전에 X, Y, Z를 標準化하였으므로 X_S, Y_S, Z_S는 1 근처의 값을 갖는다.

$$X_S = X_e$$
$$Y_S = Y_e$$
$$Z_S = Z_e - \rho$$

(2) 중앙투상(central projection)

Newman[8]에 의하면 畫面座標는 아래와 같으나, 사전에 X, Y, Z값을 標準化하였고 추후 스케일링하기 때문에 畫面폭 S를 1로 정한다.

$$W = S\cdot Z_e/D$$
$$X_S = X_e/W$$
$$Y_S = Y_e/W$$
$$Z_S = S\cdot(Z_e/D - 1)/(1 - D/F)/W$$

D : Z_e 중 최소값

F : Z_e 중 최대값

S : 1

(3) 畫面座標系에서의 平面의 方程式

깊이 비교시 畫面座標(X_S, Y_S)가 주어질 경우 깊이(Z_S)를 쉽게 구하기 위하여 $A\cdot X_S + B\cdot Y_S + Z_S + d = 0$, $[A, B, 1, d]$의 형태로 구하여 저장한다. 이때 畫面座標系가 左手系임을 주의하여야 한다.

(4) 깊이 계산

1) 동일線分 사이에서 임의점의 깊이

畫面座標(X_{S1}, Y_{S1}, Z_{S1}), (X_{S2}, Y_{S2}, Z_{S2}) 사이 임의점의 깊이 Z_S는

$$R=\frac{[(X_S-X_{S1})^2+(Y_S-Y_{S1})^2]^{1/2}}{[(X_{S2}-X_{S1})^2+(Y_{S2}-Y_{S1})^2]^{1/2}}$$

$$Z_S=Z_{S1}+R\cdot(Z_{S2}-Z_{S1})$$

2) 平面에서의 임의점의 깊이

畫面座標상에서의 임의점(X_S, Y_S)의 깊이 Z_S는 平面의 方程式이 $[A, B, 1, d]$인 경우 $Z_S=-A\cdot X_S-B\cdot Y_S-d$이다.

6.3 隱線의 처리

隱線의 처리에는 가려진 부분을 그리지 않는 경우와 點線으로 그리는 경우를 고려하여 隱面處理 알고리즘(hidden surface algorithm)을 사용하지 않고 隱線處理 알고리즘(hidden line algorithm)을 사용한다. 隱線을 點線으로 그리기 위해서는 가린 부분에 대한 정보를 계속 저장하여야 하므로 가린부분의 線分을 잘라 補助線分에 저장한다.

數많은 隱線處理 알고리즘[9]이 있으나 모든 경우에 적합한 알고리즘은 존재하지 않는다. 曲線이 曲面 또는 立體와 交叉할 때 交叉點을 기준으로 曲線을 분리하였고 面과 面의 相貫線을 實空間(real space)에서 구한 후 그 선을 따라 三角形으로 多角形을 再 分割하였기 때문에 단순히 三角形과 線分의 깊이 비교(Fig. 7)에 의하여 線分의 가림여부를 판정할 수 있다.

1. 畫面座標系에서 $X-Y$平面과 垂直인 多角形은 三角形과 多角形을 사전에 削除한다.
2. 뒷면(back face)의 처리
   (1) 모든 立體 세그멘트의 뒷면에 관련된 三角形을 削除하고, 多角形에 속한 모든 線分을 補助線分으로 대체한다.
   (2) 多角形 세그멘트의 뒷면은 해당 三角形의 꼭지점의 저장순서 $V_1-V_2-V_3$를 $V_3-V_2-V_1$으로 바꾼다.
3. 線分雙의 제거
   (1) 補助線分과 또 다른 線分(實線分, 補助線分)이 線分雙을 이룰 경우 補助線分을 削除한다.
   (2) 實線分과 實線分이 線分雙을 이룰 경우 實線分 1개를 削除한다.
4. 隱線의 補助線分化
   *(1) 三角形을 차례로 1개씩 선택한다.
   *(2) 實線分$(X_a, Y_a, Z_a)-(X_b, Y_b, Z_b)$를 차례로 1개씩 선택한다.
   (3) 三角形에 의한 實線分의 가림과 隱線의 補助線分化

