Visual Basic을 이용한 구조해석 통합환경 시스템 개발에 관한 연구

A Study on the Development of Structural Analysis Integrated Environment System using Visual Basic

장 승 조*, 이 상 갑**[1]

Chang Seung-cho, Lee Sang-Gab

Abstract

The objective of this paper is to develop a finite element structural analysis program using Visual Basic(VB) which has been recently getting popular as a development tool of Windows application program. VB provides several functions such as object based system design and object linking embedding etc. easily to develop an application program by supporting event driven programming method and graphic object as control data type.

This system is an integrated processor including preprocessor, solver and postprocessor. The preprocessing stage involves the preparation of input data, such as nodal coordinates, element connectivities, boundary conditions, applied loads and material properties. Automatic mesh generation is also available at this stage. Stiffness generation, stiffness modification and solution of equations are made at the solution st , resulting in the evaluation of nodal variables. The last postprocessing one deals wit the presentation of results. Typically, the deformed configuration and stresses dist ribution are computed and displayed at this stage.

1 * 한국해양대학교 조선공학과 석사과정

** 한국해양대학교 조선공학과 교수

1. 서 론

최근 PC(Personal Computer)의 성능향상 및 운영체계(operating system)의 발전은 PC의 단점이던 처리속도와 GUI(Graphic User Interface), 다중처리(multi-tasking), 네트워킹(net working)등의 개선으로 응용범위가 크게 향상되었다. 마이크로 소프트(Microsoft)사에 의해 개발된 윈도우즈(Windows)[1]의 출현은 이러한 운영 체계의 발전을 대표하는 것으로 볼 수 있다. 운영 체계의 발전은 그 동안 고성능화된 컴퓨터 하드웨어(hardware)의 기능을 소프트웨어(software)적으로 충분히 활용할 수 없었던 단점들을 개선할 수 있게 하였다. 그러나 새로운 운영 체계의 발전은 그 운영 체계에 맞는 프로그램의 개발을 필요로 하게 되고 기존의 운영 체계에서 지속적으로 사용되던 프로그램의 재설계를 필요로 하게 되었다.

구조해석 및 설계분야에도 PC환경에서 구동이 가능한 SAP90, XETABS, MicroFEAP, MicroAIT등 프로그램들이 있지만, DOS(Disk Operating System) 환경에서 주로 매크로 명령어(macro command)를 통한 파일이나 화면입력(TUI : Text User Interface)으로 인한 전·후처리(pre·post process)의 미비는 사용하기에 많은 불편함이 있다[2]. 그리고 새로운 운영체계에 맞게 재설계가 필요하다고 할 수 있으나 윈도우즈 프로그램의 경우 사용자 입장에서는 편리성을 제공하지만 프로그램 개발에는 많은 어려움이 있다. 즉, 기존의 운영체계와는 근본적으로 시스템 운영 방법에 차이가 있고 다양한 기능을 제공하는 만큼 새로운 개념들에 대한 이해가 선행되어야만 한다. 특히 단순한 GUI 코드(code)만을 작성하는 데에도 상당한 시간과 노력이 필요하고 그만큼 코드의 양도 많아진다.

이러한 문제점들을 개선하기 위해 **Borland C++**[3]나 **Turbo Pascal for Windows**[4], **Visual Basic**(VB)등과 같은 개발 도구(development tool)가 제작되어 윈도우즈 응용프로그램 개발을 좀 더 쉽게 하고 있다. 그 중에서도 VB는 다른 도구들보다 윈도우즈용 응용프로그램 개발을 훨씬 쉽게 할 수있다. VB는 본질적인 의미의 객체 지향(object oriented)[6]을 지원하지는 않지만 컨트롤 형태(control type)의 그래픽 객체(graphic object)를 이용하여 개개의 요소로써 프로그램을 조립하듯 제작할 수 있는 기능들을 제공하고 있다.

VB에 있는 그래픽 객체들은 윈도우즈 응용 프로그램에서 필요로 하는 객체와 그 객체가 반응하는 사건(event)들을 이미 갖추고 있으며 프로그램 인터페이스(interface) 구성에 필요한 요소들도 거의 갖추고 있다[7]. 또한 윈도우즈 API(Application Programmer's Interface)[8]의 이용과 다중처리시 발생하는 메모리(memory)의 문제점들을 위하여 다른 언어로 제작된 동적 연결 라이브러리(DLL: Dynamic Link Library)와의 접속은 VB의 기능을 크게 확장시킬 수 있고 개발 프로그램에 접속하는 과정도 단순하게 이루어진다. 그 밖에도 동적 자료 교환(DDE: Dynamic Data Exchange)과 객체 연결 및 삽입(OLE: Object Linking Embedding)[9] 기능을 이용하여 다른 응용 프로그램들과의 자료 교환 및 제어를 할 수 있다.

본 연구에서는 윈도우즈가 제공하는 다양한 기능을 이용하여 기존의 PC용 구조해석 프로그램들이 가지고 있던 전·후처리의 문제점들을 개선하고 윈도우즈 운영 체계에 맞는 구조해석 프로그램을 쉽게 효율적으로 설계하는 방법 등을 제시하고자 한다.

개발 시스템은 전처리(preprocess), 해석(solution), 후처리(postprocess)의 3단계로 이루어지는 통합 운영 시스템(integrated process system)이다. 전처리기 단계에서는 사용자

정의 요소분할 생성(user definition mesh generation)과 자동 요소분할 생성(automatic mesh generation)을 통한 그래픽 모델링(graphic modeling)기능을 포함하고 있고, 후처리 단계에서는 변형(deformation)과 응력분포(stress distribution)등의 결과를 그래픽으로 표현할 수 있다. 본 논문에서는 프로그램의 전·후처리 설계와 적용 개념등을 중점적으로 다루었다.

2. 시스템 설계 개요

본 개발 시스템은 PC의 윈도우즈 운영체계가 지원하는 GUI, OLE, API등 다양한 기능들을 적용하여 개발된 구조해석 프로그램으로, 전·후처리 단계와 해석 단계가 통합된 통합운영 시스템이다. 그리고 각 단계에서는 해당 부메뉴를 포함하고 있어 세부적인 처리 과정을 갖는다. 시스템의 전체적인 자료의 처리 과정은 Fig. 2.1과 같다.

