가시화 프로그램에서의 데이터 구조와 가시화 알고리즘

나 정 수*1, 김 기 영*2, 김 병 수*3

Data Structure and Visualization Algorithm in a Post-processing Program

J. S. Na and K. Y. Kim and B. S. Kim

Post-processing programs play an important role in the CFD data visualization and analysis. A variety of post-processing softwares have been developed and are being used in the CFD community. Developing a good quality of post-processing program requires dedication and efforts. In this paper an experience obtained through previous studies and developing post-processing programs are introduced which includes data structure and visualization algorithms.

Key Words: 객체지향(OOP), 클래스(class), 동적 메모리(dynamic allocation), 컨투어(contour), 등가면(ios-surface), 스트림라인(streamline), 스트림리본(stream ribbon)

1. 서 론

전산유체역학 분야 중 전처리 과정과 해석 분야는 많은 연구를 통하여 국내 기술이 어느 정도 축적되어 있어 우리 자체 기술에 의한 훈련기, 무인기, 비행선, 로켓 등의 개발에 CFD는 풍동실험과 함께 핵심적인 기여를 하고 있다. 더구나 최근의 국가 그리드 연구와 더불어 CFD를 통한 연구가 가속될 가능성이 점점 높아지고 있다. 이러한 환경에서는 다자간 연구를 하고 서로의 연구 정보를 공유하기 위해서 실제적인 가시화 도구가 필수적이라 할 수 있을 것이다. 현재 그리드 환경에서 사용할 수 있는 가시화 프로그램을 연구하고 있는 곳도 있다.[1] 따라서 이러한 경향에 발 맞추어 국내에서도 자체 연구자료를 확보한다는 것은 실제적인 도구 개발 능력 확보 면에서 중요하다. 외국에서는 일찍이 80 년대부터 개발을 시작하여 현재는 여러 갈래의 상용프로그램들이 사용되고 있지만 국내에서는 아직 개발 노력이 많이 진행되지 않은 실정이다.

이러한 후처리 프로그램을 개발하는 데에는 프로그램의 구조를 설계하고 읽어들인 데이터를 클래스로 구조화된 데이터로 바꾸고 이렇게 구조화 된 데이터에 가시화 알고리즘을 사용해 구현하는 과정을 거치게 된다. 데이터 구조는 크게 정렬격자구조와 비 정렬격자구조로 나뉘고 multi block으로 구성되어 있을 수 있는데 각각에 대하여 고려해야 할 사항이 있다. 가시화 방법에는 스칼라 형태로 가시화 하는 경우, 벡터형태로 가시화 해야 하는 경우 그리고 탐침 기능으로 데이터를 관찰하는 경우 등이 있다. 데이터 형태가 여러 가지인 경우, 그리고 가시화 구현기능들이 많아지면 프로그램이 복잡해질 수밖에 없다. 이러한 단점을 보완하고자 최근의 프로그램들은 대부분 객체지향개념을 반영하여 프로그램 되고 있다. 개발 중인 본 프로그램(DAVA, DAta Visualization and Analysis[2])도 앞으로의 확장성을 고려하여 객체지향 프로그램이 되고 있다.

*1 학생회원, 충남대학교 대학원 항공우주공학과
*2 학생회원, 충남대학교 대학원 항공우주공학과
*3 종신회원, 충남대학교 항공우주공학과

본 논문에서는 많은 연구가 진행되어 있는 외국 자료를 바탕으로 정렬격자와 비 정렬격자의 클래스화된 구조화 작업에 있어서 필요한 사항과 가시화 알고리즘, 개발중인 프로그램에 적용되고 있는 방법에 대해서 소개하고자 한다.

2. 프로그램의 구조화

2.1 프로그램의 순서도

가시화 프로그램이 실행되는 기본 절차는 아래 그림 Fig. 1 과 같다. 먼저 데이터를 읽어 들인 후 프로그램은 사용자에게 받음각, 레이놀즈수, 마하수를 입력할 것을 요구한다. 이 정보는 프로그램이 실행되는 동안 화면에 report 하게 되며 텍스트 파일로 저장되게 된다. 데이터를 읽은 후 데이터를 정렬격자와 비 정렬격자로 구분하여 메모리를 동적으로 생성하고 클래스화 된 구조로 저장한다.