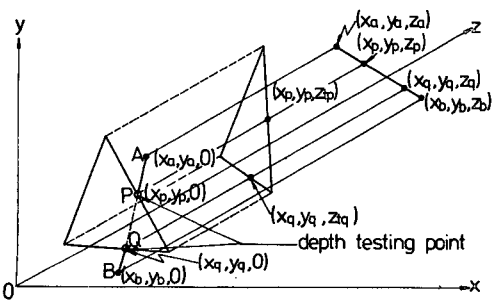


Fig. 7 Depth test between line and triangle in screen coordinates

1-1) A, B 2點 모두 內部에 있는 경우

$Z_{ta}=-A\cdot X_a-B\cdot Y_a-d$,

$Z_{tb}=-A\cdot X_b-B\cdot Y_b-d$

$|Z_{ta}-Z_a|\leq\varepsilon$, $|Z_{tb}-Z_b|\leq\varepsilon$: not hidden

$Z_a>Z_{ta}-\varepsilon$, $Z_b>Z_{tb}$: hidden

$Z_a>Z_{ta}$, $Z_b>Z_{tb}-\varepsilon$: hidden

기타 : not hidden

1-2) A는 內部, B는 外部에 있는 경우

三角型의 경계와의 交叉點을 P라 하면

$(A-P-B)$,

$Z_{ta}=-A\cdot X_a-B\cdot Y_a-d$,

$Z_{tp}=-A\cdot X_p-B\cdot Y_p-d$

$|Z_{ta}-Z_a|\leq\varepsilon$, $|Z_{tp}-Z_p|\leq\varepsilon$: not hidden

$Z_a>Z_{ta}-\varepsilon$, $Z_p>Z_{tp}$: $A-P$ hidden

$Z_a>Z_{ta}$, $Z_p>Z_{tp}-\varepsilon$: $A-P$ hidden

기타 : not hidden

1-3) B는 內部, A는 外部에 있는 경우

三角形의 경계와의 交叉點을 P라 하면

$(A-P-B)$,

$Z_{tb}=-A\cdot X_b-B\cdot Y_b-d$,

$Z_{tp}=-A\cdot X_p-B\cdot Y_p-d$

$|Z_{tb}-Z_b|\leq\varepsilon$, $|Z_{tp}-Z_p|\leq\varepsilon$: not hidden

$Z_b>Z_{tb}-\varepsilon$, $Z_p>Z_{tp}$: $P-B$ hidden

$Z_b>Z_{tb}$, $Z_p>Z_{tp}-\varepsilon$: $P-B$ hidden

기타 : not hidden

1-4) A, B 모두 外部에 있는 경우

三角形과의 交點이 2개 있는 경우

$(A-P-Q-B)$

$Z_{tp}=-A\cdot X_p-B\cdot Y_p-d$,

$Z_{tq}=-A\cdot X_q-B\cdot Y_q-d$

$|Z_{tp}-Z_p|\leq\varepsilon$, $|Z_{tq}-Z_q|\leq\varepsilon$: not hidden

$Z_p>Z_{tp}-\varepsilon$, $Z_q>Z_{tq}$: $P-Q$ hidden

$Z_p>{}_{tp}$, $Z_q>Z_{tq}-\varepsilon$: $P-Q$ hidden

기타 : not hidden

2) 隱線부분을 實線分에서 削除하여 補助線分化

한다.

3) (2)의 과정으로 백트랙킹

5. 實線分을 畫面에 實線으로 그린다.

6. 補助線分을 畫面에 點線으로 그린다.

