Visual Basic을 이용한 구조해석 프로그램 개발에 관한 연구

이 상 갑* · 장 승 조**

A study on the Development of Structural Analysis Program using Visual Basic

*Sang-Gab Lee** · *Seung-Cho Chang***

<目 次>

요 약

본 논문에서는 윈도우즈용 응용 프로그램 개발 도구로서 최근에 각광을 받고 있는 VB(Visual Basic)을 사용하여 유한요소 구조해석 프로그램을 개발하고자 한다. VB는 사건 중심적 프로그래밍 방법과 컨트롤 형태의 그래픽 객체등을 지원함으로써 응용 프로그램을 쉽게 개발할 수 있는 여러가지 기능들을 제공한다. 본 개발 프로그램은 전처리, 해석 및 후처리 기능을 포함하는 통합 환경 시스템이다. 전처리기에서는 자동 요소 분할이 가능하고, 후처리기에서는 변형 및 응력분포등의 그래픽 표현도 나타낼 수 있다.

Abstract

The objective of this paper is to develop a finite element structural analysis program using VB(Visual Basic) which has been recently getting popular as development tools of application

* 한국해양대학교 조선공학과 조교수
** 한국해양대학교 조선공학과 석사과정

program for Windows. VB provides several functions to develop an application program easily by supporting event - driven programming method and graphic object as control data type. This system is an integrated processor including preprocessor, solver and postprocessor. Automatic mesh generation is available at preprocess stage, and graphic presentation of deformation and stress is also represented at postprocess one.

Ⅰ. 서 론

최근 PC(Personal Computer)의 성능향상 및 운영체계(Operating System)의 발전은 그동안 PC의 단점으로 여기던 처리속도와 GUI(Graphic User Interface), 다중처리(Multi - Tasking), 네트워킹(Net - Working) 등의 구현으로 응용범위가 크게 향상되었다. 특히 마이크로 소프트(Micro Soft)사에 의해 개발된 윈도우즈(Windows)[1]의 출현은 이러한 운영체계의 발전을 대표하는 것으로 볼 수 있다. 운영체계의 발전은 곧 그 운영체계에 맞는 응용 프로그램(Application Program)의 개발을 서두르게 하고 있으며, 프로그램 개발과 관련하여 이미 선진국에서는 윈도우즈 응용 프로그램 개발에만 집중하는 경향이 나타나고 있다. 운영체계의 발전은 사용자 입장에서는 편리성을 제공하지만 프로그램 개발자(Programmer)의 경우에는 프로그램 개발을 더욱 어렵게 하여 다양한 분야에서 요구되어지는 응용 프로그램 개발을 더디게 하는 결과를 낳고 있다.

그러나 윈도우즈 응용 프로그램을 개발하는데 사용되는 「Borland C++」[2]나 「Turbo Pascal for Windows」[3]와 같은 개발 도구(Development Tool)의 발전은 이러한 문제점들을 크게 개선하여 좀 더 쉽게 응용 프로그램을 개발할 수 있게 하였다. VB도 윈도우즈용 응용 프로그램 개발 도구로써 손쉽게 프로그램을 개발할 수 있는 기능을 제공하고 있다. 본질적인 의미의 객체지향(Object - Oriented)[4]을 지원하지는 않지만 컨트롤 형태(Control Type)의 그래픽 객체(Graphic Object)를 이용하여 개개의 요소로써 프로그램을 조립하듯 제작할 수 있다. VB에 있는 그래픽 객체들은 프로그램에서 필요로 하는 모든 객체와 그 객체가 반응하는 사건(Event)들을 이미 갖추고 있으며 프로그램 인터페이스(Interface) 구성에 필요한 요소들도 거의 갖추고 있다[5]. 이러한 기능들은 윈도우즈 운영이 내 · 외부에서 발생하는 메시지와 사건에 응답하는 방식으로 동작되는 경우와 동일하여 프로그램의 신뢰성을 높일 수 있다. 또한 윈도우즈 API(Application Programmer's Interface) 함수의 이용과 다중처리를 위하여 다른 언어로 제작된 동적 연결 라이브러리(DLL : Dynamic Link Library)와의 접속은 VB의 기능을 크게 확장시킬 수 있고 실질적으로 매우 쉽게 이루어 진다. 그 밖에도 DDE(Dynamic Data Exchange)와 OLE(Object Likning Embedding)를 이용하여 다른 프로그램과의 자료 교환 및 제어가 가능하다[6].

본 연구에서는 VB를 이용하여 공학용 프로그램 설계를 쉽게 구현하는 방법을 제시하고 구조해석용 유한요소 프로그램을 개발하고자 한다. 시스템 구성은 전처리, 해석, 후처리의 3단계로 이루어지는 통합운영 시스템(Integrated Process System)이다. 전 · 후 처리(Pre/Post Process)에 있어서는 자동요소분할생성(Automatic Mesh Generation)을 통한 그래픽 모델링(Graphic Modeling)과 응력분포(Stress Distribution), 변위(Displacement) 등과 같은 결과를 그래픽으로 표현할 수 있는 기능들을 포함하고 있다. 프로

그램은 전반적으로 컨트롤 배열(Control Array), 동적배열(Dynamic Array)과 같은 동적 요소들을 사용하여 메모리(Memory)의 사용 범위를 넓히고 동적 프로그램의 특성을 갖게끔 설계하였다. 본 논문에서는 전처리기의 경우 메뉴구성과 사건 중심적 프로그래밍에 대해 기술하고 후처리기에서는 DLL과 API, OLE 기능을 이용하여 프로그램을 설계하는 방법에 대해 기술하였다.

Ⅱ. 시스템 설계 및 개요

본 유한요소 구조해석 시스템은 전처리, 해석, 후처리의 3단계로서 구성되어 있고 각 단계에서의 부메뉴를 포함하여 세부적인 처리과정을 갖는다. 시스템의 전체적인 구조는 Fig. 2.1과 같다.