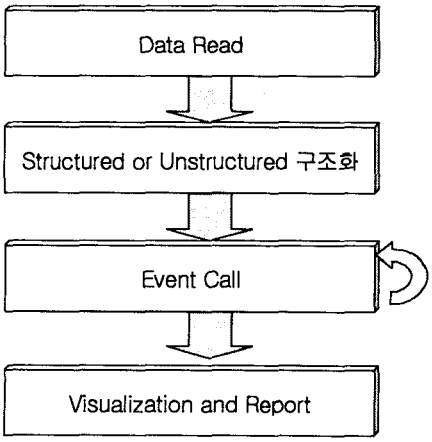

Fig. 1 순서도

Qt[3]에서 지원하는 GUI를 사용하여 Event call 하면 가시화 알고리즘을 데이터에 적용하여 계산 후 기능을 수행하고 필요한 정보를 나타내준다.

2.2 클래스를 이용한 객체지향 프로그램

본 프로그램 개발에 있어 격자 구조에 따라 그리고 Cell 의 구조에 따라 다양한 데이터 형태에 따라 일반적으로 처리하고 지속적인 확장 및 관리를 위해 객체지향개념을 반영하여 클래스구조를 만들게 되었다.

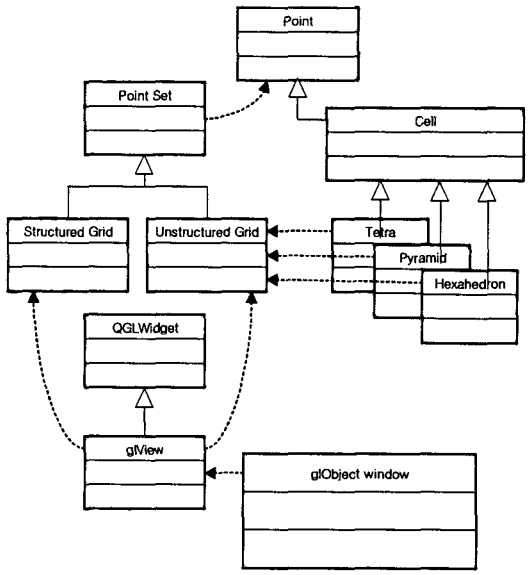

Fig. 2 클래스 상속도

본 프로그램의 클래스 상속도는 Fig. 2 에서와 같이 나와 있는데 여기서 실선은 상속관계를 그리고 점선은 협력관계를 나타내고 있다. 정렬격자와 비 정렬격자는 PointSet 클래스로부터 상속을 받고 이곳에서는 격자위치와 종속변수가 저장되는 형태를 제공하는 Point 클래스의 객체 집합을 가지게 된다. 비 정렬격자는 Tetra, Pyramid, Hexahedron 클래스 형태의 객체들을 사용하는데 이 클래스들은 가상함수가 구현되어 있는 Cell 클래스로부터 상속을 받는데 이렇게 함으로써 다형성을 지원할 수 있다. 이러한 다형성의 장점은 상속받는 객체들을 하나로 표현함으로써 프로그램이 구조화되고 간단해 진다는 것이다.

glObjectwindow 클래스의 GUI를 통해 명령을 수행 할 수 있도록 되어 있고 glView 클래스에서 3차원 그래픽 화면을 보게 된다. 프로그램이 추가 될 경우 다시 클래스를 만들어 glObjectwindow 클래스에서 선언하여 사용할 수 있도록 한다.

3. 클래스화 된 데이터 구조화

계산과정에서 사용되는 메모리는 주로 배열을 사용해 왔고 본 프로그램의 초기 버전에서도 이 데이터를 배열을 사용한 메모리에 정렬격자 데이터를 읽어 들이도록 되어 있었다. 이러한 방법은 멀티 블락

일 경우 배열을 미리 선언 해 놓으므로 서 처리를 하게 되는데 I, J, K 크기에 맞는 메모리를 할당하지 못하는 단점이 있었다. 이것은 거대문제 해석이 시도되고 있는 시점을 생각하면 해결되어야만 할 문제이다.

3.1 정렬격자 클래스

본 프로그램은 I, J, K 정보로부터 single block 의 배열 크기, 그리고 각 점 데이터의 종속변수 개수로부터 Point 클래스의 변수 저장공간을 동적으로 생성한다. 이러한 single block 들의 포인터 배열을 동적으로 생성함으로써 multi block을 처리 할 수 있도록 되어 있다.