6.4 DDA 알고리즘

實數演算을 하지 않기 때문에 마이크로 컴퓨터에 적합한 Bresenham의 알고리즘[10]을 사용하였다. 隱線을 點線으로 그을 경우도 고려하여 원래의 알고리즘을 修正하여 實線, 點線 모두 그릴 수 있도록 하되, 點線의 경우 일정 선분형식(dot pattern)을 반복하도록 하였다. 예를 들어 선분형식이 3인 경우, 3點은 그리고,

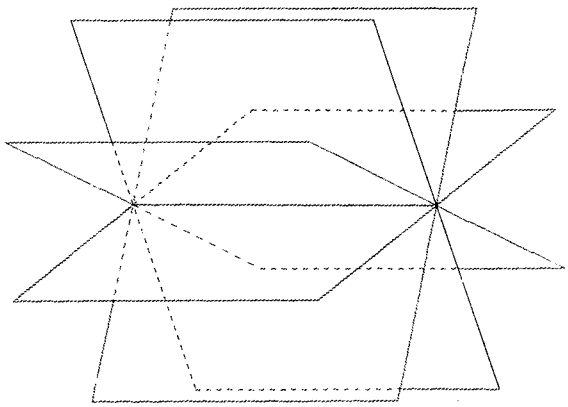


Fig. 9(b) Intersection of 4 squares(dotted hidden line)

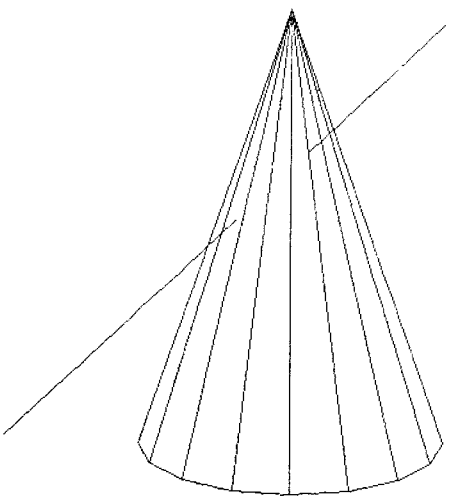


Fig. 8 Intersection between a cone and a line segment

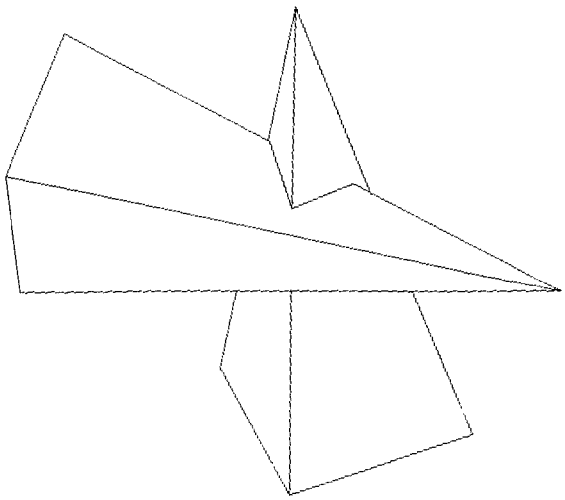


Fig. 10 Intersection of 2 pyramids

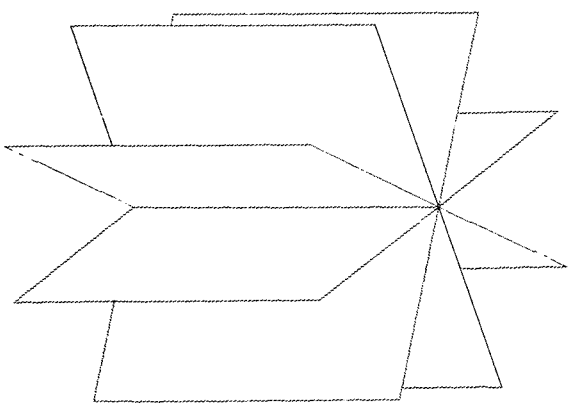


Fig. 9(a) Intersection of 4 squares(hidden line removed)

