CFD 해석을 위한 3차원 격자생성 프로그램의 개발

Development of a Three Dimensional Mesh Generation Program for CFD Simulations

*장재원[1], 김사량[2], 허남건[3]
J. Chang, S.-R. Kim & N. Hur

In the present study a mesh generation program is developed for three dimensional flow analyses with complex geometry. By the present program one can define vertices using various coordinate systems and cells for finite volume approach. Rendered display of the generated mesh can be also available in both orthographic and perspective projection modes. Through perspective projection mode, one can check the quality of generated mesh by moving around inside the mesh like a virtual reality. The examples of the program execution is given in the paper.

1. 서론

CFD 해석 프로그램은 일반적으로 격자를 생성하는 전처리장치(preprocessor), 생성된 격자를 사용하여 유동을 해석하는 해석 모듈(solver), 그리고 계산 결과를 나타내는 후처리장치(post-processor)로 구성된다. 그 중 전처리장치는 해석을 위한 격자를 생성하고 경계조건 등을 입력하는 기능을 수행하는 것으로, 사용자가 쉽게 CFD 해석을 수행하기 위해서는 이 전처리장치의 다양하고 간편한 기능 및 성능이 요구된다. 특히, 엔진룸 내부유동해석 등 최근의 복잡한 형상에서의 유동해석은 대부분의 시간을 격자생성에 소비하고 있는 실정이므로 간편한 격자생성장치의 개발이 필요하다. 또한, 생성된 격자의 형태는 계산 시 수렴속도 및 계산의 정확도 등에 큰 영향을 미치게 되므로 격자성능장치의 성능은 매우 중요하다. 상용 CFD 프로그램에서 격자를 생성하는 전처리장치로는 PROSTAR[1], GAMBIT[2], Gridgen[3], TrueGrid[4], HyperMesh[5]등이 있다. 최근 CAD 정보로부터 자동으로 격자를 생성하는 자동격자생성장치도 활발히 개발되어 SAMM[1], ICEM CFD[6] 등도 발매되고 있다.

국내에서는 아직 상용화된 CFD 프로그램이 없기 때문에 전처리 장치 또한 개발되고 있지 않으나, 몇 개의 대학에서 기본적인 격자생성기능을 수행하는 프로그램을 개발하고 있다.[7-9] 본 연구실에서는 범용 CFD 코드인 Turbo-3D solver를 개발 중에 있으며, 격자생성장치[10]와 후처리장치[11]도 개발 중에 있다.

본 연구에서는 CFD S/W package 개발의 일환으로 범용으로 사용할 수 있는 간단한 형태의 3차원 격자 생성 프로그램을 개발하였다. 격자생성방법은 먼저 격자점(Vertex)을 필요한 개수와 위치에 생성한 후 8개의 격자점이 하나의 격자(Cell)가 되도록 격자를 생성하는 방법으로 조직격자계(Structured Grid)의 가장 기본이 되는 형태이다.

2. Pre-Processor

2.1 개발환경

본 프로그램의 개발환경은 OS Windows 2000 Professional, 600MHz P-III CPU, 512MB RAM으로 개발을 수행하였으며, 사용된 Compiler는 Windows 프로그래밍에 보편적으로 사용되는 Visual C++ 6.0을 사용하였으며, 사용된 Library는 기본 툴은 MFC 6.0을 3차원 그래픽 처리를

1) 서강대학교 기계공학과 대학원 (121-742 서울시 마포구 신수동 1번지 dpme@hitel.net)
2) 강릉대학교 정밀기계공학과 (210-702 강원도 강릉시 지변동 산 1번지 dearksr@kangnung.ac.kr)
3) 서강대학교 기계공학과 (121-742 서울시 마포구 신수동 1번지 nhur@ccs.sogang.ac.kr)

반 그래픽스 기술의 필요에 의해 E&S와 Silicon Graphics에 의해 Graphics API로 개발이 시작된 OpenGL은 산업계에서 가장 광범위하게 사용되는 3D Graphics API로써 Unix상의 X-Window 및 Windows NT, Windows 98등에서 사용이 가능하다. 또한 2D Graphic과는 달리 3차원 그래픽이 표준인 OpenGL은 RealTime Rendering, Shading, Texture Mapping등 다양한 기능을 가지고 있으며, 물체의 표현 방법은 3차원좌표 값을 기준으로 하고 있다.

