한국정밀공학회지 제22권 제9호 (2005년 9월)
Journal of the Korean Society of Precision Engineering Vol. 22, No. 9, September 2005

STL 포맷의 BOOLEAN 연산에 대한 연구

손범수#, 전언찬*

A Study On the Boolean Operation Of STL Format

Beom Soo Son#, Eon Chan Jeon*

ABSTRACT

Recently, as Rapid Prototyping is attracting people's attention, many peoples are actively participating in the research on STL format. The STL files are transformed to be input for RP after modeling in CAD system. When the shape, however, needs to be changed or edited, it is impossible without the original data. So, if the original data is lost, new modeling is required. Therefore, this study suggests a way to edit the shape in STL format and to make the Boolean operation possible between the original and edited shapes. In addition, the reliability was confirmed after going over the errors of the edited data.

Key Words : RP(급속조형), STL(Stereolithography), Boolean Operation(체적연산)

기호설명

P_1, P_2, P_3 = three vertex of facet
θ = angle of rotate
N = normal vector of facet
T = tangent vector of facet
S = any point of a line at 3D
V = direction of a line at 3D
D = distance from facet to origin point
w = weight of barycentric

1. 서론

시제품 생산에 있어서 제품을 생산하기 전에 문제점을 미리 확인해 보려는 시도가 대두되면서 최근 급속조형(RP: Rapid Prototyping) 시스템에 대한 연구가 활발하게 진행되고 있다.

STL(STereoLithography) 포맷은 RP공정에 필요한 입력 데이터로 사용하고 있는데 이 포맷은 3차원 형상 정보를 가진 포맷이다. CAD SYSTEM에서 3차원 형상을 모델링한 후 그 모델링의 모든 면을 삼각형 면으로 변환하여 이 STL 파일을 생성하게 된다. 이 데이터는 또 Slicing 단계를 거치게 되고 RP 장비 속에서 광경화와 적층 공정을 거쳐 시제품이 나오게 된다.

그런데, STL 파일만으로는 최종 도형을 수정하여야 할 경우 STL 파일을 직접 수정, 편집 할 수

접수일: 2004년 12월 22일; 게재승인일: 2005년 6월 29일
교신저자: 동아대학교 기계공학과 대학원
E-mail bmsu1st@yahoo.co.kr Tel. (051) 200-7644
* 동아대학교 기계공학과

없으며, 원본 데이터를 유실하였을 경우 새로이 모델링을 하여야 한다는 문제점이 있어왔다. 그래서 STL 파일에 대한 국내외 연구가 활발히 있어왔는데 Morvan[1]등은 가상환경에 있어서 STL파일을 이용하여 구현하는 방법을 제안했으며 Tanaka[2] Son[3] 등은 facets의 오류를 구멍오류와 모서리 오류로 분류하여 구멍을 분할하고 재구성하는 것을 제안하였다. 또 M.J. Wozny[4]등은 기존의 STL 포맷의 오류를 지적하고 중복된 정점을 줄이기 위하여 STL 파일 위에 정점, 모서리, 면의 색인 리스트를 가진 새로운 RPI를 제안하였다. 그리고 A. Dolenc[5]등은 IGES를 VDAFS방식으로 변환한 다음 faceted representation 방식의 STL 포맷으로의 변환 절차를 제시하였다.

국내에서도 각각의 정점과 삼각형의 존재 조건의 관계에서 오류를 검증하여 viewer를 개발하였고,[6] 삼각형 기반 비다양체 형상을 중점적으로 다루어 오류를 수정하였다.[7] 또 구멍오류의 수정에 대한 삼각형 분할을 상태별로 다르게 적용하여 형상 및 체적의 오차를 줄인 연구도 있었다.[8-11,13] 이와 같이 STL 파일에 대한 연구가 활발히 이루어지고 있지만 아직도 일부 상용 CAD 시스템에서는 STL 파일을 직접 수정, 편집하지 못하고 있다.