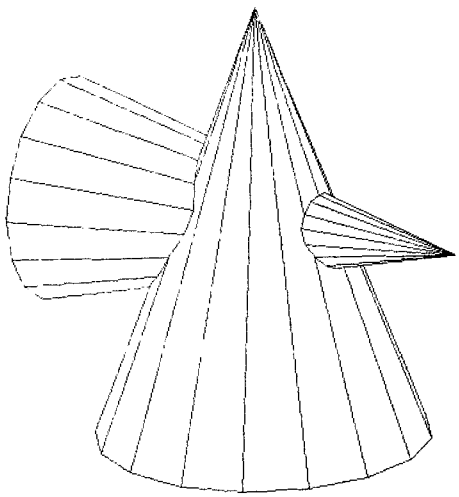


Fig. 11(a) Intersection of 2 cones

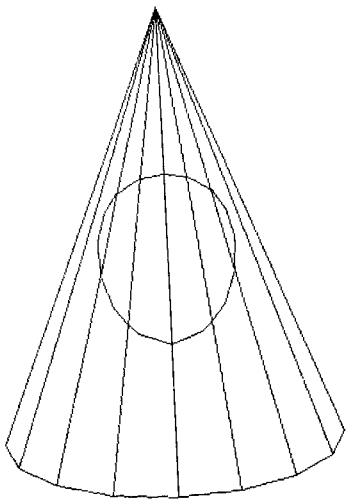


Fig. 11(b) Penetrating lines on the cone

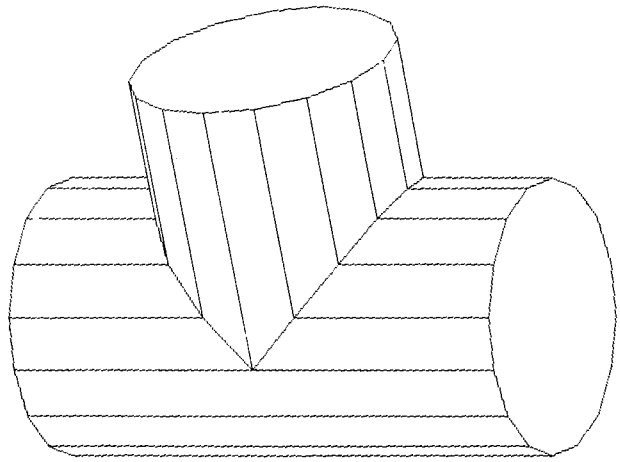


Fig. 13 Intersection of 2 circular cylinders(axis of rotations intersect)