각각의 single block 들의 데이터를 저장하면서 데이터처리에 필요한 정보들을 찾아내 저장하고 그에 필요한 멤버 함수들이 구현되어 있다. multi block에서 어느 특정공간을 검색하기 할 때 모든 I, J, K 인덱스에 대해서 검색한다는 것은 비효율적이므로 x, y, z 데이터의 최대/최소 값을 검색해서 저장하고 우선적으로 이 정보를 비교함으로써 해당 데이터가 없는 block 는 건너뛰도록 한다. 각각의 single block 클래스에 Blank on/off 정보를 기억할 수 있도록 하고 데이터의 경계에 해당하는 모서리를 알 수 있도록 되어 있다.

multi block 인 경우 각각의 block 간의 연결상태를 체크하여 정보를 가지고 있는데 이 정보는 스트림라인을 구현 할 때 해당 block의 경계 면에 인접한 다음 block으로 진입할 때 해당 데이터의 index를 찾는데 중요하게 사용된다.

동적으로 메모리를 생성 후 데이터를 저장 할 때 index 1부터 저장하게 되는데 각 I, J, K 마다 앞뒤로 가상의 데이터를 저장 할 수 있는 공간을 확보하게 되는데 이 곳에 block 들이 접한 면에 대해서는 데이터를 공유 할 수 있는 공간으로 하고 그렇지 않은 면에 대해서는 스트림라인을 위한 데이터 공간으로 한다. 즉, single block 의 가장 외각의 위치정보 x, y, z 의 최대, 최소가 이루는 박스의 영역을 완전히 포함 할 수 있도록 함으로써 스트림라인의 시작점인 seed point를 해당 데이터 앞부분부터 실행 할 수 있도록 한다.

3.2 비 정렬격자 클래스

정렬격자는 그 자체로 데이터로부터 정보를 얻는데 상당한 이점이 있지만 비 정렬격자 인 경우에는 추가적으로 각각의 셀 에 대하여 필요한 정보들을 가지고 있어야 한다. 비 정렬격자 데이터는 point 데이터와 데이터들이 어떻게 셀 들이 구성하고 있는지에 대한 연결성 정보를 기록하고 있다. 따라서 point 데이터는 Point 클래스를 이용하여 글로벌(global) 데이터로 저장하고 연결성 정보는 연결성에 대한 정보는 셀 번호와 셀 을 이루는 노드 점들을 트리 구조로 저장한다. 본 프로그램은 데이터는 Cell 클래스를 사용하여 데이터를 저장하고 가시화에 필요한 정보를 제공하는 멤버 함수를 구현하고 있다. Cell 클래스는 각각의 셀 에 대한 일반적이고 추상적인 개념을 가상함수로 구현하고 있고 이를 상속받은 각각의 클래스들은 구체화된 실제적인 기능을 구현한다. 각각의 하위 클래스에서 자신의 클래스에 맞는 멤버 함수들을 구현하여 가시화에 필요한 기능들을 구현한다. 대표적인 정보로는 셀 들을 구분할 수 있는 id 값, 이웃하고 있는 셀 들의 정보, 셀의 Min, Max, 셀 의 중앙점, contour, edge 와 face 정보 등이다.

4. 가시화 알고리즘

4.1 스칼라

4.1.1 I, J, K surface contour

contour를 그리기 위한 방법으로 많이 쓰이고 있는 방법은 marching square[4] 알고리즘이다. 하나의 셀에서 가능한 경우는 Fig. 3처럼 모두 16 가지이지만 다음과 같은 대칭성을 고려 할 수 있다.

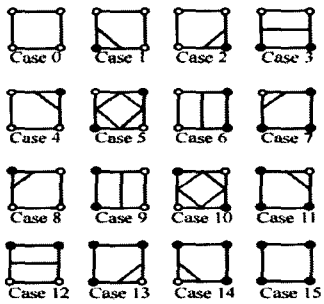

Fig. 3 marching square의 16 가지 경우

Case 1과 Case 2의 경우처럼 회전을 통해서 같은 사각형으로 변환될 수 있는 모든 경우들은 정점 중 하나를 절단하는 하나의 선분을 갖는다. 또 다른 대칭성은 Case 0과 Case 15번의 경우처럼, 검은색과

흰색을 바꿈으로써 같아질 수 있는 경우들 사이에서의 대칭성이다. 이러한 대칭성까지 감안한다면 네 가지의 서로 다른 경우들만 남기 때문에 이들만 처리해 주면 된다.