2.2 프로그램 구성

본 프로그램을 실행하면 Fig. 1과 같이 간단한 프로그램 로고가 나타난 후 작업할 Directory를 선택하는 대화 상자가 나타난다. 작업 Directory를 선택을 하고 나면 메인 윈도우가 나타난다. 메인 윈도우는 크게 4개로 나누어져 있는데 3차원 그래픽 처리를 위한 Display Window와 명령을 입력받는 Command Window, 사용된 명령을

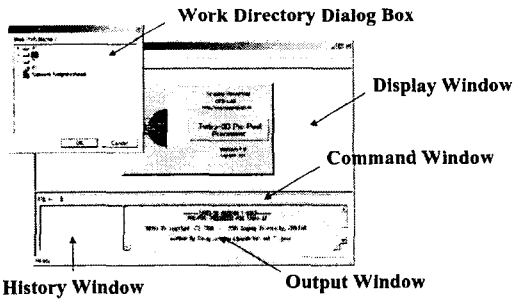
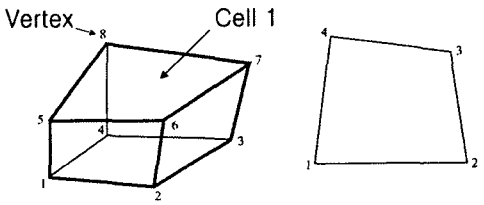


Fig. 1 Turbo 3D Pre Post Processor



(a) Cell (b) Shell

Fig. 2 Vertex, Cell and Shell

저장하는 History Window, 상태를 출력하는 Output Window로 구성되어 있다. 기본적으로는 MFC구조를 사용하고 있으며, Display Window는 CView class를 상속받아 OpenGL를 사용할 수 있도록 수정하였다.

Command Window로부터 입력받은 명령은 다음과 같은 형식으로 입력을 받는다.

Command , parameter1 , parameter2 ·····

입력받은 명령은 ' '나 ','으로 데이터를 구분하여 문자열, 정수, 실수 테이블에 각각 저장하게 된다. 이때 문자열인 경우에는 앞에서 4글자만을 인식하게 된다. 이렇게 인식된 명령은 명령어 번역기(interpreter)에 의해 실제 명령이 실행되게 된다.

2.3 격자와 격자점

본 프로그램의 격자 생성방법은 먼저 격자점을 필요한 개수와 위치에 생성한 후, Cell인 경우 8개 또는 Shell인 경우 4개의 격자점(vertex)이 하나의 격자(cell 또는 Shell)가 되도록 격자를 생성하는 방법이며 조직격자계(Structured Mesh)의 가장 기본이 되는 형태이다. 이 때 격자점은 실제 격자의 위치정보를 가지게된다. 사용 가능한 격자의 형태는 크게 cell과 shell이며, 이 2개의 형태와 격자와 격자점의 관계를 Fig. 2에 나타내었다.