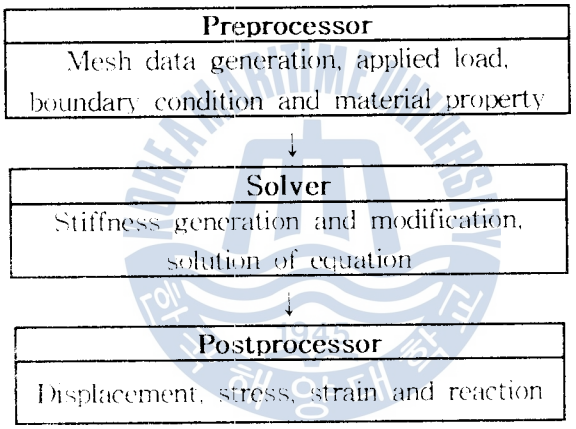

Fig. 2.1 Process of system operation

전처리기는 해석에 필요한 자료를 입력하는 단계로 입력된 자료의 저장, 수정 및 그래픽 모델링 기능을 갖는다. 해석단계는 전처리기에서 입력된 자료를 이용하여 각 절점에 대한 변위를 구하여 후처리기에서 사용할 수 있도록 파일이나 변수에 값을 저장하는 과정이다. 후처리기에서는 해석 단계에서 구해진 변위를 이용하여 응력 분포 및 반력을 구하여 그래픽 표현 및 텍스트 출력 등을 하게 된다. 따라서 다른 단계보다 그래픽과 파일처리 방법이 다양하며 많은 양의 코딩을 필요로 하게 된다.

화면(screen)상에 나타나는 시스템의 초기 구동 화면은 Fig. 2.2와 같다. 시스템 구성 요소는 크게 상단에 위치한 주메뉴 영역(main menu area)과 좌측 부분에 위치한 부메뉴 영역(sub menu area) 그리고 시스템 중앙의 그래픽 표현 영역(graphic presentation area) 으로 나뉘어진다. 주메뉴 영역은 VB에서 제공되는 버튼 객체로써 구성되고, 부메뉴 영역에서의 그래픽 요소는 판넬(panel) 및 버튼이 사용된다. 각각의 주메뉴 단계에 대한 부메뉴 객체 집단을 구성하며 집단화된 객체는 단일 요소와 같은 역할을 하게 된다.

3. 전처리기의 설계

3.1 전처리기의 구조

전처리 단계에서는 Fig. 3.1과 같이 절점좌표(nodal coordinates), 요소연결(element connectivity), 경계조건(boundary condition), 작용하중(applied load), 재료 특성치(material properties)등과 같은 새로운 데이터를 생성하는 과정과 기존의 데이터를 읽는 과정으로 구성되어 있고, 확인 및 수정 과정을 포함한다.

전처리기의 구동은 주메뉴 영역에 있는 **Preprocessor** 버튼의 응답으로 생성되는 부메뉴들의 세부적인 입력 단계를 거치면서 이루어진다. 생성되는 부메뉴의 종류는 Fig. 3.1과 같이 나타낼 수 있다. 이 때 부메뉴의 속성 변경을 이용하여 동일한 객체에 대해 다양한 조건에서의 응답을 유도할 수 있다. 객체의 사건처리 과정은 미리 정의된 사건 프로시져 가운데 코딩이 되지않은 사건 프로시져는 응답하지 않으며 정의된 사건에만 응답하도록 설계된다. 전처리기의 부메뉴들은 동일한 그래픽 객체의 이름 속성(name attribute)을 갖는 동적 객체 배열(dynamic object array)로 구성하여 동적 특성을 갖는다.

3.2 자동 스케일 변환

전처리기에서 자료를 입력하여 모델링 할 경우, 모델이 그래픽 영역의 범위 내에서 값의 크기와 상관없이 적당한 크기로 위치해야 할 필요성이 있다. 이것은 자동 스케일 기능으로 그래픽 표현 기능을 갖는 시스템에서 기본적인 요소라 할 수 있다. 자동 스케일 방법은 절점 좌표의 최대값과 최소값을 기준으로 그래픽 영역에 모델을 적당한 크기로 위치시킨다. 전처리기에서 자동 스케일 과정은 Fig. 3.2와 같이 나타낼 수 있다.

최초 정의된 스케일에 맞게 좌표의 입력이 끝나면 입력된 좌표값에서 X, Y의 최대값과 최소값을 구하고 그 값에 상·하, 좌·우의 여유 공간을 더한 값으로 스케일을 정의하게 된다. 최초 스케일에서 100 이상의 좌표값을 입력하면 그래픽 영역을 벗어나 화면상에는 나타나지 않는다. 이러한 사용자 좌표계는 후처리기의 변형된 형상을 나타낼 때까지 사용되나 응력분포를 표현할 경우는 적용이 되지 않는다. 그것은 API 함수가 자료를 처리하는 과정에서 그래픽 표현영역에 대한 속성을 화면 좌표계로 전환시키기 때문이다. 좌표계에 관한 언급은 4절에서 자세히 다루었다.

Fig 3.2의 흐름도는 시스템 초기 구동시 정해진 스케일에 맞게 절점 좌표를 입력 받게되고, 프로그램은 입력된 절점 값에서 X, Y방향으로 최대값과 최소값을 구하게 된다. 자동 스케일 변환 함수를 이용하여 최초 입력된 절점 좌표를 화면에서 지우고 다시 정의된 스케일에 맞게 전체적인 절점 좌표를 화면상에 나타나는 것을 표현하고 있다.

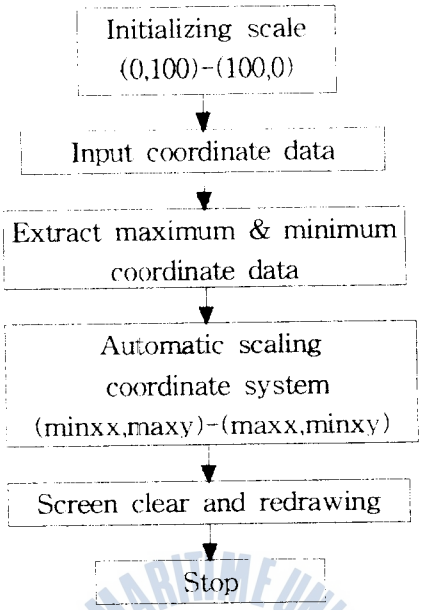

Fig. 3.2 Automatic scaling process diagram at Preprocessor

3.3 객체 배열 및 속성 변형

부메뉴의 경우 같은 종류의 객체들을 중복해서 사용해야 할 경우 객체 이름 및 개발자가 필요로 하는 속성을 정의하고, **index**를 첨자로 하여 배열화하는 객체 배열을 적용한다. 객체 배열을 사용하면 **index** 만큼의 객체를 생성할 수 있으며 배열된 객체의 수와 관계없이 하나의 사건 프로시저를 공유하게 되므로 코드 작성 및 프로시저 설계를 단순하게 할 수 있다. Table 3.1은 자동 요소분할 과정을 포함한 전처리 단계의 부메뉴의 구조를 나타낸다.