Fig. 4 marching square의 최종 4가지 경우

그림 4 같이 명료하지 않은 경우가 생기지만, 이 같은 경우는 그래픽으로 나타내어 질 때 매우 가깝기 때문에 예외처리를 하지 않고 어느 한 경우만 고려해주기로 한다.

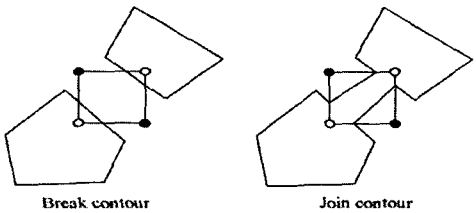

Fig. 5 marching square 방법의 예외 처리

개발중인 프로그램에서는 contour level을 바꿀 수 있게 되어 있으며 level에 따라 색으로 구분을 해 주고 있다.

4.1.2 Iso-surface

iso-surfaces를 구현하기 위한 알고리즘으로 널리 사용되고 있는 방법은 Marching Cubes[5] 이다. contour 에서와 비슷하게 대칭성을 고려하면 Fig. 7 의 14가지 경우가 있을 수 있다. contour 의 경우처럼 계산한 edge 상의 점을 이어 iso-surface 영역 내부와 외부 사이의 cube를 분리하는 삼각 patches를 만들 수 있다. 한 개 이상의 삼각형 patches 도 iso-value가 속해있는 edge를 구하고 이 edge에서 point 간의 선형 보간을 통해 isovalue point를 구한다. 2^8 = 256가지의 경우에 대해 리스트를 작성하고 요건에 맞는 경우를 그려주면 marching cube를 구현할 수 있다. 예를 들면, v1, v3, v4 가 선택되었다면

$2^0+2^2+2^3=13$ 을 이용하여 13번째 리스트의 경우를 디스플레이 해준다.

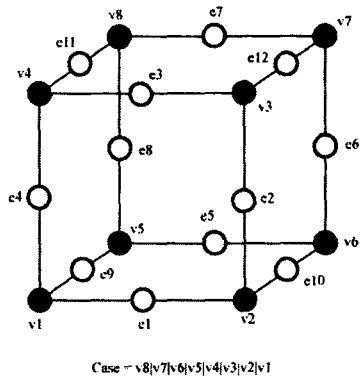

Fig. 6 marching cube

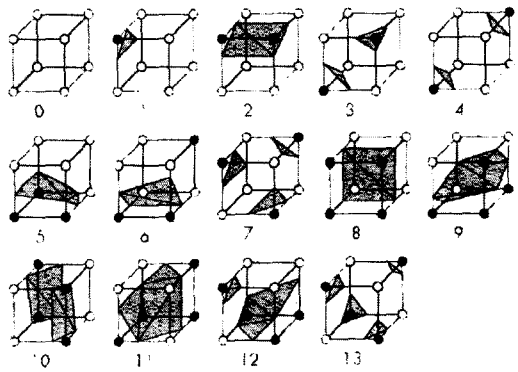

Fig. 7 marching cube의 14가지의 경우

위의 경우를 정리하면,

1. Cell을 선택
2. Cell의 각 vertex의 내/외부 상태를 계산
3. Edge lists를 생성
4. Edge Tabel에서 Cell의 상태를 찾아 읽기
5. Table에 있는 각각의 edge에서 두 꼭지점의 가중치를 선형 보간 하여 edge 에서의 좌표를 찾기

4.2 벡터 방법

4.2.1 Streamline

벡터들의 접선을 따라 이은 선이 스트림라인 이다. 스트림라인을 구현하는 절차는 먼저 데이터 물리적인 공간상에 좌표 x, y, z를 갖는 시작점 seed 포인트가 어느 block 에 있는지, 어느 I, J, K index 셀에 위치해 있는지를 검색하는 단계, 해당 셀의 꼭

지점으로부터 가중치를 곱하여 속도 벡터 $\vec{u}(\vec{\xi})$ 를 보간 하는 단계, 그리고 일정한 시간 간격으로 적분을 하여 포인트의 위치를 구해내고 화면에 그려주는 단계를 거치게 된다.