본 프로그램은 몇 개의 Command 모듈로 구성되어 있다(Fig. 3). 그 중, 격자점 모듈(Vertex Module)은 좌표 시스템을 정의하는 명령어들과 격자점을 정의하는 명령어로 구성되어 있다. 격자점은 현재의 좌표 시스템에 영향을 받아 만들

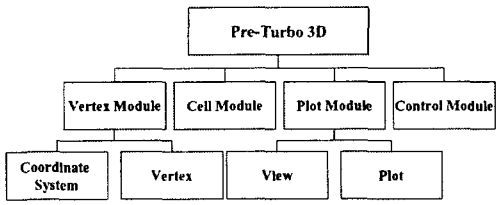


Fig 3. Command Modules of Pre-Turbo Program

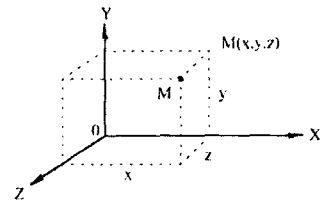
어지며, 사용 가능한 좌표 시스템은 x, y, z축을

사용하는 Cartesian 좌표계와 r, θ, z축을 사용하는 원통 좌표계, r, θ, ϕ축을 사용하는 구형 좌표계이다.(Fig. 4) 하나의 좌표 시스템은 절대 좌표계(원점이 0,0,0인 Cartesina 좌표계)로부터의 원점의 위치, 각 축의 회전각도, 그리고 좌표 시스템의 종류를 입력하여 정의할 수 있으며, 이와 관련된 명령어는 다음과 같다.

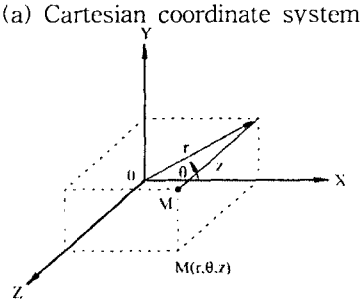
local,ic,system type,xc,yc,zc,rotxy,rotyz,rotzx
- 좌표 시스템을 정의한다.

csys,ic
- 현재의 좌표 시스템을 변경한다.

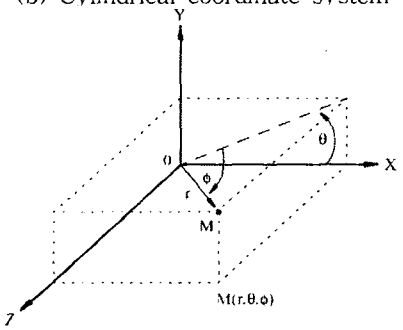
하나의 격자점은 격자점 번호(nv), 절대 좌표계에서의 격자점의 위치 x, y, z의 값을 가지게 된다. 격자점은 csys명령에 의해 결정된 좌표 시스템의 영향을 받아 만들어지며, 실제 저장은 절대 좌표계로 저장되게 된다. 격자점과



(a) Cartesian coordinate system



(b) Cylindrical coordinate system



(c) Spherical coordinate system

Fig. 4 Coordinate system

관련된 명령은 다음과 같다.

v(vertex),nv1,x,y,z
- 격자점(nv1)을 정의한다.

vcompress,nv1,nv2
- 범위(nv1,nv2)사이에 사용하지 않는 격자점번호를 압축시킨다.

vdelete,nv1,nv2,nvinc
- 범위(nv1,nv2,nvinc)안에 있는 격자점들을 제거한다.

vfill,nv1,nv2,nnum,nvstrt,nvinc,nrep,nrinc,ratio
- 이미 만들어진 격자점(nv1,nv2)사이를 격자점으로 채워진다.

vgenerate,nset,nvoff,nv1,nv2,nvinc,dx,dy,dz,ratio
- 범위(nv1,nv2,nvinc)안에 있는 이미 만들어진 격자점들을 이용하여 격자점번호는 nvoff 증가시키고 각 축으로 dx, dy, dz 증가시킨 후 격자점을 생성한다.

vlist,nv1,nv2,nvinc,ic1
- 격자점들(nv1,nv2,nvinc)의 위치를 출력한다.

vmerge,nv1,nv2,tol,xmin,xmax,ymin,ymax,zmin, zmax
- 범위(nv1,nv2)안에 있는 격자점 중 중복된 격자점을 제거한다.

vmodify,nv1,x,y,z
- 격자점(nv1)을 x, y, z로 수정한다.

vread,filename,nvoff
- filename의 파일로부터 격자점 번호를 nvoff 증가시킨 격자점 위치를 읽어온다.

vscale,nv1,nv2,nvinc,sfx,sfy,sfz
- 선택된 격자점들을 각 축에 대해 sfx, sfy, sfz만큼 늘린다.

vwrite,filename,nvoff,nv1,nv2,
- filename의 파일에 범위 안에 있는 격자점의 위치를 격자점 번호에 nvoff 증가시킨 후 저장한다.