Table 3.1 Sub menu structure of Preprocessor using object array

object	name	procedure	index	caption	code
Panel	prep()	prep_Click()	1	1-D	Select Case Index+1 Case 1 : End Select For i=1 to 2 (or 3) prep(i).Caption =meshCap(i) Next i * Meshcap * - Meshcap(1)="1-D" - Meshcap(2)="2-D" - Meshcap(3)="3-D" :
			2	2-D	
			3	3-D	
			1	Triangle	
			2	Quadrilat	
			1	KeyPoints	
			2	MidPoints	
			3	PLOT	
			1	BOUN	
			2	FORC	
			3	MATE	
			4	SAVE	

Table 3.1에서와 같이 최대 첨자 값은 4가 되며 객체를 4개 제작한 것과 같은 효과를 갖는다. 일반적인 방법으로 객체를 4개 제작했을 경우 마우스 클릭 사건 프로시져만 작성한다 할지라도 4개의 프로시져를 작성해야만 한다. 그러나 배열을 사용할 경우 외형적인 모습은 4개의 객체를 제작한 것과 같으나 실질적으로 사용되는 프로시져는 각 사건에 대해 1개만을 코딩하면 된다. 그러나 1개의 프로시져로써 4개의 형상을 갖는 객체를 제어할 수는 없다. 따라서 속성변형을 이용하여 각각의 속성에 대한 조건식으로 4개의 효과를 얻어야만 한다. 속성변형을 이용하여 프로그램 처리되는 과정은 Table 3.1의 **code**란에 기술된 것과 같다. .

전처리기의 부메뉴에서 입력되는 내용은 그래픽으로 동시에 표현된다. 동일한 **index**를 갖는 객체의 응답에 있어서는 객체의 속성을 변화시킴으로써 사건 프로시져를 공유할 수 있도록 설계할 수 있다. 전처리기에서 자동요소분할 과정에서 나타나는 부메뉴 구조를 **index**의 조건에 의해 나타내면 Table 3.2와 같다.

Table 3.2 Submenu structure using object array by index and caption property

index	**1**	**2**	**3**	**4**	**5**	**6**
caption	1-D(a)	2-D(a)	3-D(a)	SAVE(u)		
	KeyPoint(a)	MidPoint(a)	PLOT(a)			
	BOUN(a)	FORC(a)	MATE(a)			
	COOR(u)	ELEM(u)	BOUN(u)	FORC(u)	MATE(u)	SAVE(u)

(a)로 표시된 영역은 자동요소분할 기능을 나타내고 (u)로 표시된 영역은 사용자 정의 요소분할 기능을 실행했을 경우다. 각각의 영역에 나타난 문자(string)는 객체의 **caption** 속성을 나타낸다. 즉, 하나의 객체에 해당되는 **index**조건은 최대 5개까지 포함되지만 **EDIT** 와 **VIEW** 기능을 포함한다면 7개까지 가능하다.

객체 배열을 사용할 경우 부메뉴에서 현재의 입력 단계에서 불필요한 입력 메뉴를 나타내지 않음으로써 입력 과정에서 발생할 수 있는 입력 오류(error)를 줄일 수 있다. 입력 귀환(input return)도 마찬가지의 개념으로 설계된다. 개발 프로그램의 전처리기에서의 입력 귀환은 세부적인 단계의 데이터 입력에서 오류를 범하였을 경우 사용자가 원하는 단계부터 입력이 다시 이루어지게 할 수 있는 기능을 말한다. 이러한 기능은 구조해석 프로그램뿐만 아니라 다른 일반적인 자료 입력을 필요로 하는 시스템 설계시에도 필요한 항목이다. 특히 입력 자료의 양이 많은 구조해석 프로그램의 경우, 전처리기에서의 입력 귀환 메뉴는 중요하다. 전처리기에서의 세부적 단계와 입력 귀환 메뉴는 Table 3.3과 같이 구성하였다.

Table 3.3 Input return menu at Preprocessor

	Input return menu			
	return-0	return-1	return-2	return-3
sub menu	AutoMesh	1-D	Elem-Type	BOUN
		2-D		MATE
		3-D		FORC
	UserMesh	COOR, ELEM, MATE, FORC, BOUN		

Table 3.3과 같이 입력 메뉴 설계에 있어서 각 세부 단계를 표현할 수 있는 메뉴를 하나씩만 추가하면 된다. 위의 테이블에서 추가된 메뉴는 **return=0,1,2,3**이다. 추가된 귀환 메뉴는 단순히 귀환된 범위 내의 객체에 대한 **caption** 속성만을 표현하면 된다. **caption** 속성을 변형하였을 경우 앞서 설명된 속성 변형에 따른 조건식의 실행으로 해당 단계에서부터 입력을 할 수 있다. 따라서 귀환되는 범위는 바로 전 단계일 수도 있으며 무조건적 귀환으로 최초, 또는 사용자가 원하는 단계에서부터 언제든지 입력을 실행할 수 있다.

3.4 자동 요소 분할 과정

개발 시스템에 적용된 자동 요소 분할 과정[12,13]은 Fig. 3.3과 같다. 원의 형상을 갖는 요소를 분할할 경우에는 **MidPoint**의 과정을 필요로 하지만 그렇지 않은 경우는 몇 개의 **KeyPoint**만을 키보드나 마우스를 이용하여 입력하면 된다. **Stage 1**과 **Stage 2**는 각각의 객체로 구성된 독립적인 동작 과정을 나타내고**Stage 3**과 **Stage 4**는 연속적으로 처리되는 루틴(routine)이다.

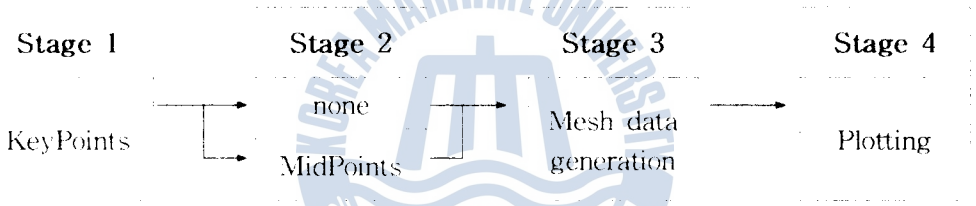

Fig. 3.3 Automatic mesh generation process at **Preprocessor**

KeyPoint는 모델을 구성하는 각 점(point)을 사용자가 분할하고자 하는 의도에 맞게 상세적으로 입력한 것이다. 따라서 **KeyPoint**의 입력만으로 모델의 형상에 대한 대략적인 윤곽이 나타나게 된다. **MidPoint**는 **KeyPoint**로 정의된 모델에 세부적인 사항을 점으로 추가함으로써 실질적인 모델링 형상을 표현하는 것으로 설명할 수 있다.