따라서 본 논문에서는 STL 파일에 대한 형상의 위치, 크기, 회전 변환(Move, Scale, Rotate) 및 체적 연산(Boolean Operation)에 관한 연구를 하였으며 이로 인한 결과로 STL Editor를 개발하게 되었다.

2. STL Editor 개발

2.1 개발 환경

본 논문의 모델들은 CATIA 와 AutoCAD에서 모델링하여 STL 파일로 변환해 준 것이다. 이 변환된 STL 파일을 구현하고 편집할 수 있도록 응용프로그램을 개발하였는데 Windows XP 환경에서 Visual Basic 6.0을 사용하였다.

2.2 STL 포맷의 구조

STL 파일의 포맷은 삼각형 면(facet)에 대한 법선벡터와 세 개의 정점(vertex) 정보로 이루어졌으며 Binary 또는 ASCII로 되어 있다. Binary의 경우는 이진 데이터로 되어 있어 직접 확인할 수 없지만 ASCII의 포맷인 경우에는 Fig. 1에서와 같이 눈으로 치수를 확인할 수 있다. 세 개의 정점과 하나의 법선벡터가 모여 하나의 면으로 구성되며 도형의 외형을 이루는 표면을 따라 여러 개의 면이 모여 하나의 도형이 이루어진다는 것을 알 수 있다.

```
solid AutoCAD
  facet normal 0.0000000e+000 0.0000000e+000
               1.0000000e+000
    outer loop
     vertex 1.1859983e+001 4.3492240e+000
            3.8241639e+000
     vertex 1.1859983e+001 8.6209809e+000
            3.8241639e+000
     vertex 5.6365373e+000 8.6209809e+000
            3.8241639e+000
    endloop
    endfacet
      .....
  endsolid AutoCAD
```

Fig. 1 STL ASCII format

STL파일이 정상적인 데이터가 되기 위한 조건은 다음과 같다.

첫째, 한 개의 facet은 법선 벡터와 세 개의 정점이 존재해야 한다. 둘째, 세 개의 정점은 일직선이 되거나 두 정점 이상의 위치가 중복되지 말아야 한다. 셋째, 삼각형의 세 모서리는 각 모서리 당 하나씩 이웃하는 facet이 한 개씩만 존재해야 한다. 넷째, 이웃하는 facet의 edge는 현재 facet의 edge와 정점의 방향이 반대이며 위치는 같아야 한다. 다섯째, 법선벡터와 세 정점의 방향은 오른손 법칙에 따라야 한다. 여섯째, 하나의 도형이 이루어지기 위해서는 모든 facet이 모여 닫혀진 폐도형이 되어야 한다. 즉, 모든 삼각형 면이 서로 중첩되지 않고 형상을 덮고 있어야 하며 구멍이 나 있으면 안된다.

3. 도형의 수정과 편집

3.1 가시선의 결정

STL 포맷으로 이루어진 도형은 Fig. 1에서 보는 것과 같이 3차원 공간상의 점들로 이루어진 삼각형 면의 정보와 그 면의 법선 벡터로 이루어져 있다.

뷰어에 나타나는 이미지에 있어서 도형의 가시

선 뒷면에 존재하는 삼각형 면들을 나타내지 않도록 하려면 면의 법선 벡터와 가시선의 벡터 값을 대입한 코시-슈바르츠 부등식[12] 을 이용하여 Fig. 2에서처럼 벡터의 내적을 이용한다.

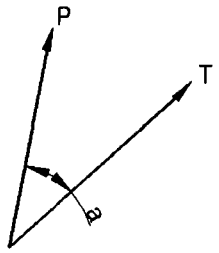

Fig. 2 inner product

뷰의 가시선의 벡터를 P라하고 삼각형 면의 법선 벡터를 T라 할 때 두 벡터 P와 T의 내적을 P · T로 표기한다. 두 벡터가 서로 직각일 때 P · T의 값은 0이다. 두 벡터의 내적의 값이 0보다 크다면 법선 벡터 T는 가시 영역 안에 있는 것이며 0보다 작을 때는 가시영역 밖에 있는 것이다.