하나의 격자(cell)는 격자번호(nc), 4개 또는 8개의 격자점(vertex)의 번호(nv), 격자 형태 번호, 격자 Table 번호를 가지게 된다. 이때 격자 형태 번호는 Star-CD와 호환을 위해 Star-CD에 사용되는 격자 형태 번호를 그대로 사용하였다. 격자와 관련된 명령은 다음과 같다.

c(cell),nv1,nv2,····,nv8
- 격자를 정의한다.

ccompress
- 사용하지 않는 격자 번호를 압축시킨다.

cdelete,nc1,nc2,ncinc
- 범위(nc1,nc2,ncinc)안에 있는 격자를 제거한다.

cgenerate,nset,nvoff,nc1,nc2,ncinc
- 범위(nc1,nc2,ncinc)안에 있는 이미 만들어진 격자들이 이용하여 격자점 번호를 nvoff 증가시켜 격자를 생성한다.

clist,nc1,nc2,ncinc
- 격자를 구성하고 있는 격자번호 및 격자 Table 번호를 출력한다.

cmodify,nc,nv1,nv2,nv3,nv4,nv5,nv6,nv7,nv8
- 격자의 구성을 수정한다.

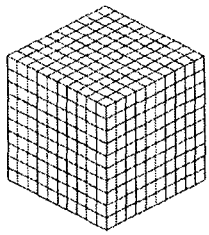
cread,filename,nvoff
- filename 파일에서 격자점 번호를 nvoff 증가시킨 격자정보를 읽어 온다.

cwrite,filename,nvoff,nc1,nc2
- filename 파일에 범위(nc1,nc2)안에 있는 격자들을 격자점 번호 nvoff 증가시킨 후 저장시킨다.

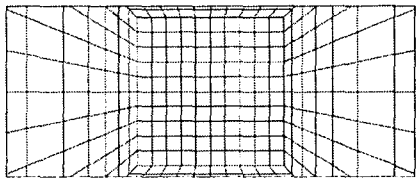
2.4 View

plot 모듈은 view와 plot으로 구성되었으며, plot에는 cplot, vplot, replot 명령이 있다. cplot은 선택된 범위의 격자(cell)를 화면에 그릴 때 사용되며, vplot은 격자점(vertex)을 화면에 출력할 때 사용된다. 격자를 화면에 그리기 위해서는 선택된 범위의 격자 중에서 가장 바깥에 있는 격자의 바깥 면만 선택해서 그려야 하므로 상당한 계산시간이 소요된다. 이를 줄이기 위하여 한번 cplot하고 나면 그 바깥 면 정보를 저장하여, view를 바꾸거나, 확대/축소하여 다시 그릴 때에 간단히 그리기 위해서 replot이란 명령을 사용할 수 있다.

본 프로그램에서는 투영 방법에 따라 직교 투영(Orthographic Projection)과 투시 투영(Perspective Projection)의 2가지의 view를 사용할 수 있다. 이 두 가지 투영방법에 대한 예는 Fig. 5에 나타내었다. 먼저 직교 투영은 원근감이 없는 투영방법으로 시선 방향으로부터 깊이가 향상 일정한 투영방법이다. 직교 투영에서는 회



(a) Orthographic Projection



(b) Perspective Projection

Fig. 5 Projection

전을 위한 rotate와 확대/축소를 위한 zoom명령이 사용 가능하다. rotate 명령은 항상 화면 우측을 x축으로 위쪽을 y축으로 한 절대 좌표에서 회전축을 계산한 후 회전을 시키며, 이 절대 좌표는 식 (1), (2)을 이용하여 회전 시마다 계산하여 저장하도록 하였다.