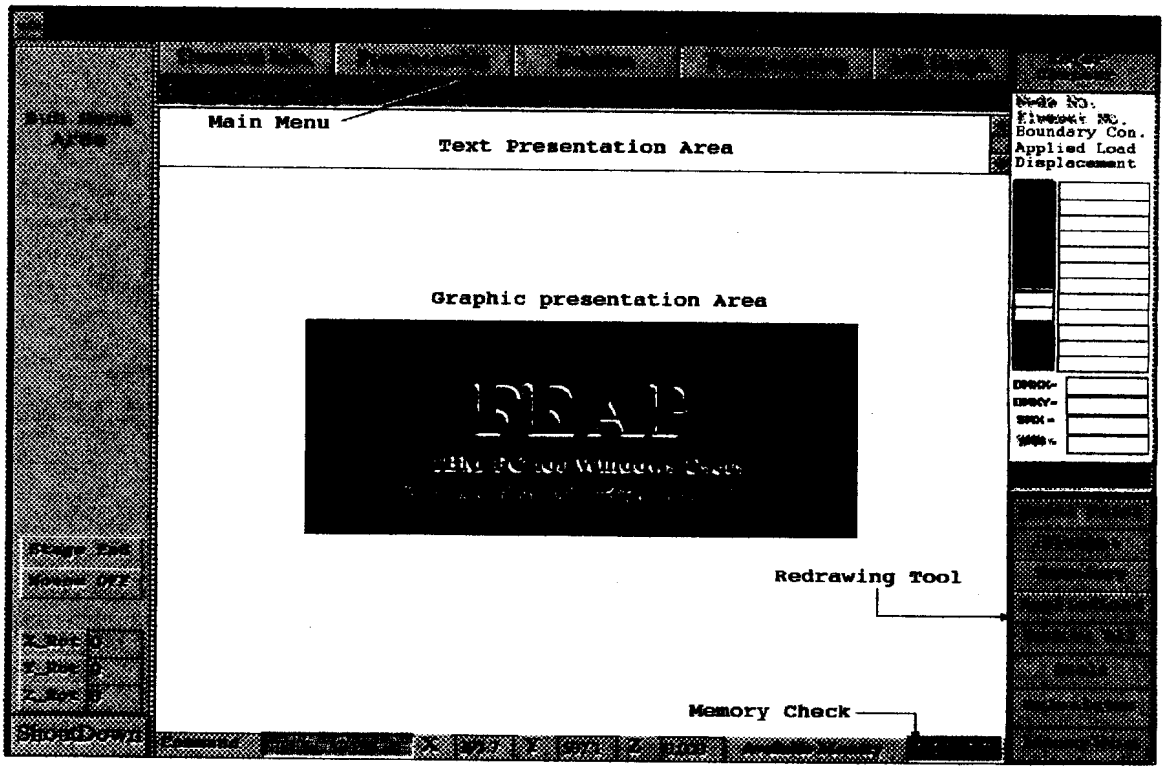


Fig. 2.1. Overall structure of integrated process system

시스템 구성요소는 주메뉴 영역(Main - Menu Area)과 부메뉴 영역(Sub - Menu Area), 그리고 그래픽 표현 영역(Graphic Presentation Area)으로 나뉘어진다. 주메뉴 영역은 VB에서 제공되는 버튼(Button) 컨트롤로써 구성되고, 부메뉴 영역에서의 그래픽 요소는 판넬(Panel) 및 버튼이 사용된다. 그래픽 표현 영역은 그림상자(Picture Box)의 요소로 구성된다. 각각의 메뉴를 구성하는 그래픽 객체의 특성이나 사건 프로시져(Procedure)의 작성, 자료의 범위(Scope) 등은 이후에 설명되는 각 단계의 세부 설계 방법에서 자세히 표현하고자 한다.

Ⅲ. 전처리기의 설계

1. 전처리기의 구조 및 사건중심적 프로그래밍

전처리 단계는에서는 절점좌표(Nodal Coordinates), 요소연결(Element Connectivity), 경계조건(Boudary Condition), 작용하중(Applied Load), 재료 특성치(Material Properties)등과 같은 새로운 데이타를 생성하는 과정과 기존의 데이타를 읽는 과정으로 구성되어 있고, 확인 및 수정 r기능을 포함한다. 전처리기의 구동은 주메뉴 영역에 있는 「Preprocess」버튼의 응답으로 발생하는 부메뉴의 세부 입력단계를 거치면서 이루어 진다. 이 때 부메뉴의 속성 변경을 이용하여 동일한 객체에 대해 다양한 조건에서의 응답을 유도한다. 주메뉴의 「Preprocess」객체에 해당되는 사건 프로시져는 마우스 클릭(Mouse - Click) 사건에 응답하도록 되어있다. 전처리기의 부메뉴는 동일한 그래픽 객체의 이름(Name)을 갖는 동적 객체 배열로 구성된다. Fig. 2.2는 전처리기의 데이타 처리 과정 및 메뉴 구성을 나타낸다.

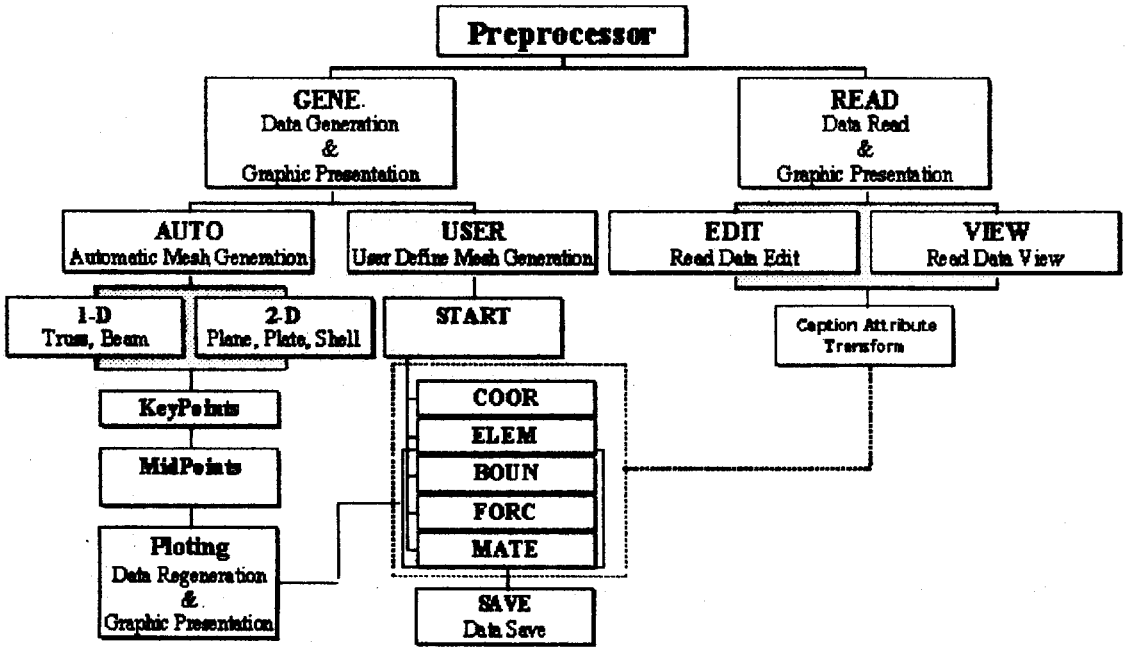


Fig. 2.2. Data input process and sub - menu structure of 「Prepocessor」

주메뉴 「Preprocess」 객체는 Table 2.1과 같은 사건 프로시져를 가지고 있고, 객체가 가지는 사건 프로시져와 응답에 대한 동작도 표현되어 있다. 사건 프로시져는 모든 것을 표현하지는 않고, 개발 시스템에서 주로 사용되는 사건에 대한 것만 기술하고 있다. 응답의 경우 이해를 돕기 위해 단순한 동작만을 표현하였다.