3.2 위치, 크기 및 회전 변환

공간상의 정점에 대한 값을 변환할 경우에는 다음과 같은 행렬식을 사용한다. 위치, 크기 및 회전 변환에 사용되는 행렬식으로서 많이 알려진 식이다.

다음 식 (1), 식 (2), 식 (3)은 위치, 크기 및 회전 변환에 사용되는 행렬식이다.

$$P' = \begin{bmatrix} l \\ m \\ n \end{bmatrix} + \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} \tag{1}$$

$$P' = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} \tag{2}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

식 (3)에서 R_z 은 z축을 기준으로 회전할 경우를 나타냈으며 θ 는 회전각이다. x축과 y축에 대한 회전 변환 행렬식도 각각 yz축과 xz축으로 유도될 수 있다. 임의의 축(A)에 대한 회전은 아래 그림과 같이 주어져 식 (4)와 같이 유도된다.

$$R_A(\theta) = \tag{4}$$
$$\begin{bmatrix} c+(1-c)A_x^2 & (1-c)A_xA_y - sA_z & (1-c)A_xA_z + sA_y \\ (1-c)A_xA_y + sA_z & c+(1-c)A_y^2 & (1-c)A_yA_z - sA_x \\ (1-c)A_xA_z - sA_y & (1-c)A_yA_z + sA_x & c+(1-c)A_x^2 \end{bmatrix}$$

A는 임의의 축의 벡터 성분이며, c는 $\cos\theta$, s는 $\sin\theta$ 를 의미한다. 또한 θ 는 회전하고자 하는 각도이다. Fig. 3은 식 (4)를 유도하기 위한 임의의 회전축과 임의의 좌표에 대한 그림이다.

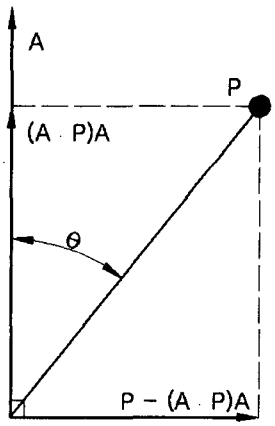

Fig. 3 Rotate of Point from any Axis

3.3 직선과 삼각형의 교차

서로 다른 두 도형에 대한 교차를 판단하려면 기본 단위인 삼각형 면끼리의 교차 여부를 판단하는 것에서부터 시작된다.

공간상에 세 꼭지점 P_1, P_2, P_3를 지나는 삼각형 면에 그 내부를 다른 삼각형의 한 변인 직선이 통과하는지를 판단하여야 한다.

Fig. 4에서 보는 것과 같이 평면의 법선 벡터를 N이라 하고 평면위의 임의의 한점을 P라 하며 원점과의 부호있는 거리를 D라 할 때 평면식은 N·P(t)

+ D = 0이다. 공간상의 직선의 식을 P(t) = S + tV 라 하고 이를 평면식에 대입하여 t에 관하여 풀면 식 (5)와 같다. 여기서 S는 직선상의 임의의 한점이며 V는 직선의 방향 벡터이다.

$$t = \frac{-(N \cdot S + D)}{N \cdot V} \tag{5}$$

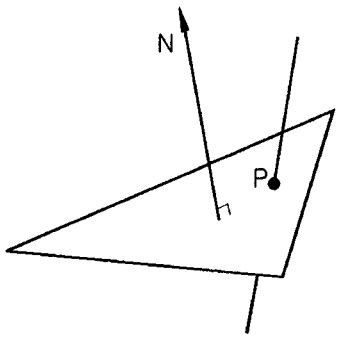

Fig. 4 An Intersection Point with the Facet

이것을 직선 P(t)에 대입하면 교점이 나온다. 이 교점이 세 점 내부에 존재하는지를 판별하여야 하는데 이것은 세 점 P의 무게 중심 좌표(barycentric coordinates)를 이용한다. 스칼라 w_0 , w_1 , w_2 는 각 꼭지점의 가중치로서, 식 (6)에서처럼 세 스칼라 값의 합은 1이다.

$$w_0 + w_1 + w_2 = 1 \tag{6}$$

여기서 각 가중치의 치수가 양수이면 세 점의 내부에 교점이 존재하게 된다. 이 가중치를 w_1 과 w_2 에 대하여 풀면 식 (7)과 같다.