$$(X_n\ Y_n\ Z_n\ 1)=(X_o\ Y_o\ Z_o\ 1)Rot(n,\theta) \qquad (1)$$

$$\begin{bmatrix} n_x^2+(1-n_x^2)\cos\theta & n_xn_y(1-\cos\theta)+n_z\sin\theta \\ n_xn_y(1-\cos\theta)-n_z\sin\theta & n_y^2+(1-n_y^2)\cos\theta \\ n_xn_z(1-\cos\theta)+n_y\sin\theta & n_yn_z(1-\cos\theta)-n_x\sin\theta \\ 0 & 0 \end{bmatrix}$$

$$\begin{matrix} n_xn_z(1-\cos\theta)-n_y\sin\theta & 0 \\ n_yn_z(1-\cos\theta)+n_x\sin\theta & 0 \\ n_z^2+(1-n_z^2)\cos\theta & 0 \\ 0 & 1 \end{matrix} \Bigg] = Rot(n,\ \theta) \qquad (2)$$

여기서 사용된 변수는 다음과 같으며, 실제 회전은 OpenGL 함수를 이용하였다.

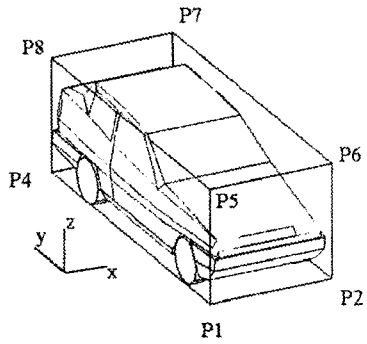


Fig. 6 Maximum Minimum Vertex

X_0, Y_0, Z_0 : 이전 좌표
X_n, Y_n, Z_n : 새로운 좌표
n_x, n_y, n_z : 회전 중심 축
θ : 회전각도

zoom off 일 때는 먼저 격자들의 각 축에서의 최대, 최소 값을 검색한 후 Fig. 6과 같이 8개의 새로운 점을 만들었다. 이 점들을 회전된 model 좌표축에서 화면 좌표에 투영하여 필요한 view의 크기를 계산하였으며, 이것을 다시 Display Window의 가로세로 비율에 맞추어 실제 view의 크기를 결정하였다. 이렇게 8개의 점들만을 view의 크기를 계산하는데 이용함으로써 화면 출력시간을 단축하였다. zoom on 일 때는 마우스의 버튼이 눌러진 위치와 버튼이 올라온 위치를 윈도우로부터 얻어 이것을 다시 view의 좌표로 변환하고, view의 x축과 y축 만을 축소하여 확대 기능을 제작하였다.

본 프로그램에서는 향후 비조직격자계(Unstructured Grid)에의 확장을 고려하여 일정한 규칙이 없이 격자를 생성할 수 있도록 하기 위하여, 이웃하고 있는 격자의 번호를 바로 알 수 없다고 가정하였다. 그래서 먼저 격자점에 자신을 포함하고 있는 격자 번호를 저장시킨다. 이때 하나의 격자점을 포함하고 있는 격자의 개수 또한 모르기 때문에 이 부분은 linked list data 구조로 처리하였다. 하나의 격자에서 이웃하고 있는 격자를 검색할 때 격자점에 있는 linked list data를 검색하여 알 수 있도록 하여 격자 검색 시간을 단축시켰다.

투시 투영은 원근감을 주는 투영 방법으로 관측자로부터 멀리 떨어진 물체를 작게 축소된 형태로 나타낸다. 투시 투영에서는 먼저 눈의 위치, 눈의 시선 방향, up 벡터로 구성되는 시선 좌표계(eye coordinate system)를 구성하여 투영하게 되며, 키보드의 화살표 키를 이용하여 이동 및 회전이 가능하도록 제작하였으며, 각 키의 조작은 다음과 같다.