자동요소분할 방법은 해석하고자 하는 문제의 형상을 전체적으로 4 개의 구석점(coner point 이하 **cp**)으로 구성된 사각형(eight node quadrilateral) 요소로 나누고 그것을 완전한 사각형의 형상을 갖는 블록(block)으로 맵핑(mapping)하여 생각한다. 본 논문에서는 Fig. 3.6와 같이 중앙에 구멍(hole)이 있는 평판에 인장력이 작용하고 있는 평면응력(plane stress) 문제를 해석하기 위하여, 대칭 조건을 고려한 1/4 자동 요소 분할 예를 이용하여 전체적인 시스템의 자료처리 과정을 표현하고자 한다. 실질적인 해석 모델은 **Area 4** 영역이다.

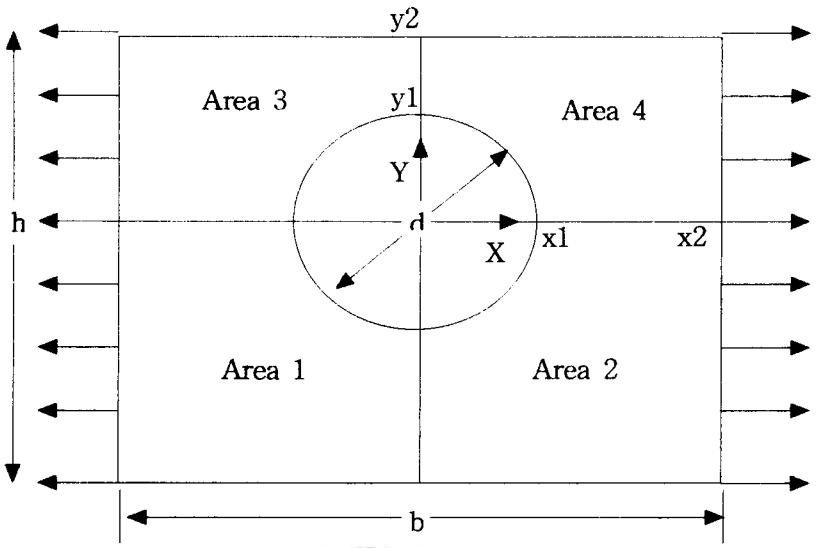

Fig. 3.5 Example model

해석 모델의 자동요소분할 방법은 Fig. 3.6와 같이 모델을 직사각형 요소로 맵핑을 하게된다. 이 때 각 블록은 점선이 지나는 점을 포함하여 8개의 **cp**로 구성된다. 각각의 블록을 구성하는 **cp**들은 **KeyPoint**만을 이용하여 요소 분할을 할 경우, **RBLOCK 1**만을 예를 들면 N1→N2→N5→N4으로 구성된다. 그리고 X1→Y2→X3→Y1은 **MidPoint**로써 자동적으로 4개의 요소로 분할하는 역할을 하게된다.

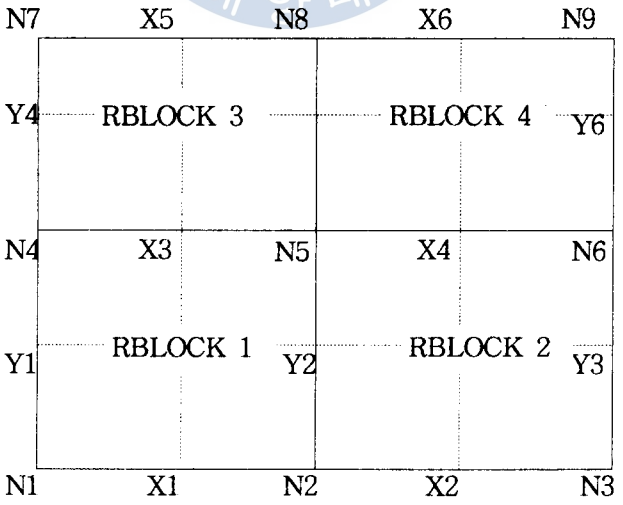

Fig. 3.6 Block diagram of rectangular mapping model

실질적인 해석 모델을 분할한 형상은 Fig. 3.7과 같이 나타낼 수 있다. 전체적인 요소를 분할하는 경우, **BLOCK 1**은 N1→N2→N5→N4으로 구성됨을 알 수 있다. 이와 같은 분할은 호의 형상을 갖는 요소로 Y1과 같이 **MidPoint**를 필요로 한다.

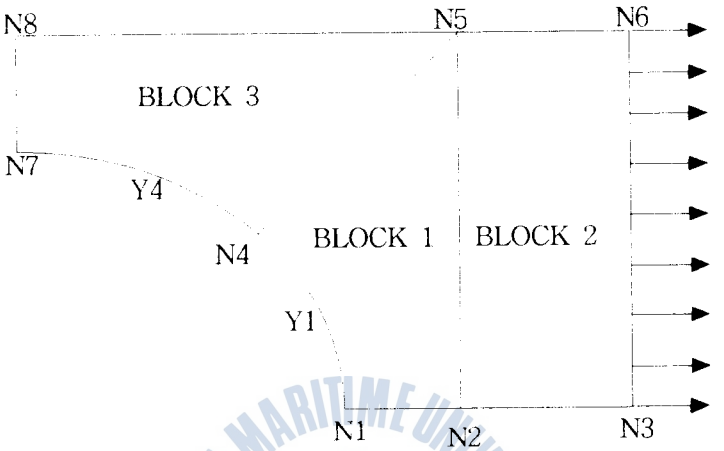

Fig. 3.7 Divided region of analysis model

맵핑하는 방법은 해석 모델의 분할된 영역을 구성하는 **cp**와 맵핑 모델을 구성하는 **cp**의 번호가 같아야만 한다. 이러한 **cp** 구성은 맵핑 모델에서도 일치해야만 한다. 이 때 맵핑 모델을 구성하는 **cp**와 해석 모델을 구성하는 **cp**의 수가 다르다는 것을 알 수 있다. 다른 **cp**는 실제 해석 모델을 구성하지 않는 N9 **cp**이다. 그리고 N9이 속해 있는 블록은 해석 모델에서는 존재하지 않고 맵핑 모델에서만 존재한다는 것도 알 수가 있다. N9이 속해 있는 블록은 맵핑 모델에서의 **RBLOCK 4** 이다. **RBLOCK 4**는 실제 모델링에서는 사용되지 않는 블록으로 **Void Block**으로 간주한다. 따라서 자동요소분할을 위한 입력 과정에서 실제 존재하지 않는 블록의 번호를 입력하는 단계가 포함된다. 자동요소분할을 이용한 경계조건, 하중, 재료 특성치등 전처리기의 모든 입력과정을 마친 화면은 Fig 3.8과 같다.

4. 후처리기의 설계

후처리 단계는 해석 단계에서 얻어진 변위값을 이용하여 응력값을 구하고 그래픽 및 텍스트로 표현한다. 그리고 표현된 내용을 프린터(printer)를 이용하여 출력하는 과정을 포함한다. 따라서 외부 장치와 연결되는 프로그램 코드와 API 함수 호출, DLL 연결등 복잡한 요소로써 설계된다. 후처리 단계의 전체적인 부메뉴 구성은 Fig. 4.1과 같이 크게 4부분으로 나뉘어지고 각 부분들은 전처리기와 같은 방법으로 그림상자와 글상자를 중심으로 상호 연결성을 갖고 결과를 표현한다.