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \frac{1}{Q_1^2 Q_2^2 - (Q_1 \cdot Q_2)^2} \begin{bmatrix} Q_2^2 & -Q_1 \cdot Q_2 \\ -Q_1 \cdot Q_2 & Q_1^2 \end{bmatrix} \begin{bmatrix} R \cdot Q_1 \\ R \cdot Q_2 \end{bmatrix} \tag{7}$$

여기서,

$R = P - P_1, Q_1 = P_2 - P_1, Q_2 = P_3 - P_1$ 가 된다. 가중치들이 모두 음이 아니라는 것은 $w_1 + w_2 \leq 1$ 임을 뜻하기도 한다.

3.4 면과 면이 교차하는 직선

삼각형의 면과 면이 만나 경계를 이루는 직선의 두 정점은 반드시 두 개가 나오며 두 번 나온다. 면과 직선이 교차하는 두 개의 정점(O1, O2)은 Fig. 5에서 보는 것처럼 두 삼각형 면이 하나씩 관통되는 교점을 가지는 경우이거나 한쪽 삼각형이 두 번 관통되는 경우이다. 만나는 두 정점이 두 번 나오는 것은 삼각형 모서리가 이웃하는 삼각형과 같이 공유하기 때문인데 이는 교점의 연결이 면과 면이 만나는 직선으로서 폐다각형이 되도록 안내하는 역할을 한다.

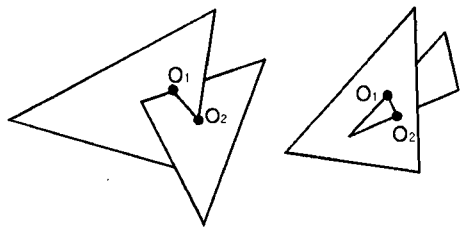

Fig. 5 An Intersection of facets

3.5 도형의 Boolean 연산

두 도형이 교차하는 체적 부분에 대한 Boolean 연산을 하고자 할 때는 교차하는 두 도형의 경계 정보와 중첩되는 정점의 정보를 파악하여 계산한다.

삼각형의 세 직선 중 교차점이 홀수 개이면 두 꼭지점 중 하나는 다른 도형의 체적 내부에 존재하게 되며, 짝수 개이면 직선의 두 꼭지점은 상대방 도형의 체적 내부를 통과하여 외부에 존재하게 된다.

Fig. 6은 직선에 교차점이 짝수개가 되어 두 끝점이 체적 외부에 있게 된 그림이다.

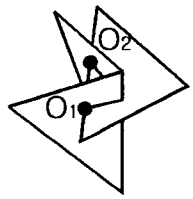

Fig. 6 The Number of the Intersection Point

상대방의 체적 내부에 존재하게 되는 꼭지점은 Boolean 연산에 따라 삭제하거나 상대방 체적에 꼭지점의 정보로 제공하게 된다.

3.6 삼각형 분할

두 도형이 교차하여 생성된 경계선에 의해 삼각형은 다각형으로 변하게 되는데 이를 STL 포맷에 맞게 삼각형으로 분할하여야 한다. 삼각형의 분할은 단순 분할인 경우는 Delaunay의 삼각형 분할법[4]을 이용한다.

Fig. 7은 Delaunay의 삼각형 분할법을 이용하여 삼각형을 분할하는 모습이다. 삼각형의 한 변에 교차점이 형성되어 다각형으로 구성되어질 때 새로이 추가된 정점을 기준으로 삼각형 분할을 시작한다.