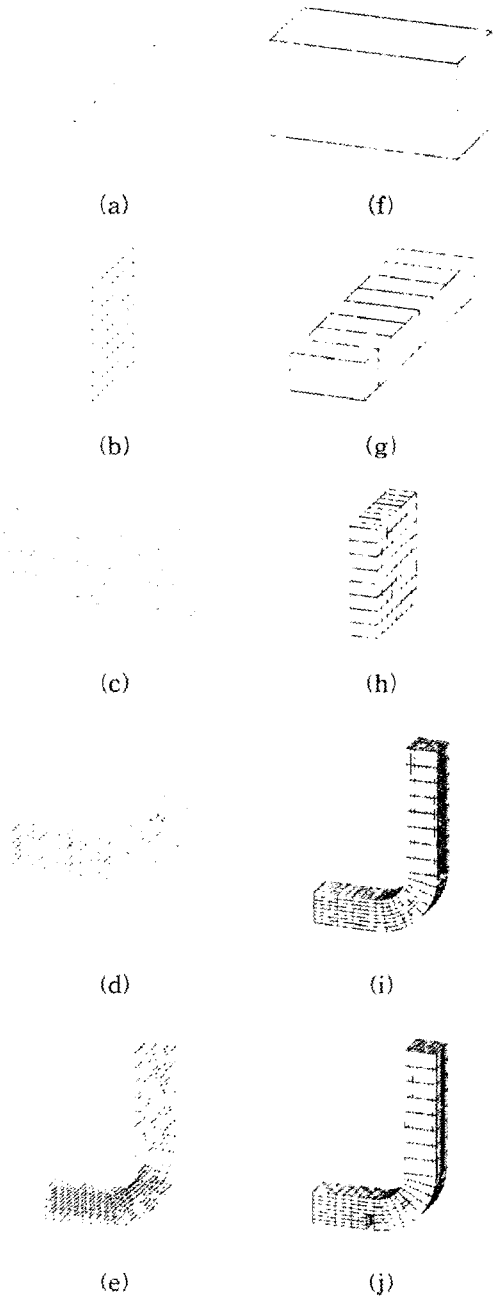
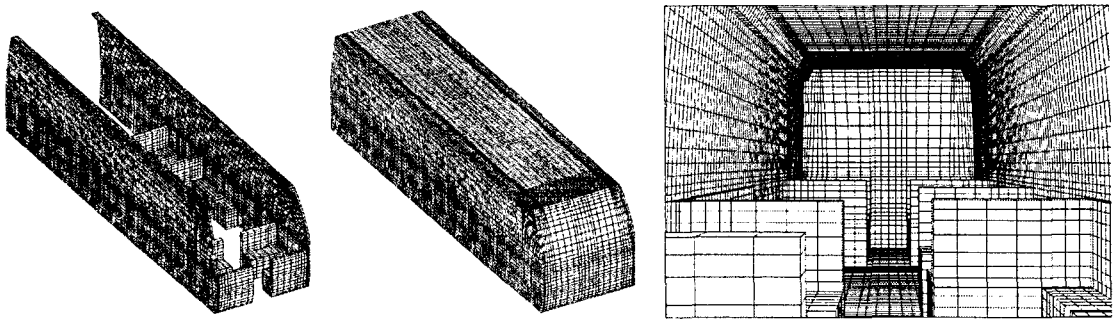


Fig. 7 Order of Mesh Generator



(a) Solid cells (b) Fluid Cells (c) Perspective view for grid test

Fig. 8 Computational grids for a train ventilation simulation

⇧ : 전진
⇩ : 후진
⇦ : 좌측으로 이동
⇨ : 우측으로 이동
alt + ⇧ : 위쪽으로 이동
alt + ⇩ : 아래쪽으로 이동
ctrl + ⇧ : 위쪽으로 회전
ctrl + ⇩ : 아래쪽으로 회전
ctrl + ⇦ : 좌측으로 회전
ctrl + ⇨ : 우측으로 회전

이렇게 관찰자의 위치를 변화시키므로 사용자가 원하는 위치에서 격자를 관찰 할 수 있도록 하였다. 시점의 위치가 격자내부에 있을 때에는 격자의 내부를 관찰할 수 있도록 하였다. 화각(field of view angle)은 60 °로 하였다.