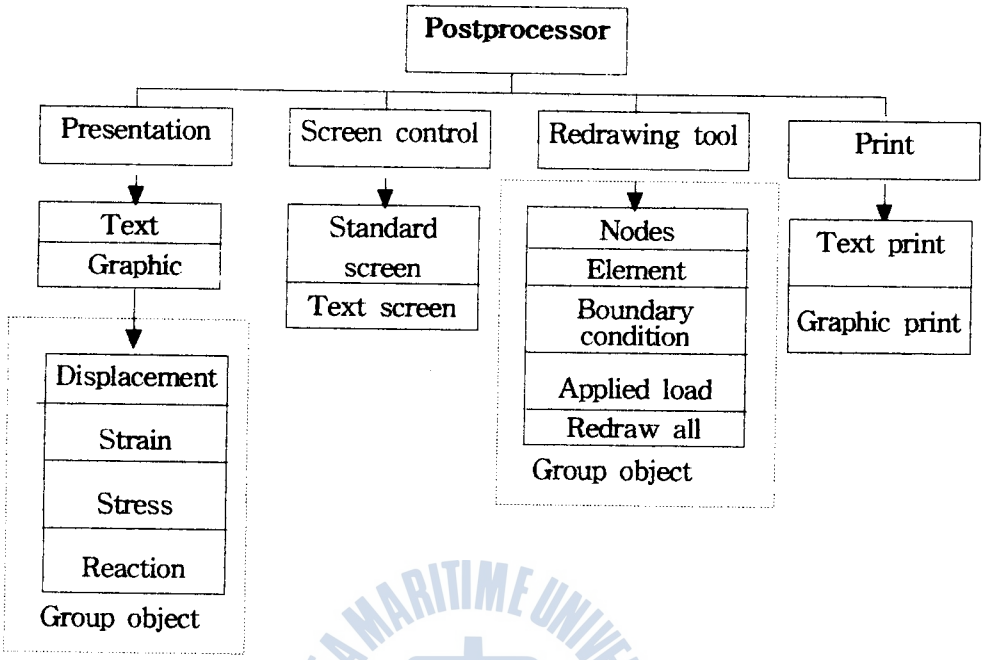

Fig. 4.1 Sub menu structure of **Postprocessor**

4.3 변형된 형상의 그래픽 표현

후처리기에서 변형된 결과를 표현하는 방법은 아래와 같이 단순한 수식에 의해 이루어진다. 변위값은 해석 단계에서 파일에 기록한 값들을 그대로 사용한다. 아래의 코드와 같이 각 절점에 대해 파일에 기록된 변위는 후처리기에서 그래픽 표현을 위해 disp_X()와 disp_Y()의 배열 변수에 값을 기록한다. 기록되는 값은 변위값과 최초의 절점 좌표를 합한 값이다. 그래픽 영역의 좌표계는 사용자 좌표계를 이용한다.

```
For i = 1 to numnp
    disp_X(i)=dvX(i)+X(1,i)
    disp_Y(i)=dvY(i)+X(2,i)
Next i
```

여기서,

numnp : number of node
dvX(i), dvY(i) : X, Y 변위
X(1, i) : X 좌표, X(2, i) : Y 좌표
disp_X(i) : 변형된 X 절점 좌표 ,disp_Y(i) : 변형된 Y 절점 좌표

최종적으로 그래픽으로 표현되는 값은 disp_X(i)와 disp_Y(i)값이 된다. 첨자 i 는 1부

더 선체 절점 수까지의 배열을 나타낸다. 변형된 형상에 대한 스케일링은 아래의 식과 같이 disp_X(i)와 disp_Y(i)에 대하여 스케일링 값을 곱하면 된다.

Scale_disp_X(numnp)=disp_X(numnp)×(Scale factor)
Scale_disp_Y(numnp)=disp_Y(numnp)×(Scale factor)

Scale factor는 X와 Y에 대해 동일한 비율로 정의되어야만 한다. **Scale factor**의 입력은 개발 프로그램에서 입력 상자를 이용하도록 설계 되었다. 변형된 형상을 그래픽으로 표현한 화면은 Fig. 4.2이다.

4.2 API를 이용한 응력 분포의 그래픽 표현

후처리 기능에 있어서 윈도우상에서 그래픽 효과를 얻기 위해서는 윈도우즈 GDI 객체를 사용한다. 후처리 단계에서 각각의 요소에 대한 결과치를 색표현하기 위하여 개발 시스템에서는 API를 이용하였다. API 함수의 호출은 VB에서 아래의 코드와 같은 방식으로 호출하여 사용된다. 호출되는 함수는 다각형(polygon)의 형상을 구성하는 함수와 내부의 색을 결정하는 함수로 구성된다. API 함수선언은 후처리 단계를 구성하는 모듈의 일반 선언부(general declaration section)에서 선언된다[16]. API 함수로 전달되는 매개 변수의 특성은 Table 4.1와 같이 나타낼 수 있다.

Declare Sub Polygon Lib "GDI" (ByVal hDC%, lpPoints as **PointAPI**, ByVal nCount)
Declare Sub SetPolyFillMode Lib "GDI" (ByVal hDC%, ByVal nPolyFillMode%)

Table 4.1 Application Programmer's Interface function properties

Parameter	Type	Description
lpPoints	PointAPI	The first PointAPI structure in an array of nCount PointAPI structures
nCount	Integer	Number of points in the polygon
hDC	Integer	Handle to a device context
nPolyFillMode	Integer	Alternates filling or Fill based on drawing direction

4.3 응력 분포

후처리기에서 응력분포를 색으로 표현하기 위한 스케일 변화를 포함하는 전체적인 흐름을 Fig. 4.3과 같이 나타낼 수 있다. 응력 분포를 색 표현하고자 할 경우 **Stress()**의 클릭 사건으로부터 시작된다. **Stress()**는 각 방향의 응력을 표현할 때 사용되는 패널 객체를 배열화 시킨 것이다. **index**가 0일 경우 X방향 응력을 나타내고, 1일 경우 Y방향, 2일 경우 전단 응력을 표현한다. **Stress()** 클릭 사건이 발생함과 동시에 실행 코드는

PointAPI 구조체로 정의된 PT() 변수에 좌표값을 전달하기 위해 파일을 열게된다. 이때 열리는 파일은 한줄에 8개의 좌표가 기록된 파일이다. 8개의 좌표는 4절점의 X좌표와 Y 표이다. 4절점은 하나의 요소를 이루는 4개의 점을 의미한다.