주메뉴 영역에 있는 「Preprocess」객체에 대한 응답 프로시져는 앞서 설명한 사건 프로시져의 생성에 대한 간략한 예로써 나타내었다. 즉, 마우스 클릭 사건에 응답하는 동작을 필요로 할 경우 「Mouse -

Table 2.1. Event procedure and response of 「Preprocess」object

Graphic Object	Name	Event Procedure	Response
Button	PREBUT	Mouse_Click( )	Sub - Menu.Visible=TRUE
		Mouse_DblClick( )	
		GetFocus( )	prebut.forecolor=&HFF0000
		LostFocus( )	prebut.forecolor=&H80&
		KeyPress( )	

Click( )」 프로시져에 동작되는 코드를 작성하면 된다. 마우스의 촛점(Focus)과 관련한 프로시져들은 현재의 단계를 표현하는 방법으로 주로 사용되고 동작하지 않는 메뉴들과 비교될 수 있게 배경에 대한 글자색을 변화시킨 것이다. 이것은 버튼 객체에 대한 객체속성(Object Attribute)을 변화시킨 것이다. 따라서 본 연구에서 사건중심적 프로그래밍의 적용은 기술한 내용과 같이 각각의 객체에 정의된 사건 프로시져만을 작성함으로써 이루어진다. 객체마다 정의된 사건 프로시져는 개발자가 필요로 하는 것만 작성하면 된다.

2. 객체배열 및 입력귀환

부메뉴의 경우 같은 종류의 객체들을 중복해서 사용해야 할 경우 객체이름 및 개발자가 필요로하는 속성을 정의하고, 「Index」를 첨자로 하여 배열화하는 객체배열을 적용한다. 객체배열을 사용하면 동일한 사건 프로시져를 공유하게 되며 코드작성을 효율적으로 할 수 있다. Table 2.2는 자동요소분할 과정[8,9]의 「Index」를 이용한 부메뉴의 일부 구조를 나타낸다.

Table 2.2. Sub - menu structure of Preprocessor using object array

Object	Name	Procedure	Index	Caption	Code
Panel	prepro( )	Mouse_Click( )	1	1 - D	Select Case Index+1
			2	2 - D	Case 1
			1	Key Points	:
			2	Mid Points	End Select
			3	PLOT	For i=1 to 2 (or 3)
			1	BOUN	
			2	FORC	
			3	MATE	prepro(i). Caption=meshCap(i)
			4	SAVE	Next i

부메뉴를 구성하는 그래픽 객체는 판넬로써 대부분 3차원 효과를 가지는 배경으로 많이 사용되나 프로그램의 흐름을 표현하는 속성이 다양하여 메뉴로 사용된다. 전처리기의 부메뉴에서 입력되는 내용은 그래픽으로 동시에 표현된다. 동일한 「Index」를 갖는 객체의 응답에 있어서는 객체의 속성을 변화 시킴으로써 사건 프로시져를 공유하게끔 설계할 수 있다. 전처리기에서 부메뉴 구조를 「Index」의 조건에 의해 나타내면 Table 2.3과 같다. (a)로 표시된 영역은 자동 메쉬생성 기능을 나타내고 (u)로 표시된 영역은 사용

Table 2.3. Sub menu structure using object arrary by Index and caption property

Index=1	Index=2	Index=3	Index=4	Index=5	Index=6
1 - D(a)	2 - D(a)	PLOT(a)	SAVE(u)		
Key Point(a)	Mid Point(a)	MATE(a)			
BOUN(a)	FORC(a)				
COOR(u)	ELEM(u)	BOUN(u)	FORC(u)	MATE(u)	SAVE(u)

자 정의 메쉬 생성 기능을 실행했을 경우다.

각각의 영역에 나타난 문자(String)은 객체의 「Caption」을 나타낸다. 즉, 하나의 객체에 해당되는 「Index」조건은 최대 2개까지 포함되지만 「EDIT」와 「VIEW」기능을 포함한다면 4개까지 포함된다. Fig. 2.3은 상기한 내용을 객체의 「Caption」속성을 변형하여 나타나는 메뉴를 표현한 것이다.

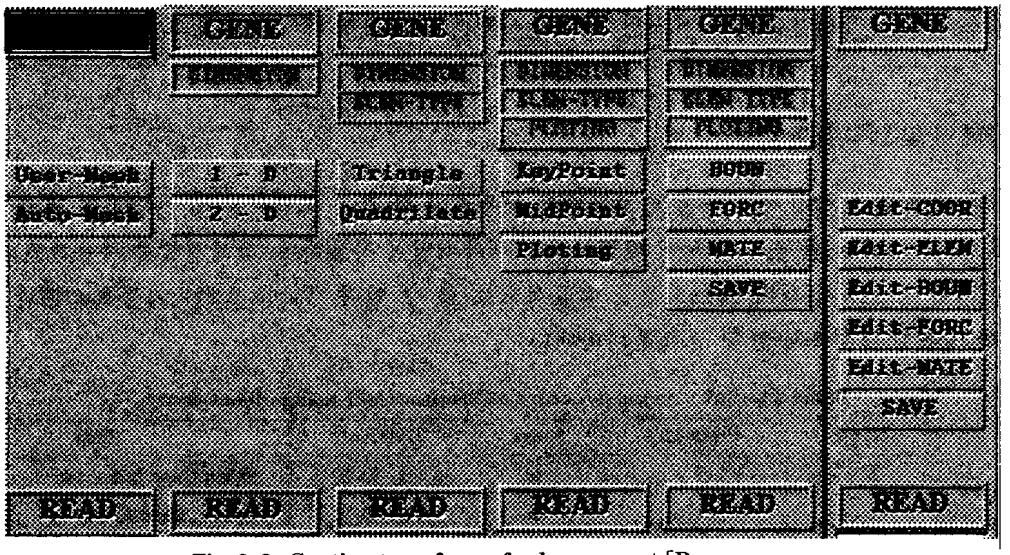


Fig. 2. 3. Caption transform of sub - menu at 「Preprocess」

동일한 「Index」나 「Caption」에 대해 개발 프로그램에서 실행되는 BASIC 코드는 다음과 같이 나타낼 수 있다.

```
Select Case Index
  Case 1
    if Object.Caption = "1 - D" then
        "1 - D" 에 해당하는 코드 실행
    elseif Oject.Caption = "KeyPoint" then
        "KeyPoint" 에 해당하는 코드 실행
                    ⋮
```

객체 배열을 사용할 경우 시스템 전체적으로 동적특성을 갖는 시스템을 설계할 수 있다. 또한 부메뉴에서 현재의 입력 단계에서 불필요한 입력메뉴(Input Menu)를 나타내지 않음으로써 입력과정에서 발생할 수 있는 입력 오류(Error)를 줄일수 있다. 입력귀환(Input Return)도 마찬가지의 개념으로 설계된다. 개발 프로그램의 전처리기에서의 입력귀환은 세부적인 단계의 데이타 입력에서 오류를 범하였을 경우 사용자가 원하는 단계부터 입력이 다시 이루어지게 할 수 있는 기능을 말한다. 전처리기에서의 세부적 단계와 입력귀환 메뉴는 다음과 같다.