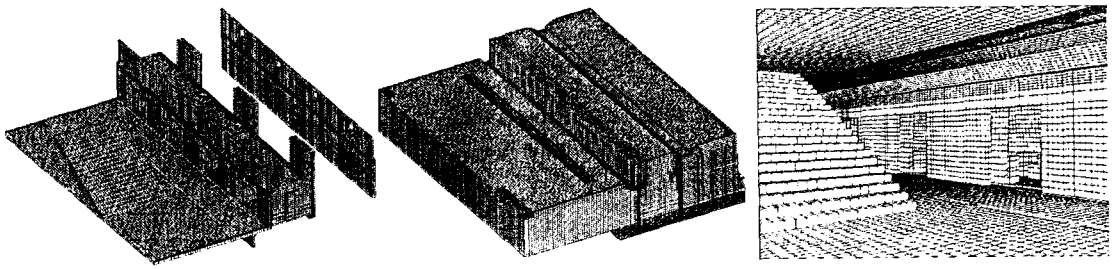
3. 적용 예

3.1 격자 생성

먼저 프로그램을 테스트하기 위해 Star-CD Tutorials에 있는 Partially Blocked Elbow 격자를 생성하여 Star-CD와 비교하였다. 격자생성 과정은 Fig. 7 (a)~(j)에 나타내었다. (a)~(e)는 격자점 생성과정이며, (f)~(j)는 격자 생성과정이다. (a)에서는 v(vertex)명령을 사용하여 2개의 격자점을 생성한 후 vfill 명령을 사용하여 사이를 채워 선상에 격자점을 만들었다. (b)에서는 (a)에서 만든 격자점을 vgenerate 명령을 사용하여 위쪽으로 복사하여 면으로 만들었으며, (c)에서는 (b)에서와 마찬가지로 vgenerate 명령을 사용하여 육면체 형상으로 만들었다. (d)에서는 먼저 local 명령을 사용하여 좌표 시스템의 원점이 elbow의 회전 중심인 원통 좌표 시스템을 만들었다. 다음 csys 명령을 사용하여 좌표 시스템을 변경한 후 vgenerate 명령을 사용하여 (d)와 같은 형상을 만들었다. (e)에서는 csys 명령을 사용하여 원래 좌표 시스템으로 변경한 후 vgenerate명령을 사용하여 격자점의 최종 형상인 (e)를 만들었다. (f)는 c(cell) 명령을 사용하여 격자 하나를 생성하였으며, (g), (h), (i)에서는 cgenerate명령을 사용하여 격자를 생성하여 (i)와 같은 형상을 만들었다. (j)에서는 cdelete 명령을 사용하여 block부분의 격자를 삭제하여 최종형상인 (j)를 얻을 수 있다.

3.2 투시 투영

본 연구에서 개발된 프로그램의 투시 투영 성능 검증을 위하여 본 실험실에서 수행하고 있는 연구과제 중 경량전철[13]과 지하철[14] 격자를 가지고 테스트해 보았다. 투시 투영은 격자를 생성한 후에 그 격자를 살펴보기 위하여 꼭 필요한 도구이다. Fig. 8에는 경량전철을 나타내었으며, Fig. 9에는 지하철을 나타내었다. 두 개의 그림에서 (a)는 solid 만을 직교 투영으로 본 것이며, (b)는 전체 격자를 직교투영으로 본 그림이다. Fig. 8 (c)는 투시 투영으로 시점을 열차 내부에 위치 시켜 격자 내부를 관찰한 그림이다. Fig. 9(c)는 투시 투영으로 시점의 위치를 전체 격자의 좌측에 위치시키고, 시선을 우측으로 회전시킨 상태이다. 여기서 모든 그림들이 3차원 형상과 hidden line를 잘 처리하고 있는 것을 볼 수 있다. 여기서 hidden line은 OpenGL을 사용함으로써 처리되었다. 경량 전철을 투시 투영으로 본 경우 의자 배치와 전철내부를 잘 표현되고 있다. 또한 관측자로부터 멀리 떨어지면서 의자