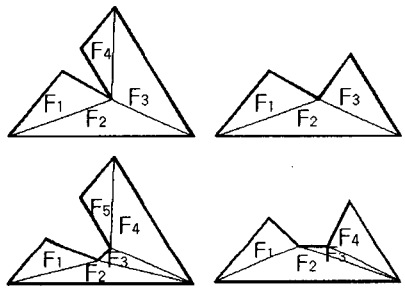

Fig. 7 The Division of Facets

하지만 복합 형태의 광범위한 삼각형 면이 훼손되었을 때는 Fig. 8과 같은 훼손된 상태에 따른 분할법 적용 방법을 이용한다.

체적의 연산으로 사라지거나 새로이 생성하여야 할 경우에 각 형태별로 적용하여 새로운 삼각형 면을 생성해 내는데 유용하다. Fig. 8의 첫 번째 그림은 평면상에 불규칙한 다각형을 삼각형으로 분할하는 그림이다. 이는 Fig. 7에서 삼각형의 세 꼭지점 중 두 개의 정점이 제거되고 적어도 두 개이상의 교점이 새로이 생성되었을 때 적용할 수 있다.

Fig. 8의 네 번째 그림은 급격한 곡면에 대한 삼각형 면을 생성하는 방법인데 현존하는 삼각형의 정점에 대한 벡터를 이용하여 다음과 같이 곡면을 생성해 낸다.

먼저 구멍오류의 경계 좌표를 구한다. 다음 경계 좌표에 인접하고 구멍오류의 중심으로 향하는 임의의 두 정점에 대한 접선 벡터와 거리 값을 구한다. 그리고 구멍오류의 경계 좌표 중에 상호 마주보는 임의의 두 정점을 선정한다. 이어서 두 정점에서의 좌표와 접선 벡터 값을 가지고 스플라인 곡선 방정식에 적용하여 곡선식을 구하고 거리 값에 따라 정점을 찍는다. 마지막으로 각각의 정점과 연결하는 삼각형 분할을 하여 각각 리스트에 추가한다.

이러한 구멍오류 수정법을 적용하였을 때는 STL 파일에 대한 데이터의 신뢰성을 향상 시킨다.

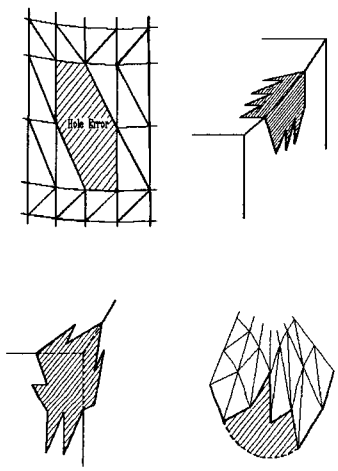

Fig. 8 Type of Hole Error

4. 결과에 대한 고찰

STL 파일을 일반 CAD System의 도형 데이터와 달리 법선 벡터와 정점의 좌표 값만을 가지게 된다. 따라서 CAD System에서 볼 수 있는 다양한 수식 연산을 적용시키기 위하여 각 정점의 좌표값에 대하여 변환 행렬식을 적용하여 도형의 크기 변환, 체적 계산, Boolean 연산이 가능하도록 하여야 한다.

우선 STL 파일을 볼 수 있도록 Viewer를 개발하였다. 이는 STL 파일을 열어서 수정 편집의 가시성을 확보하기 위하여 필수적인 항목이다. 다음 Fig. 9는 STL Viewer에서 열어본 STL 포맷인 도형의 그림이며, Fig. 10은 STL 도형의 뒷면을 제거하여 나타낸 그림이다.