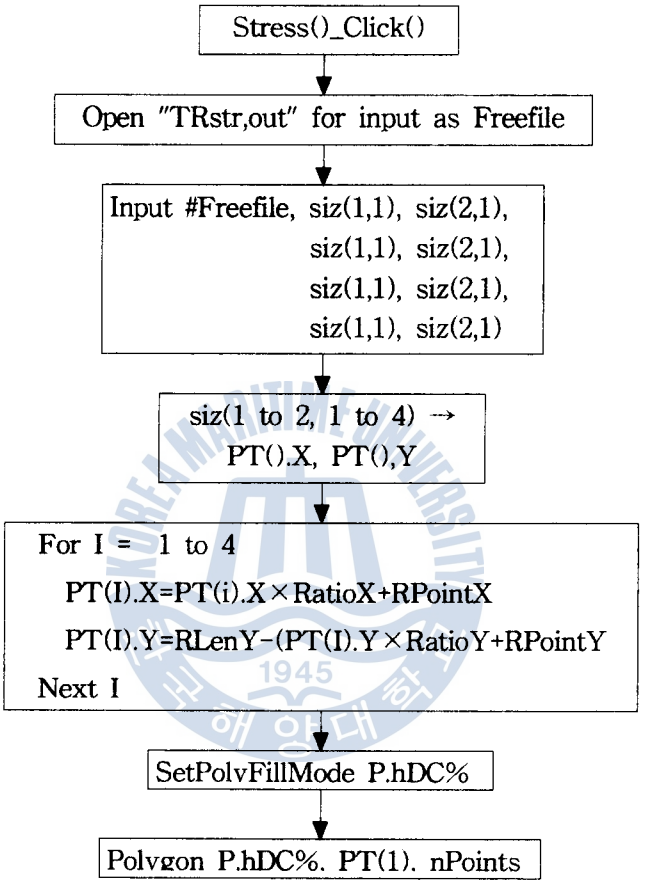

Fig. 4.3 Stress presentation process using color value

예제로 다루고자 하는 문제를 4절점 사각형 요소로 분할하여 해석하고자 하기 때문에 4절점은 하나의 요소를 이루게 된다. PT() 구조체로 값을 전달하기 전에 파일에서 읽어 들이는 값은 새로운 변수인 siz()에 할당된다. 할당되는 변수의 양식은 siz(1 to 2, 1 to 4)의 형태로 할당된다. 여기서 배열 선언부에 앞의 부분인 1 to 2는 좌표의 방향을 의미하는 것으로 1일 경우 X축 좌표값을, 2일 경우 Y축 좌표값을 나타낸다. 1 to 4의 배열 선언부는 하나의 요소를 구성하는 절점 번호를 나타낸다. 구조체로 전환된 값은 사용자 좌표계로 나타낼 때는 아무런 문제가 없지만 API 함수에서 실제 적용되는 좌표값은 화면 좌표계이기 때문에 값을 다시금 수정해야만 한다. 이때 수정되는 코드는 다음과 같이 나타낼 수 있다.

```
For i = 1 to 4
    PT(i).X=PT(i).X × RatioX+RPointX
    PT(i).Y=RLenY-(PT(i).Y × RatioY+RPointY)
Next i
```

RatioX, RatioY : X, Y축 방향의 여유
RPointX, RPointY : X, Y축 방향의 사용자 좌표계와 화면 좌표계의 비
RLenY : 화면 좌표계의 Y축 최대값

화면 좌표계로 나타낼 모든 값이 전환되면 그래픽 표현 영역을 나타내는 객체에 대한 핸들을 통해 다각형을 구성하는 함수에 대해 다각형 내부의 색이 결정될 수 있도록 **SetPolyFillMode()** 함수를 사용한다. 최종적으로 다각형 형상을 구성하는 **Polygon()** 함수로 색표현을 하게 된다. Figs. 4.4는 전단응력 분포를 나타낸 것이다.

4.4 OLE를 이용한 컴포넌트 시스템 구축

OLE는 윈도우즈 응용 프로그램들 사이의 자료교환 및 제어를 자연스럽게 이루어지게 하는 특징이 있다. 또한 프로그램 개발에 있어 데이터 베이스(data base)나 그래픽 처리를 위해 별도의 프로세서(processor)의 개발없이 기존의 개발된 동일한 기능을 갖는 프로그램을 개발 프로그램에 그대로 접속하여 사용할 수 있다. 따라서 프로그램의 개발 시간을 현저히 줄일 수 있을 뿐만 아니라 전문적인 기능을 갖는 독립 프로그램을 그대로 사용함으로써 개발 프로그램의 신뢰성을 높일 수 있다[17].

후처리기에서 각 요소에 대한 응력 분포를 2차원 그래프로 나타내는 과정은OLE를 이용하여 간단하게 이루어 진다. 본 개발 프로그램의 예에서는 선 그래프를 선택하였다. 자료가 전달되어 그래프로 나타나는 과정을 도식하면 Fig. 4.5와 같이 표현할 수 있다.

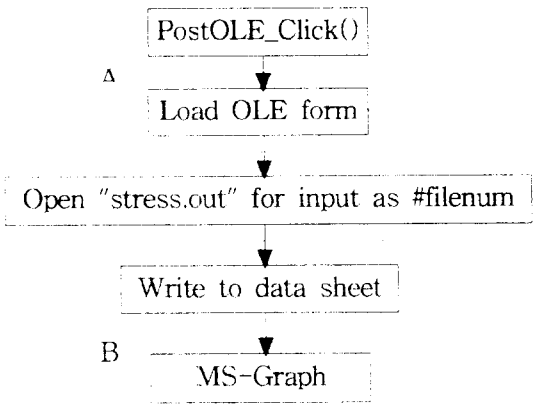

Fig. 4.5 Graph presentation diagram using OLE

Fig. 4.5에서 주메뉴 영역에 위치한 **Post-OLE** 메뉴를 클릭하게 되면 새로운 폼이 생성되고 그 폼에는 미리 정의된 OLE 객체를 포함하고 있다. OLE객체는 자신이 포함된 폼이 생성될 때 Form_Load() 프로시져를 구동하게 되고, 이 프로시져 내에는 응력 결과가 기록된 파일을 열어 값을 읽는 코드가 기록되어 있다. [A] 단계는 이러한 과정을 나타낸다. [B] 단계의 경우 읽은 파일을 DDE를 이용하여 **MS Graph**에 내장된 데이터 시트로 자료를 전송하게 된다. 전송된 자료는 개발 시스템에서 미리 정의한 선 그래프 나타나게 된다. Fig. 4.6는 OLE 기능을 이용하여 요소에 대한 각 방향의 응력을 2차원 그래프로 나타낸 화면이다. Fig. 4.7는 각 가우스 점에 대한 응력 결과를 텍스트로 표현한 것이다.