Table 2.4. Input return menu at 「Preprocess」

<table>
<tr><td></td><td colspan="4">Input return menu</td></tr>
<tr><td rowspan="3">Sub - Menu</td><td>Return 0</td><td>Return 1</td><td>Return 2</td><td>Return 3</td></tr>
<tr><td rowspan="2">Auto Mesh</td><td>1 - D</td><td rowspan="2">Element Type</td><td rowspan="2">BOUN
MATE
FORC</td></tr>
<tr><td>2 - D</td></tr>
<tr><td>User Mesh</td><td colspan="3">COOR, ELEM, MATE, FORC, BOUN</td></tr>
</table>

Table 2.4와 같이 입력메뉴 설계에 있어서 각 세부단계를 표현할 수 있는 메뉴를 하나씩만 추가하면 된다. 추가된 귀환메뉴는 단순히 귀환된 범위내의 객체에 대한 「Caption」속성만을 표현하면 된다. 입력 데이타를 모두 입력한 후에 다시 「EDIT」기능으로 수정할 수가 있지만 전처리 단계에서의 입력과정을 표현하는 역할을 동시에 함으로써 사용자의 인식이 쉽다는 장점이 있다.

3. 전처리기의 구동

Fig. 2.4는 중앙에 구멍이 있는 평판에 인장력이 작용하고 있는 경우를 평면응력(Plane Stress) 문제로 해석하기 위하여, 대칭조건을 고려한 1/4 자동요소분할과 경계조건, 하중, 재료 특성치를 입력한 화면이다. 부메뉴에 있어서 마우스 클릭 사건에 대한 객체의 응답관계 및 조건식, 호출함수는 Table 2.5와 같다.

Table 2.5. Action by sub - menu property transfor

Name	Index	Condition	Caption Property	Call Procedure	Note
Prepro( )	1	.and.	1 - D		TRUSS, BEAM
	2	.and.	2 - D		TRIANGLE, QUADRILATERAL
	1	.and.	KeyPoint	Keypoint( )	Extraction of Keypoint
	2	.and.	MidPoint	Midpoint( )	Extraction of Midpoint
	3	.and.	PLOT	Plot( )	Data Creation & Graphic Modelling
	1	.and.	BOUN	Boun__Click( )	B.C. & Graphic Presentation
	2	.and.	FORC	Forc__Click( )	Loading & Graphic Presentation
	3	.and.	MATE	Mate__Click( )	Material Properties
	4	.and.	SAVE	Save__Data( )	Input Data Save

전처리기에서의 자료의 입력 방법은 키보드에 의해 정확한 좌표점을 입력하는 방법과 마우스를 직접 이용하여 좌표값를 입력하는 방법이 있다. 키보드에 의한 경우 VB에서 지원되는 대화 상자(Dialog Box)

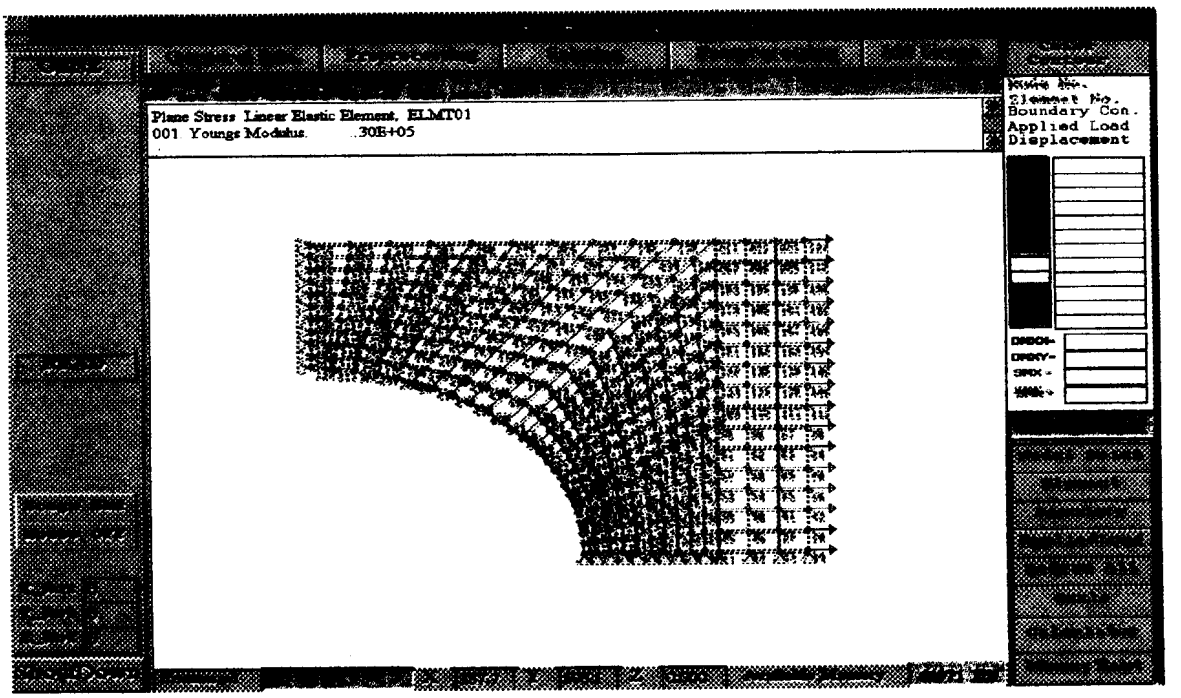


Fig. 2.4. Automatic mesh generation of input data at「Preprocessor」

를 이용한다. 그래픽 영역에 대한 모듈 수준에서의 접근은 모듈에 포함되어 있는 프로시져가 그래픽 객체에 출력을 필요로 할 경우 프로시져의 매개변수로 그래픽 객체를 넘기면 된다. 마우스를 이용하여「Keypoint」와「Midpoint」기능을 수행하는 과정은 Fig. 2.5와 같이 나타낼 수 있다.