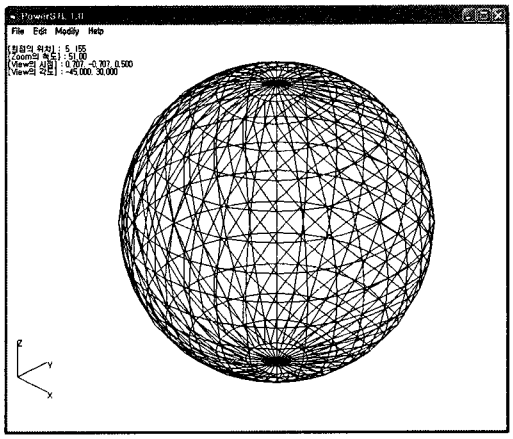

Fig. 9 STL Viewer

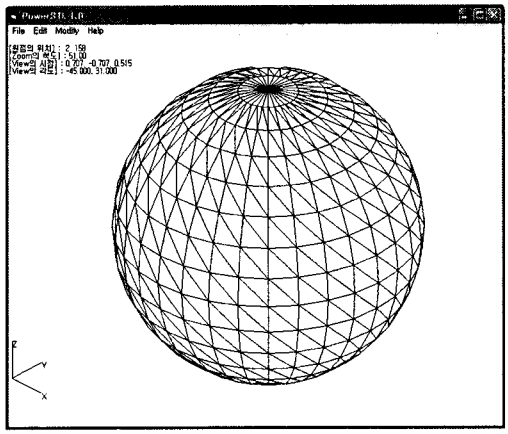

Fig. 10 Hide Line Viewer

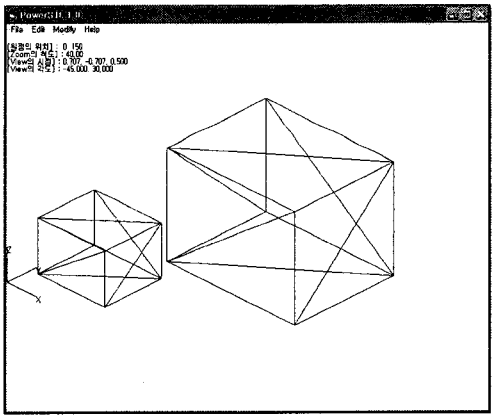

Fig. 11 Before Move Command

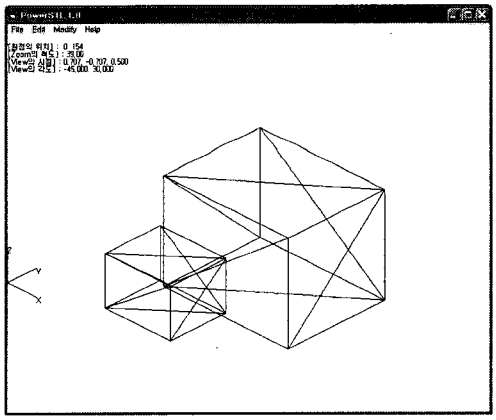

Fig. 12 After Move Command

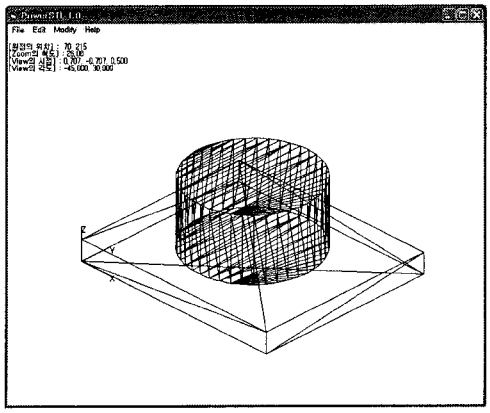

Fig. 13 Before Union Command

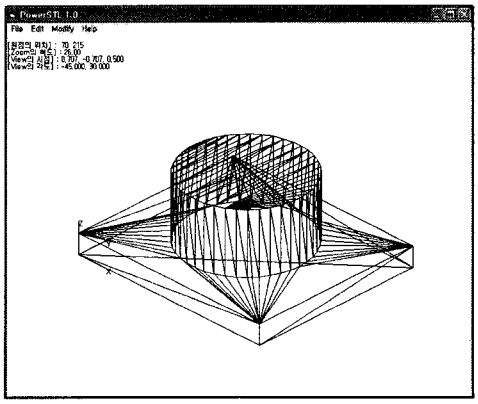

Fig. 14 After Union Command

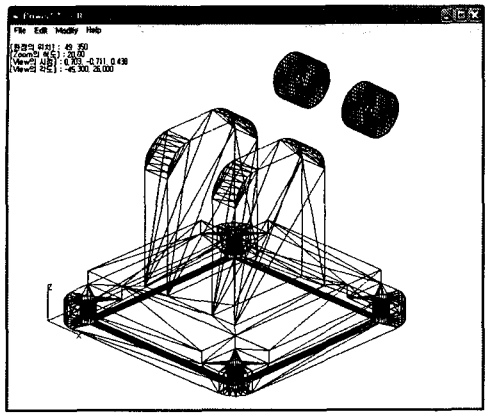

Fig. 15 Before Subtraction Command

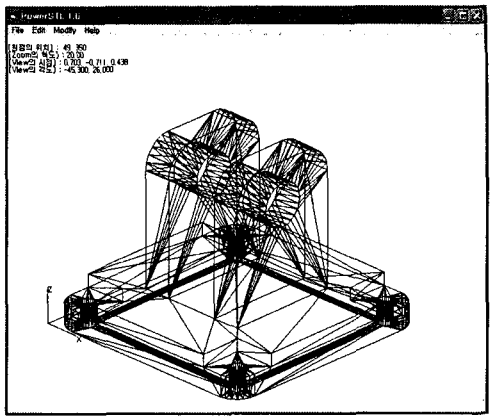