먼저 검색과정에서 2차원은 해당 셀 을 검색하기 위한 방법으로 Area Test[6]를 사용할 수 있지만 3 차원에서는 이 방법이 적용되지 않는다. Strid and Eliasson[7] 등이 사용한 방법에 의하면 임의의 point $\vec{x}$ 에 해당하는 computational 공간의 $\vec{\xi}$ 를 아래 Fig. 8 에서 구하고 적당한 셀

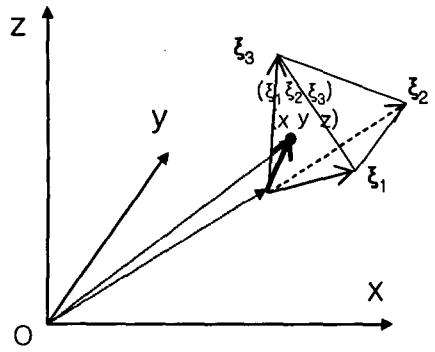

Fig. 8 computational 공간의 좌표

에 대해서 Newton-Raphson 방법으로 수렴 여부를 조사하게 된다. 각각의 셀 타입에 대해서 computational 공간 (ξ_1, ξ_2, ξ_3) 의 Fig. 9 와 같이 standard 셀 들이 존재한다.

이들과의 mapping 은 수식(1)과 같이 표현된다.

$$\vec{x}(\vec{\xi}) = \sum_j f_j(\vec{\xi})\,\vec{x}_j \qquad (1)$$

여기서 각 셀의 노드 개 수 만큼 합을 하고 $\vec{x}_j$ 는 j^{th} 노드의 coordinate vector 이다. f_j 는 j 번째 노드에서는 1 이고 나머지에서는 0 이 되도록 다음과 같이 만들어 준다. 대표적인 tetra 와 hexahedra 의 경우 수식 (3), (4)와 같다.[8]

초기 $\vec{\xi}^{(0)}$는 추측한 셀의 중앙값을 취하고 다음 수식 (2)과 같이 반복을 한다.

$$\vec{\xi}^{(n+1)} = \vec{\xi}^{(n)} + \left(\frac{\partial \vec{x}}{\partial \vec{\xi}}\right)^{-1}(\vec{x} - \vec{x}^{(n)}) \qquad (2)$$

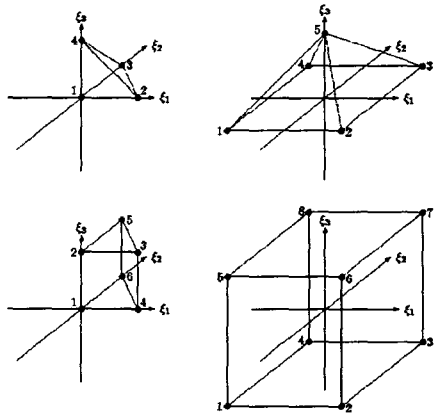

Fig. 9 computational 의 standard 셀

tetrahedra

$$f_1 = 1-\xi_1-\xi_2-\xi_3 \qquad (3)$$
$$f_2 = \xi_1$$
$$f_3 = \xi_2$$
$$f_4 = \xi_3$$

hexahedra

$$f_1 = \frac{1}{8}(1-\xi_1)(1-\xi_2)(1-\xi_3)$$
$$f_2 = \frac{1}{8}(1+\xi_1)(1-\xi_2)(1-\xi_3)$$
$$f_3 = \frac{1}{8}(1+\xi_1)(1+\xi_2)(1-\xi_3)$$
$$f_4 = \frac{1}{8}(1-\xi_1)(1+\xi_2)(1-\xi_3) \qquad (4)$$
$$f_5 = \frac{1}{8}(1-\xi_1)(1-\xi_2)(1+\xi_3)$$
$$f_6 = \frac{1}{8}(1+\xi_1)(1-\xi_2)(1+\xi_3)$$
$$f_7 = \frac{1}{8}(1+\xi_1)(1+\xi_2)(1+\xi_3)$$
$$f_8 = \frac{1}{8}(1-\xi_1)(1+\xi_2)(1+\xi_3)$$

$\vec{x}$에 해당하는 셀에 대해서 반복을 하면 수렴하게 되고 수렴하지 않으면 이웃하는 셀을 다시 위의 절차를 반복하여 수렴하는 셀을 찾는다. 또 다른 방법으로 본 프로그램에 적용되고 있는 방법은 보다 빠른 검색을 위하여 먼저 min, max test[6]를 통해 가능성이 높은 셀 들을 찾아 낸 다음 그 셀에 있는지를 조사하는 방법으로 I 방향에 대하여 각각 셀에 평행한 평면을 생각하면 normal vector 와 추측을 하고 있는 $\vec{x}$ 와의 inner product 들을 계산한다.