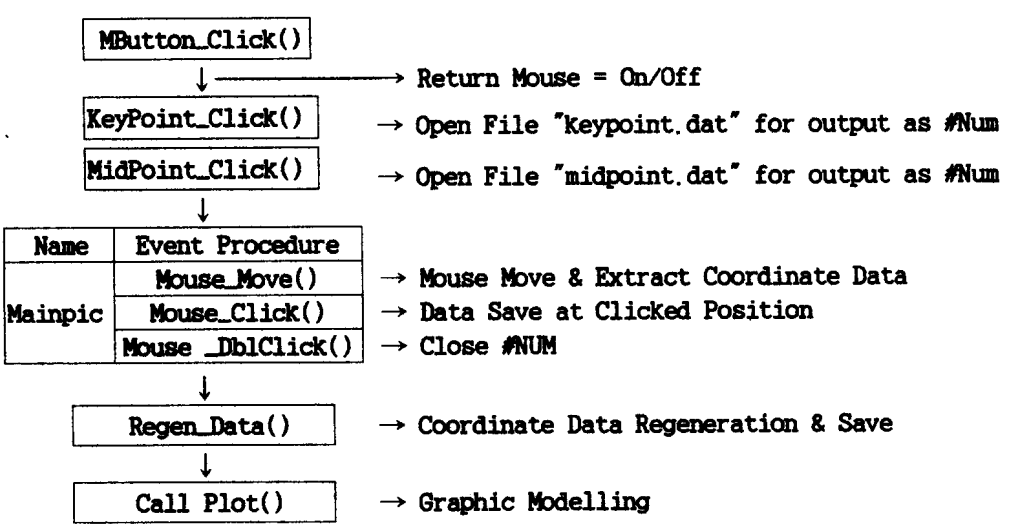


Fig. 2.5. Graphic modelling process using Mouse device

마우스를 이용하는 경우 정확한 좌표를 추출하기 위해 현재의 마우스 위치를 값으로 표현하여 사용자에게 알려준다. 위치에 대한 표현은 Fig. 2.5에서의 「Mouse_Move( )」프로시져를 작성하면 된다. 마우스의 제어(Control) 기능을 가지는 버튼에서 반환되는 값은 ON으로 가정하여 나타내었다.

Ⅳ. 후처리기의 설계

후처리 단계는 해석 단계에서 얻어진 결과를 그래픽 및 텍스트로 표현하고, 프린터(Printer) 출력의 처리 과정을 갖는다. 따라서 외부 장치(External Device)와 연결되는 프로그램 코드와 API 함수 호출, DLL 연결등 복잡한 요소로써 설계된다. 후처리 단계의 전체적인 부메뉴 구성은 Fig. 2.6과 같다. 후처리 단계는 Fig. 2.6과 같이 크게 4 부분으로 나뉘어지고 각 부분들은 그림상자와 글상자를 중심으로 상호 연결성을 갖고 결과를 표현한다. 응력 분포등과 같이 색표현(Color Contour) 방식에 의한 결과의 표현은 각 요소의 가우스 점(Gauss Point)에서의 결과치 또는 요소 전체에 걸친 평균치를 이용하여 표현할 수 있다.

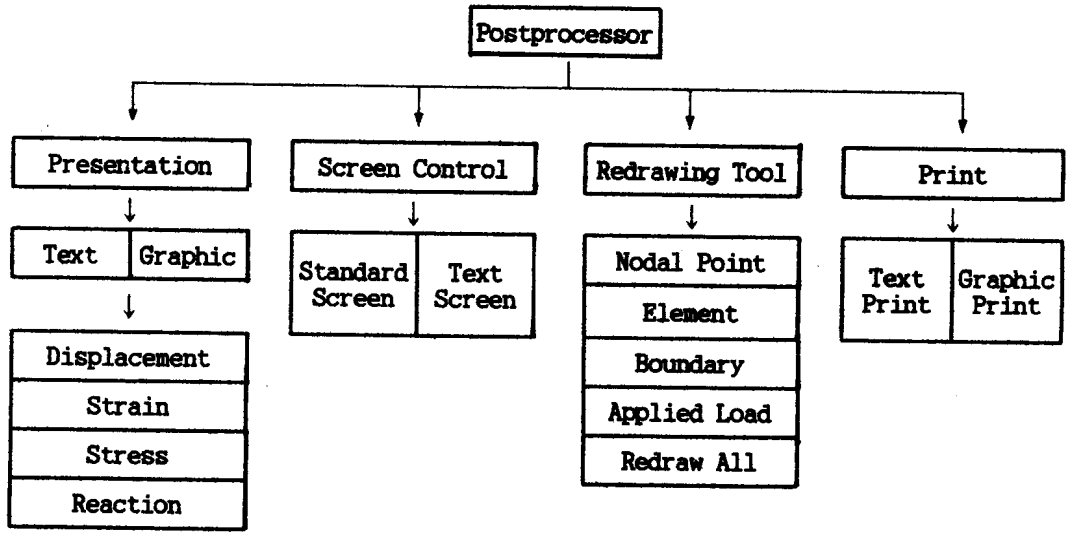


Fig. 2.6. Sub - menu structure at Postprocessor

1. 동적연결 및 API

라이브러리(Library)는 일반적으로 정적 연결라이브러리(SLL : Static Link Library)와 동적 연결라이브러리(DLL : Dynamic Link Library)로 나뉘어 진다. SLL의 경우 프로그램을 컴파일(Compile)한 후에 생기는 오브젝트 파일(Object File)을 링크시킬 때 사용하는 라이브러리다. 이 파일에는 가장 많이 사용되는 함수의 수행 코드가 담겨져 있으며, 링크시 사용한 함수의 수행코드를 실행파일에 삽입하게 된다[10]. 따라서 실질적으로 수행속도 면에 있어서는 DLL보다 빠르지만 윈도우즈와 같이 다중처리 환경을 이용하는 운영체계의 경우 SLL은 시스템에서 많은 메모리를 차지하기 때문에 적합한 방법이 되지는 못한다. 정적

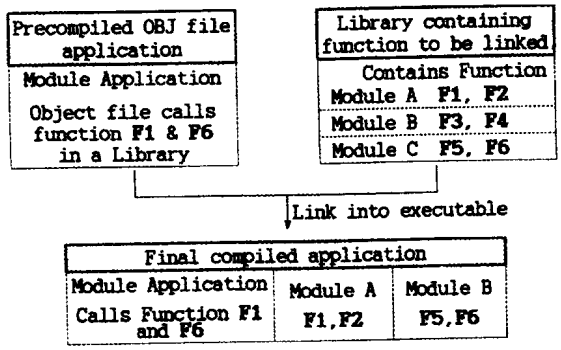


Fig. 2.7. Static link process

링크 과정을 그림으로 나타내면 Fig. 2.7과 같다.

반면에 동적 연결라이브러리는 링크시 함수에 대한 수행코드의 삽입이 이루어지지 않고 수행코드가 있는 라이브러리 이름과 파일 내의 코드 위치만을 수행코드에 적어둔다. 프로그램이 실행되면 DLL이 메모리에 상주해 있다가 해당 함수를 호출하면 DLL에 있는 수행코드가 실행된다. Fig. 2.8은 동적 연결과정을 도식화한 것이다.