Fig. 16 After Subtraction Command

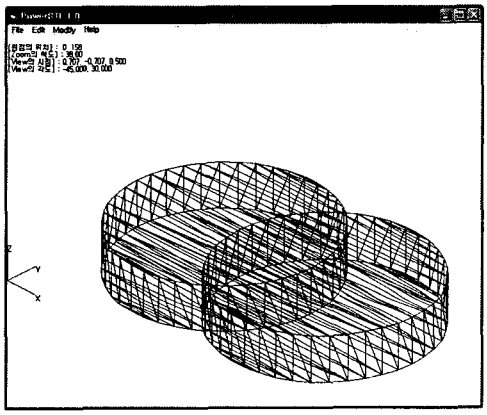

Fig. 17 Before Intersection Command

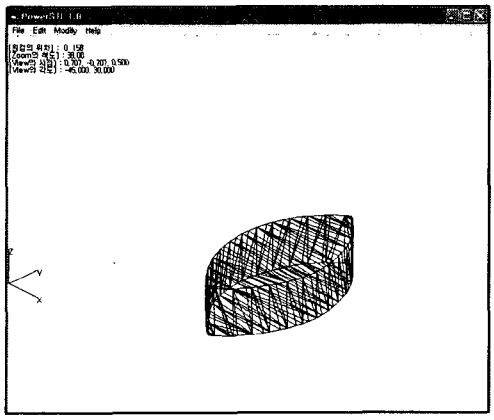

Fig. 18 After Intersection Command

둘째, 각 도형을 독립적인 객체로 분류하였고 각 도형에 대하여 기초적인 편집 명령이 적용되도록 하여 이동, 크기, 회전 변환이 가능하게 하였다. 개별적인 독립 객체로 선택되어 필요한 변환이 가능하여야 한다. 3D 도형을 편집하는 도구로서는 필수적이다. Fig. 11과 Fig. 12는 큰 도형 옆에 작은 도형을 x축 방향으로 10point 이동하기 전과 이동한 후의 모습을 나타냈다.

셋째, 두 도형의 체적에 대한 Boolean 연산이 가능하게 되었다. Union, Subtract, Intersection의 연산이 가능하게 되었는데 이는 도형의 수정 및 편집에 큰 편리성을 가져다 준다.