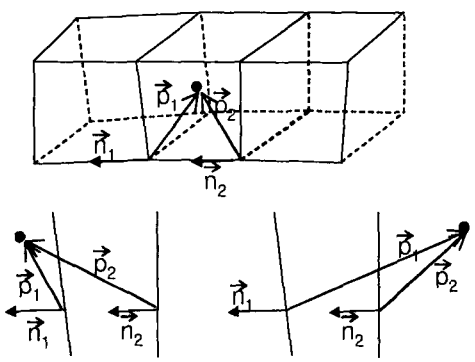

Fig. 12 셀 내부의 존재여부 판단

$\vec{x}$가 두 평면 사이에 있으면 이 둘을 곱했을 때 음의 값이 나오게 되고 밖에 있으면 양의 값이 나오게 된다. 이 과정을 나머지 J, K 방향에 대해서 거치고 만약 이들 중 양의 값이 나오면 바로 옆의 셀 에 대해서 조사를 한다.

다음으로 찾아진 셀 의 꼭지점 좌표와 $\vec{x}$ 로 속도 벡터 $\vec{u}(\vec{\xi})$ 를 보간을 하게 되는데, 2 차원에서는 bi-linear interpolation 방법을 사용할 수 있지만 3 차원에 적용하기에는 번거로운 작업이 많이 있으므로 구해진 $\vec{\xi}$ 와 f_j 로부터 수식 (5)

$$\vec{u}(\vec{\xi}) = \sum_j f_j(\vec{\xi})\,\vec{u}_j \tag{5}$$

과 같이 구한다.

마지막으로 $\vec{u}$ 로 부터 오일러 적분이나 런지쿠타 적분으로 $\vec{x}$ 를 구하여 전진을 하면서 자취를 저장하게 된다.

4.2.2 Stream ribbons

Stream ribbons은 stream line에 약간의 width를 주어 표현하는데 stream 방향의 vorticity를 나타낼 수 있다. 본 프로그램의 초기단계에서 두 개의 stream line을 이어서 표현하고 있지만 변화가 심한 영역에서는 이들이 발산하는 경우가 있어 회전하는 정도를 표현하기에는 문제가 있다. Visual3[8] 에서 사용하고 있는 방법은 하나의 stream line을 그려나가면서 stream 방향에 수직으로 일정한 길이의 타일을 붙여 가면서 회전하는 정도를 다음과 같이 구하여 적용한다.

$$\frac{d\theta}{dt} = \frac{1}{2}(\vec{\omega}\cdot\vec{s})$$
$$\vec{s} = \frac{\vec{u}}{|\vec{u}|} \tag{6}$$
$$\vec{\omega} = \nabla \times \vec{u}$$

4.3 탐 침

탐침의 방법에는 공간상의 어떤 점의 종속변수의 값에 대한 정보를 알 수 있도록 point probe 기능과 line에 대하여 관찰 할 수 있는 line probe, 그리고 streamline 상의 변화를 살펴 볼 수 있는 streamline probe 등이 있다.

5. 결 론

본 논문에서는 유체 후처리 프로그램에 필요한 데이터들의 효율적인 가시화를 위해 객체지향개념을 도입하고 있는 모습과 가시화 알고리즘들에 대해서 제시하였다. 개발되고 있는 본 프로그램은 비 정렬격자 가시화 기능 구현이 지속되고 있다.

참고문헌

[1] http://www.hlrs.de/organization/vis/projects/
[2] 나정수, 김기영, 김병수 "입체구현기능을 지닌 데이터 분석 및 가시화 프로그램의 개발", 한국전산유체공학회 2002년도 춘계학술논문집, p.158-163, (2002)
[3] www.trolltec.com
[4] *openGL을 이용한 컴퓨터 그래픽스 영진출판사.* Edward Angel
[5] *Introduction to computer Graphics* Foley, Van DAm
[6] 옥영종, Data 후처리용 다기능 Software의 개발. 석사학위논문, 충남대학교, (2002)
[7] T. Strid and A. Rizzi. Development and use of some flow visualization algorithms. VKI Lecture Series on Computer Graphics and Flow Visualization in CFD, (1989)
[8] R. Haimes and M. Giles. Visual3 - A Software Environment for flow Visualization, (1991)