결론적으로 윈도우즈 환경에서의 DLL의 사용은

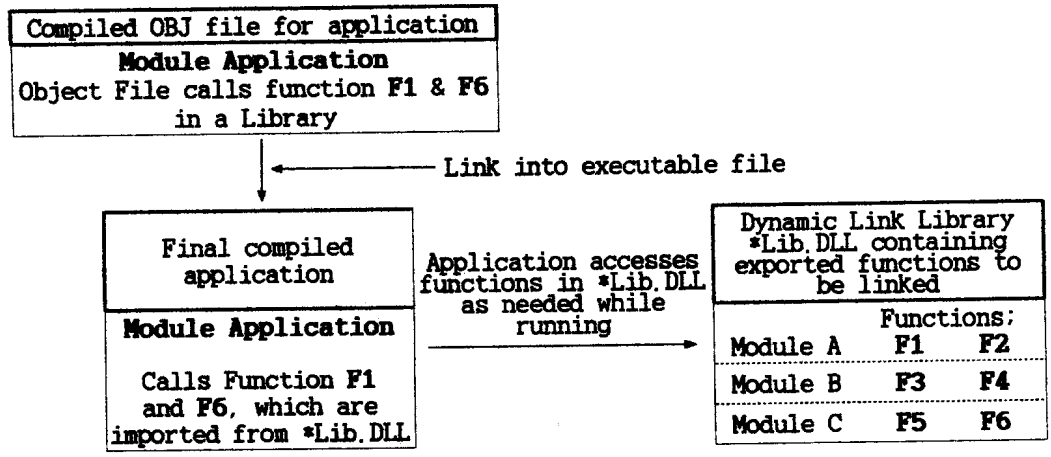


Fig. 2.8. Dynamic link process

수행코드를 공유하여 효율적인 메모리 사용가능하게 한다. 그러나 메모리의 경우에서 뿐만 아니라 하드웨어 공유도 가능하기 때문에 DLL의 사용은 윈도우즈 응용 프로그램에 있어 필수요소라고 할 수 있다. 실질적으로 하드웨어는 다중처리가 불가능하다. 따라서 대부분 다중처리 기능을 갖는 운영체계는 장치드라이버(Device driver)를 이용하여 하드웨어를 사용하게 된다. 즉, 동시에 두개의 프로그램이 하드웨어를 사용하는 경우에는 장치드라이버를 이용하여 소프트웨어적으로 해결하게 된다. 이 때 장치드라이버는 DLL이 된다. VB도 실행파일(Executable File)을 만들게 된다. 그러나 실행파일은 보통 윈도우즈 실행파일들과 다르다. 그것은 가상코드(Pseudo code) 또는 「P - Code」[11]로 표현되는 특별한 종류의 간접적인 코드를 함유하며, VBRUN***.DLL(*** : VB version)로 나타내는 VB DLL에 의해 번역된다.

API는 DLL과 수입라이브러리(Import library)를 총괄하여 표현하는 것으로 프로그램 사이의 자료교환과 장치 독립(Device Independent), 다중처리, 동적연결등의 특성을 갖는 응용 프로그램을 개발을 가능하게 한다. 특히 후처리 기능에 있어서 윈도우상에서 그래픽 효과를 얻기 위해서는 윈도우즈 GDI(Graphic Device Interface) 객체를 사용한다. 후처리 단계에서 각각의 요소에 대한 결과치를 색표현하기 위하

여 개발 시스템에서는 API[12]를 이용하였다. API 함수의 호출은 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[13].

```
Declare Function Polygon% Lib "GDI" (ByVal hDC%, lpPoints as POINTAPI, ByVal nCount)
Declare Function SetPolyFillMode% Lib "GDI" (ByVal hDC%, ByVal nPolyFillMode%)
```

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

Fig. 2.3에서 자동요소 분할된 평판에 인장력이 작용하고 있는 평면응력문제를 해석한 결과를 그래픽과 텍스트로 나타내면 각각 Fig. 2.9~Fig. 2.11과 같다.

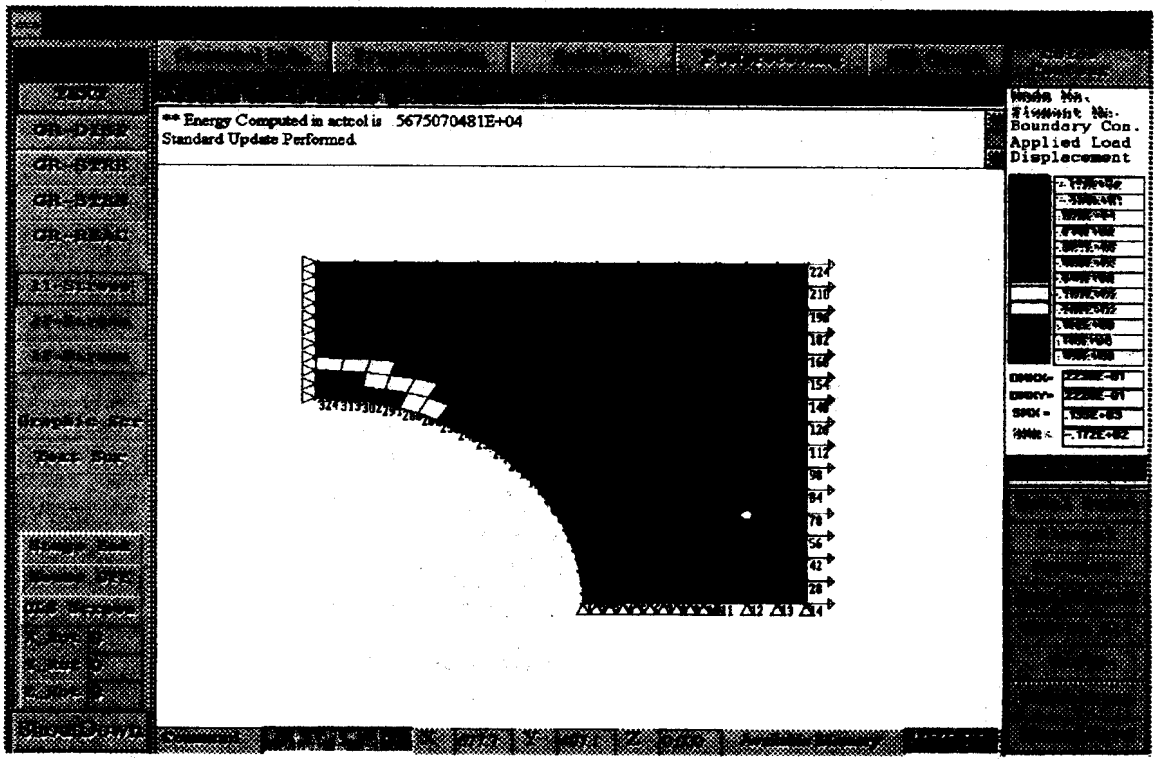
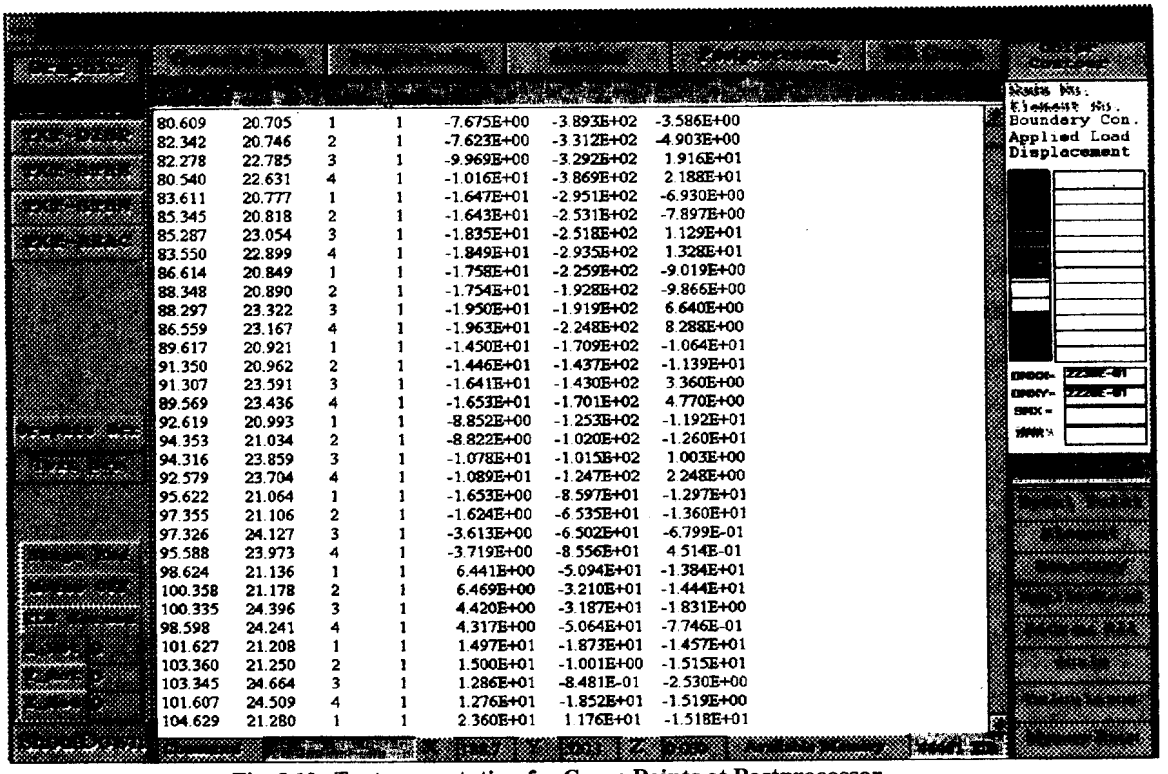