Fig. 13과 Fig. 14는 Union을 수행하기 전과 후의 모습을 비교하였으며, Fig. 15와 Fig. 16은 Subtraction을 수행하기 전과 후의 모습을 비교하였다. 또한 Fig. 17과 Fig. 18은 Intersection을 수행하기 전과 후의 모습을 나타낸 것이다.

여기서 Intersection과 Subtraction으로 도형에 발생하는 hole을 facet으로 새로이 생성하는 곳에는 삼각형 분할법을 적용함에 있어서 문제점이 발생하지 않았다는 것을 알 수 있다.

5. 결 론

본 연구의 결과로 STL 파일의 정보를 수정 편집 할 수 있는 Editor를 개발하게 되었다. STL Editor를 개발함으로 해서 얻을 수 있는 효과와 결론을 다음과 같이 정리하였다.

1) 원본 CAD 데이터를 분실하거나 없어도 STL 파일만으로 도형을 수정 편집할 수 있어서 CAD 데이터의 분실에 따른 정보 손실의 위험이 줄어든다.

2) 도형을 수정하기 위하여 CAD System으로 복귀하고 다시 변환해야하는 공정을 줄일 수가 있어서 STL 파일의 수정을 위한 작업 사이클이 줄어들 수 있다.

3) STL Editor에서의 수정, 편집은 조형을 하기 이전에 STL 파일에 대한 오류를 검증하는 것과 같아서 데이터에 대한 신뢰성이 향상된다.

참고문헌

1. Morvan, S. M., Fadel, G. M., "Virtual Prototyping Using STL Files," International Body Engineering Conference/IBEC, Cobo Exposition Center, October 1-3, Detroit, Mi, 1996.
2. 田中文基, 岸浪建史, "光造形法における問題点とその解決法," 第6回 光造形システムシンポジウム, pp. 39-45, 1994.
3. Son, Y. J., "A Study On the Triangulation Method for Hole Error Modification of STL Format," A Masters Thesis, Dong-A University, pp. 8-29, 1997.
4. Wozny, M. J., "Data Driven Solid Freeform Fabrication," IFIP Transactions B-3: Human Aspects in Computer Integrated Manufacturing, pp. 71-82, 1992.
5. Dolenc, A., I.Mäkelä, R. Hovtun, "Better Software for Rapid Prototyping with INSTANTCAM," IFIP Transactions B-3: Human Aspects in Computer Integrated Manufacturing, pp. 449-456, 1992.
6. Choi, H. T., Lee, S. H., "A Study on Error Verification of STL format for Rapid Prototyping System," Korean Society of Precision Engineering 1996 Spring Annual Meeting, pp. 597-601, 1996.
7. Whang, D. G., Chea, H. C., "A Verification of STL Using the Triangle Based Geometric Modeling," Korean Society of Precision Engineering 1996 Autumn Annual Meeting, pp. 578-582, 1996.
8. Kim, J. A., Paik, I. H., "CAD/CAM System Development for StereoLithography," Korean Society of Precision Engineering 1996 Spring Annual Meeting, pp. 592-596, 1996.
9. Choi, H. T., Lee, S. H., "A Study of Real-Time Geometry Modeling and VRML Application of Rapid Prototype Technology," Korean Society of Precision Engineering 1997 Spring Annual Meeting, pp. 321-326, 1997.
10. Son, Y. J., Kim, S. K., Jeon, E. C., "A Study on Developing A System for the Modification of Hole Errors in STL Formats and Its Efficiency," Proceeding of the PCMM 1998, pp. 887-892, 1998.
11. Ibrahim, Zaid, "Original CAD/CAM Theory and Practice," McGraw-Hill Book Co., Singapore, pp. 229-234, 479-516, 1995.
12. Eric, Lengyel, "Methematics for 3D Game Programming and Computer Graphics Second Edition," Charles River Media, INC., pp. 37-201, 2004.
13. Son, Y. J., Jeon, E. C., "A Study on DB Construction for Error Modification of STL Format and Efficiency by Shape Restoration," J. of the KSPE, Vol. 15, No. 12, pp.21-27, 1998.