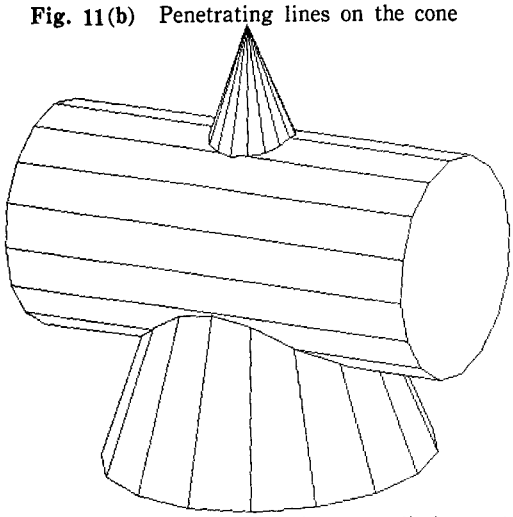


Fig. 12(a) Intersection of a circular cylinder and a cone

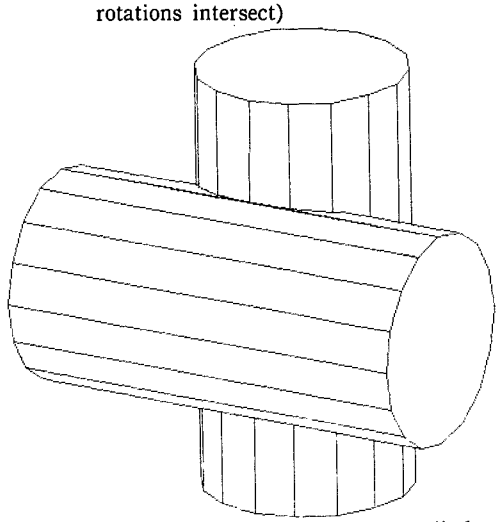


Fig. 14(a) Intersection of 2 circular cylinders

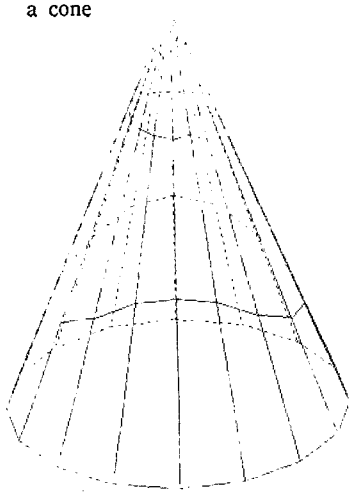


Fig. 12(b) Penetrating lines on the cone

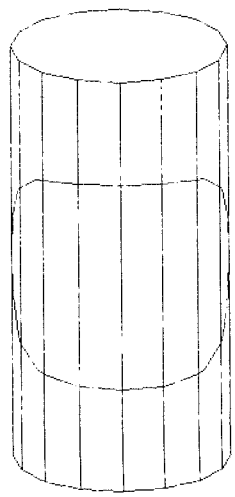


Fig. 14(b) Penetrating lines on the cylinder

3點은 그리지 않는 것이다.

PROLOG 自體에서 제공하는 그래픽 명령들이 畫面解像度가 좋지 않아(320·200) 본 논문에서 사용한 DAEWOO CPC2600-XT 에서 사용가능한 解像度(640·400)를 얻기 위하여 별도의 어셈블리 言語로 프로그램을 작성하여 컴파일 한 후, 링크시 PROLOG와 結合하였다.

7. 결 과

本 硏究에서 개발된 프로그램을 利用한 결과를 Fig. 8~Fig. 14 에 圖示하였다. 이 그림들은 640·400 dot의 畫面을 24 핀 프린터에 의하여 덤프그래픽한 것이다. 펜 프롯터에 의하여 출력한다면 보다 정확한 그림을 얻을 수 있다.

8. 결 론

1차원 曲線, 2차원 曲面 또는 3차원 立體도형이 交叉할 때, 相貫線을 實空間(real space)에서 구하는 프로그램이 개발되었다. 關係데이터베이스를 내장한 PROLOG의 컴퓨터 그래픽에의 적용은 복잡한 데이타 構造를 쉽게 다룰 수 있고, 프로그램 개발을 용이하게 한다.

有向線分의 리스트로 多角形을 표시하면, 線分數가 증가하지만 리스트내의 순서를 고려할 필요가 없이 多角形에 대한 位相的인 演算을 쉽게 할 수가 있다. 또한 相貫線은 모서리 雙으로서 位相的 一貫性을 維持시키면서 多角形의 모서리 리스트에 기록할 수 있다.

參 考 文 獻

(1) Pfeifer, H., 1985, "Methods used for Intersecting Geometrical Entities in the GPM Module for Volume Geometry", Computer-Aided Design, Vol. 17, No. 7. pp. 311~318.

(2) Lasser, D., 1986, "Intersection of Parametric Surfaces in the Bernstein-Bezier Representation", Computer-Aided Design, Vol. 18, No. 4, pp. 186~192.

(3) Giloi, W.K., 1978, "Interactive Computer Graphics", Prentice-Hall, pp.151~171.

(4) Sterling, L., 1986, "The Art of Prolog", MIT Press, pp.19~67.

(5) Mäntylä, M. 1983, "Topological Analysis of Polygon Meshes", Computer-Aided Design, Vol. 15, No. 4, pp. 228~234.

(6) Baumgart, B.G., 1975, "A Polyhedron Representation for Computer Vision", NCC, Vol. 44, pp. 589~596.

(7) Myers, R.E., 1984, "Microcomputer Graphics for the IBM PC", Addison-Wesley, pp.145~175.

(8) Newman, W.M. and Sproull, R.F., 1979, "Principles of Interactive Computer Graphics", McGraw-Hill, pp.339~363.

(9) Griffiths, J.G., 1978, "Bibliography of Hidden-Line and Hidden Surface Algorithm", Computer-Aided Design, Vol. 10 No. 3, pp. 203~206.

(10) Bresenham, J.E., 1965, "Algorithm for Computer Control of Digital Plotter", IBM Syst. J., Vol. 4 No. 1, p. 25.