Fig. 2.9. Stress graphic presentation at Postprocessor



80.609	20.705	1	1	-7.675E+00	-3.893E+02	-3.586E+00
82.342	20.746	2	1	-7.623E+00	-3.312E+02	-4.903E+00
82.278	22.785	3	1	-9.969E+00	-3.292E+02	1.916E+01
80.540	22.631	4	1	-1.016E+01	-3.869E+02	2.188E+01
83.611	20.777	1	1	-1.647E+01	-2.951E+02	-6.930E+00
85.345	20.818	2	1	-1.643E+01	-2.531E+02	-7.897E+00
85.287	23.054	3	1	-1.835E+01	-2.518E+02	1.129E+01
83.550	22.899	4	1	-1.849E+01	-2.935E+02	1.328E+01
86.614	20.849	1	1	-1.758E+01	-2.259E+02	-9.019E+00
88.348	20.890	2	1	-1.754E+01	-1.928E+02	-9.866E+00
88.297	23.322	3	1	-1.950E+01	-1.919E+02	6.640E+00
86.559	23.167	4	1	-1.963E+01	-2.248E+02	8.288E+00
89.617	20.921	1	1	-1.450E+01	-1.709E+02	-1.064E+01
91.350	20.962	2	1	-1.446E+01	-1.437E+02	-1.139E+01
91.307	23.591	3	1	-1.641E+01	-1.430E+02	3.360E+00
89.569	23.436	4	1	-1.653E+01	-1.701E+02	4.770E+00
92.619	20.993	1	1	-8.852E+00	-1.253E+02	-1.192E+01
94.353	21.034	2	1	-8.822E+00	-1.020E+02	-1.260E+01
94.316	23.859	3	1	-1.078E+01	-1.015E+02	1.003E+00
92.579	23.704	4	1	-1.089E+01	-1.247E+02	2.248E+00
95.622	21.064	1	1	-1.653E+00	-8.597E+01	-1.297E+01
97.355	21.106	2	1	-1.624E+00	-6.535E+01	-1.360E+01
97.326	24.127	3	1	-3.613E+00	-6.502E+01	-6.799E-01
95.588	23.973	4	1	-3.719E+00	-8.556E+01	4.514E-01
98.624	21.136	1	1	6.441E+00	-5.094E+01	-1.384E+01
100.358	21.178	2	1	6.469E+00	-3.210E+01	-1.444E+01
100.335	24.396	3	1	4.420E+00	-3.187E+01	-1.831E+00
98.598	24.241	4	1	4.317E+00	-5.064E+01	-7.746E-01
101.627	21.208	1	1	1.497E+01	-1.873E+01	-1.457E+01
103.360	21.250	2	1	1.500E+01	-1.001E+00	-1.515E+01
103.345	24.664	3	1	1.286E+01	-8.481E-01	-2.530E+00
101.607	24.509	4	1	1.276E+01	-1.852E+01	-1.519E+00
104.629	21.280	1	1	2.360E+01	1.176E+01	-1.518E+01