5. 결 론

본 연구에서는 실질적인 프로그램 개발을 통하여 VB을 이용하여 윈도우즈용 구조해석 프로그램을 효율적으로 제작하는 방법을 제시하였다. 시스템 전체적으로 윈도우즈 응용 프로그램이 필수적으로 갖게 되는 GUI 구현은 VB가 지원하는 그래픽 객체로써 쉽게 구현할 수 있었고, 미리 정의된 사건 프로시져를 작성함으로써 운영체계에 적합한 신뢰성 있는 프로그램을 설계할 수 있었다. 또한 윈도우즈 API 함수를 사용함으로써 임의 요소형상의 그래픽 표현과정에서 발생되는 문제점을 해결하였다. 또한 VB 기능을 확대하여 구조해석 프로그램 설계시에 발생되는 VB의 제한적 기능들을 충분히 확장시킬 수 있음을 알 수 있었다.

윈도우즈가 제공하는 DDE 및 OLE기능을 이용한 프로그램 설계는 프로그램의 컴포넌트 설계를 용이하게 할 수 있어서 프로그램의 설계시간 단축 및 확장성을 얻을 수 있었다. 그러나 기존의 도구가 제공하는 기능들 외에 본 개발 프로그램은 전·후 처리기를 완성한 상태에서 요소라이브러리만의 추가로써 해석 범위를 확대할 수 있게 설계한 것이 큰 장점이라 할 수 있다.

추후 지속적인 연구과제로서 비선형 정적해석과 동적해석도 가능하게 해석 범위를 확장시키고, 다양한 유한요소도 개발하고자 한다. 전·후처리에서도 회전 기능을 포함하는 3차원 모델링도 가능하게 할 것이다.

참고 문헌

[1] Microsoft, **Window 3.1 User Guide**, 1993.

[2] 신영식, 서진국, 송준엽, "유한요소 구조해석을 위한 전후처리 통합운영 시스템에 관한 연구", 한국전산구조공학회지, Vol. 8, No. 1, pp. 161-172, 1995

[3] Clayton, W., **Borland C++ for Windows Programming**, 2Ed., Prentice -Hall, 1994.

[4] Leestma, M., **Object-Oriented Programming Turbopascal**, Merril- Macmillan, 1993.

[5] Namir Clement Shamas, **Windows Programmer's Guide to Object Windows Library**, SAMS, 1992

[6] James Martin, **Principles of Object-Oriented Analysis and Design**, Prentice-Hall, 1993.

[7] David I., **An Introduction to Programming Using Visual Basic**, Prentice-Hall, 1995

[8] James L. Conger, **Windows API Bible**, SAMS, 1994

[9] Geary Entsminger, **The Secrete of Visual Basic 3.0 for Windows**, SAMS, 1994

[11] 한국전산구조공학회 기술강습회, Cad의 활용 및 프로그래밍 기법, 제 12회, 1995

[12] Zienkiewicz, O. C. and Philips, D. V., "An automatic mesh generation scheme for plane and curved surfaces by isoparametric coordinates", International Journal for Numerical Engineering, Vol. 3, pp. 519-528, 1971.

[13] P.L. George, **Automatic Mesh Generation Application to Finite Element Method**, WILEY, 1991

[14] 홍정부, "Windows Library", 시사 컴퓨터, pp. 92-99, 7월 1994

[15] Geary Entsminger, **Visual Basic 3.0 Master**, SAMS, 1994

[16] Ziff-Davis, 1993.Daniel Appleman, **Visual Basic Programmer Guide to the Windows API**,