(a) Solid cells (b) Fluid cells (c) Perspective view for grid test

Fig. 9 Computational grids for ventilation simulation of a subway station

의 크기가 감소하면서 원근감이 잘 표현되고 있다. 지하철에서는 역사의 내부의 층계와 screen door, screen door 너머로 전철의 형상을 잘 표현하고 있다. 이렇게 격자의 내부를 관찰함으로써 생성된 격자의 질을 확인 할 수 있다. 실제 프로그램에서는 마우스를 조작하여 격자 내부의 여러 지점을 직접 들어가서 살펴 볼 수 있으므로, 가상현실과 같은 기능으로 생성된 격자의 형태와 질을 확인할 수 있다.

4. 결론

본 연구에서는 CFD 해석을 위한 범용 3차원 격자생성 프로그램인 Pre-Turbo 프로그램을 개발하였다. 이 프로그램은 격자점과 격자의 생성, 그리고 3차원 그래픽 처리 및 축소, 확대, 회전등의 기능 등을 가지고 있다. 특히, 본 프로그램에서는 격자를 화면상에 나타낼 때 투영 방법을 선택할 수 있게 하였으며, 투시투영인 경우 시점의 위치를 이동시키면서 격자를 관찰할 수 있도록 하였다. 이 투시 투영법은 마우스의 조작으로 일종의 가상현실 기법과 같이 직접 격자 내부를 이동하며 격자의 형태 및 질을 관찰할 수 있다. 이 프로그램은 향후 본 연구실에서 개발하고 있는 Turbo-3D solver와 연결하여 package 형태로 개발을 진행할 계획이다.

참고 문헌

[1] http://www.cd.co.uk
[2] http://www.fluent.com
[3] http://www.pointwise.com
[4] http://www.truegrid.com
[5] http://www.altair.com
[6] http://www.icemcfd.com
[7] 김병수, "Delaunay 삼각화 기법을 활용한 다중-블록 정렬 격자의 자동 생성 기법," 한국전산유체공학회 1999년도 추계학술대회논문집, pp.108-114. 1999.
[8] 이상욱, 권장혁, 권오준, "선형 격자 형성 방정식을 이용한 직교 격자 형성에 관한 연구," 한국전산유체공학회지, 제5권 제1호, pp.14- 19, 2000.
[9] 박종렬, 사종엽, "Geometry-based Adaptive Octree 방법에 대한 고찰," 한국전산유체공학회 2000년도 추계학술대회논문집, pp.86-91. 2000.
[10] 김사량, 이경헌, 허남건, "CFD프로그램을 위한 전처리 장치의 개발(I)," 강릉대학교 공학연구소 논문집 제4권 제1호, pp. 71-76, 1996.
[11] 장재원, 허남건, "화재 simulation을 위한 Po- st Processor 개발," 한국전산유체공학회 2001년도 춘계학술대회논문집, pp.155-160, 2001.
[12] http://www.opengl.org
[13] 허남건, "경량전철 시스템 안전프로그램 계획을 위한 화재 방재 대책수립," 경량전철시스템 기술개발사업 2차년도 연구결과보고서, 한국천도기술연구원, 2000.
[14] 허남건, "지하철 9호선 TES방식 환기시스템의 환기효율연구," (주)화승엔지니어링 보고서, 2001.(발간중)