[17] Richard Mansfield, **Visual Basic Power Tool Kit**, Ventana Press, 1995

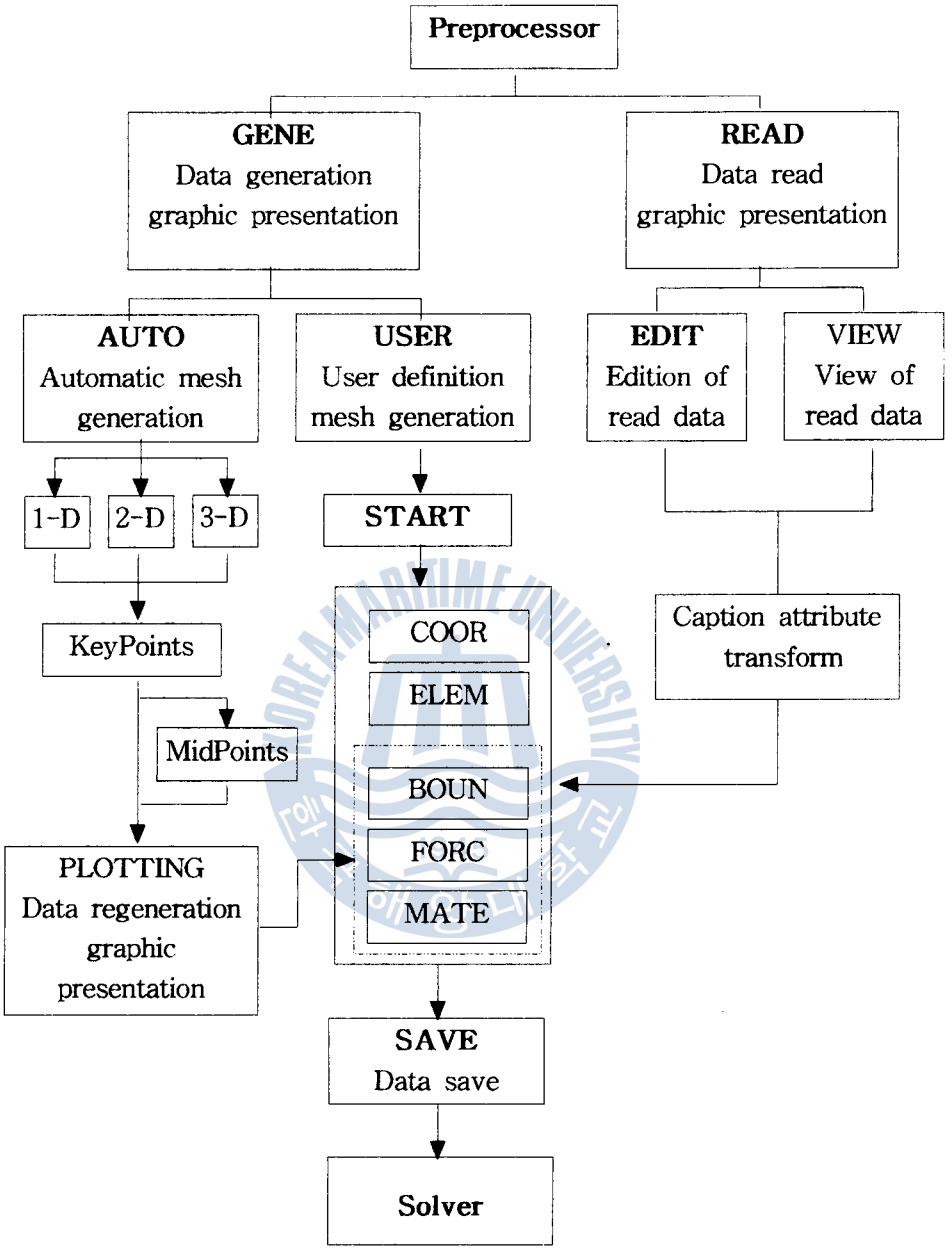

Fig. 3.1 Overall structure of integrated process system at Preprocessor

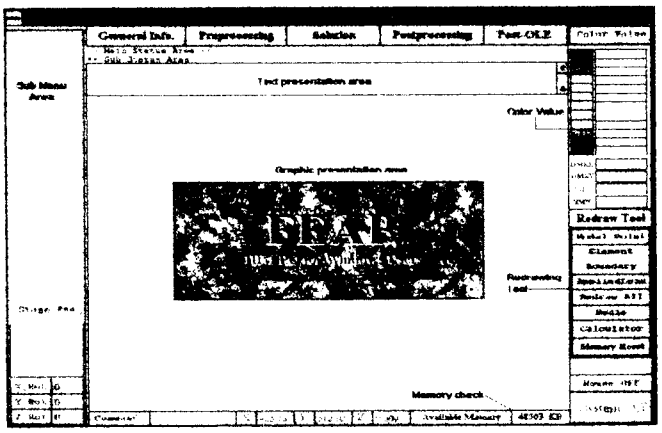

Fig. 2.2 Overall structure of integrated process system

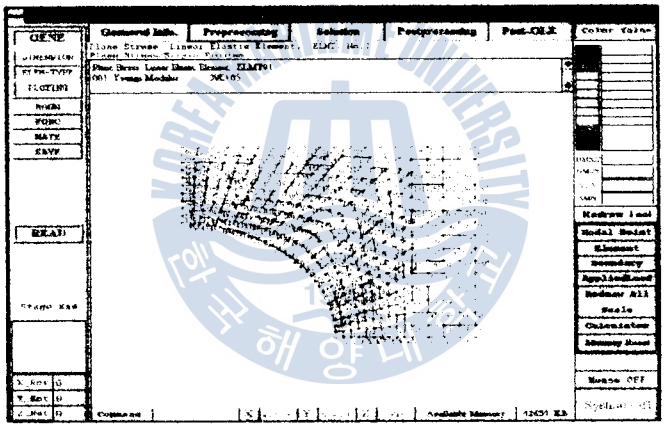

Fig. 3.12 Analysis model using automatic mesh generation at Postprocessor

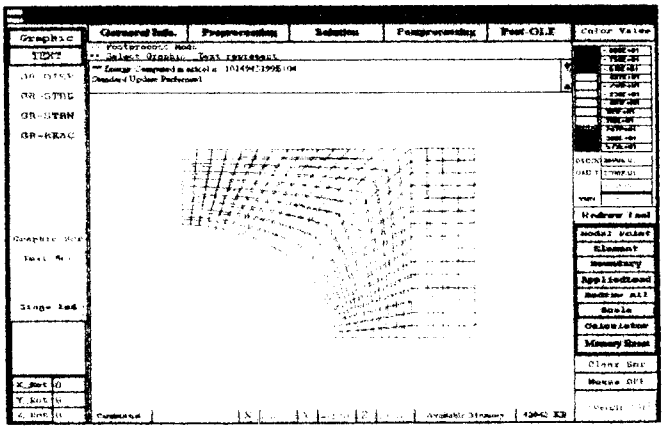

Fig. 4.2 Deformation configuration at Postprocessor

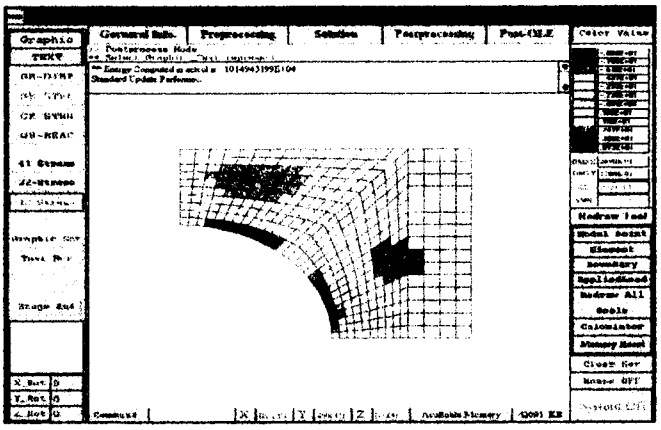

Fig. 4.4 Shear stress distribution at Postprocessor

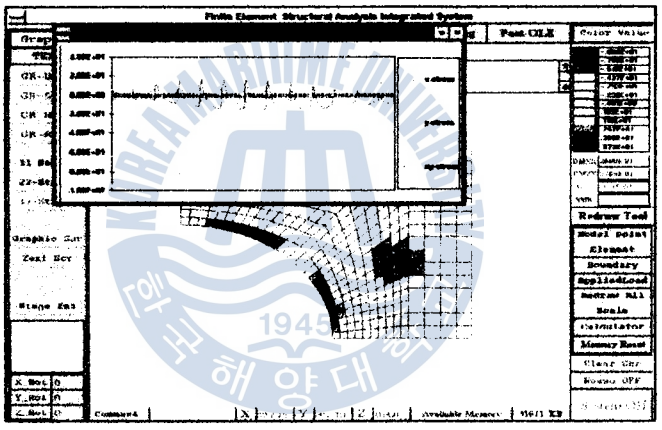

Fig. 4.6 Two dimension graph presentation using OLE

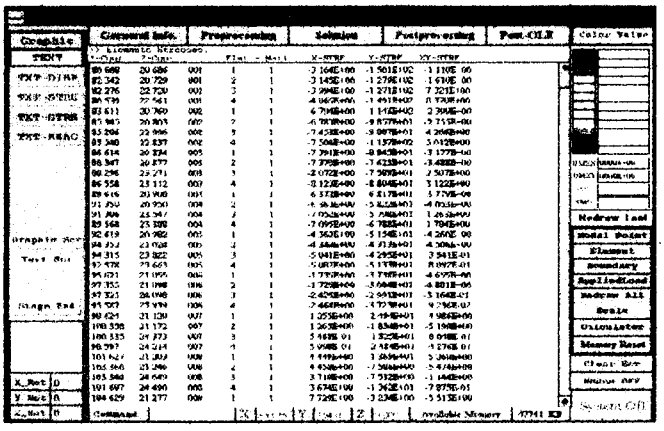

Fig. 4.7 Text result presentation for Gauss Points at Postprocessor