Fig. 2.10. Text presentation for Gauss Points at Postprocessor

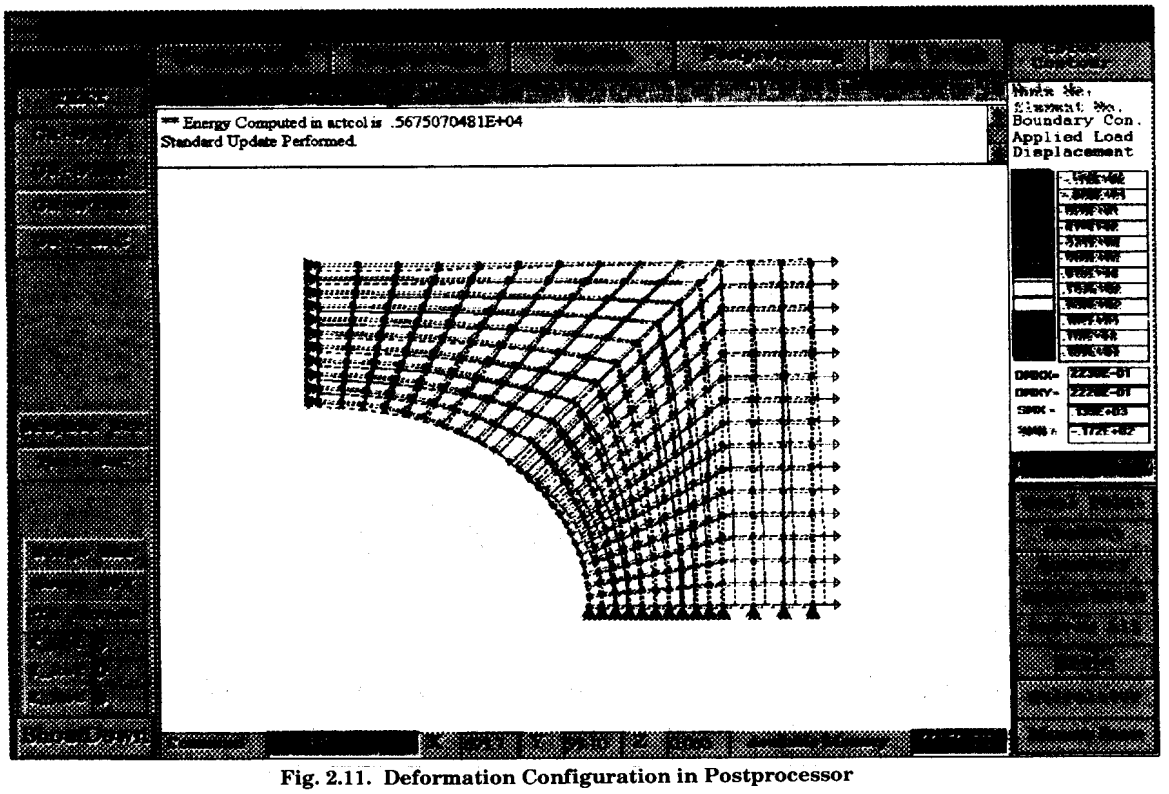


Fig. 2.11. Deformation Configuration in Postprocessor

2. 객체연결 및 포함(OLE)

OLE는 윈도우즈 응용 프로그램들 사이의 자료교환 및 제어를 자연스럽게 이루어지게하는 특징이 있다. 또한 프로그램 개발에 있어 「DBase」나 그래픽처리를 위해 별도의 프로세서(Processor)의 개발없이 기존의 개발된 동일한 기능을 갖는 프로그램을 개발 프로그램에 그대로 접속하여 사용할 수 있다. 따라서 프로그램의 개발 시간을 현저히 줄일수 있을 뿐만 아니라 전문적인 기능을 갖는 독립 프로그램을 그대로 사용함으로써 개발 프로그램의 신뢰성을 높일 수 있다. OLE를 이용한 데이타 교환방식은 Fig. 2.12와 같이 나타낼수 있다.

OLE는 두개의 프로그램이 1 : 1의 성질로 연결이 되는 것이 아니라 중심 프로그램을 두고 자료교환이 이루어지게끔 만든다. 즉, 데이타를 제공하는 프로그램과 데이타를 받는 프로그램으로 나뉘게 된다. 이때 데이타를 제공하는 프로그램을 클라이언트(Client) 프로그램, 자료를 받는 프로그램을 서버(Server) 프로그램으로 표현한다. OLE는 윈도우즈

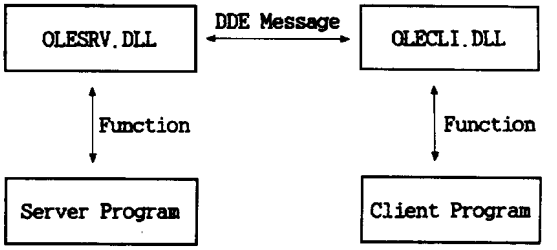


Fig. 2.12. Dynamic data exchange by Object Linking Embedding

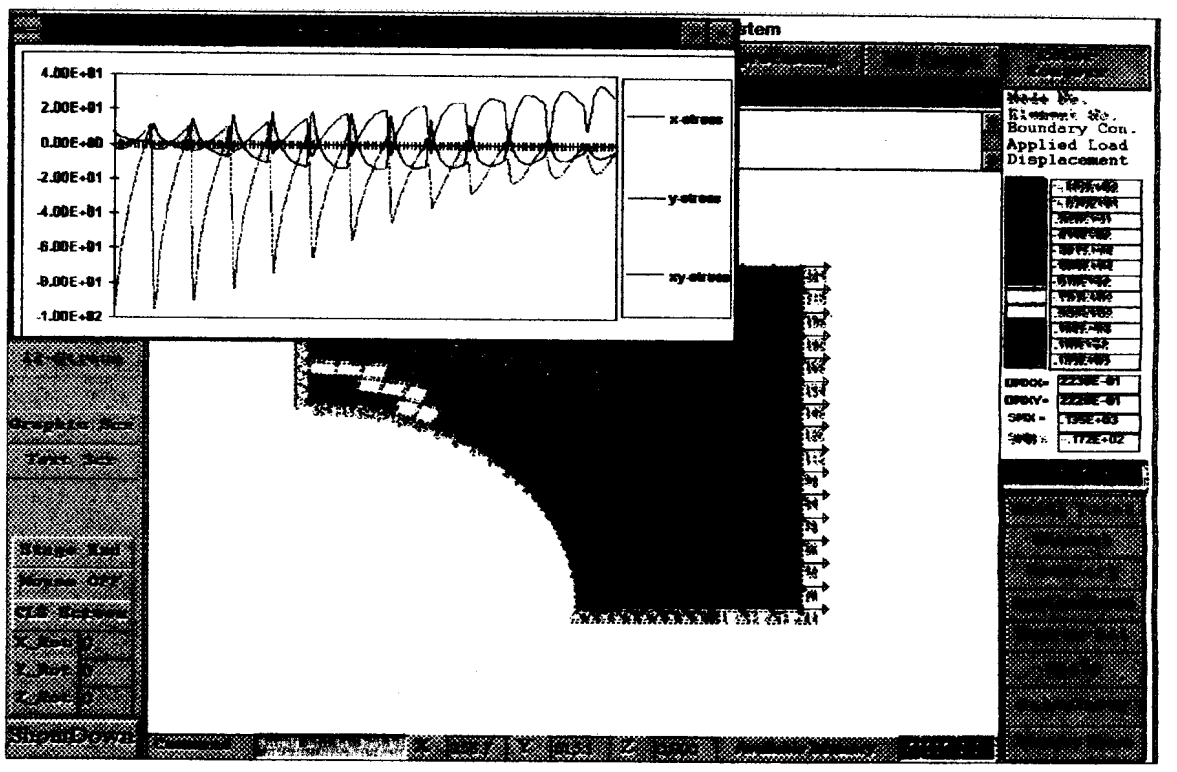


Fig. 2.13. Graph presentation using Object Linking Embedding

가 표준으로 제공하는 함수, 즉 API를 제공한다. 따라서 프로그램을 개발할 때 이 API를 이용하여 프로그래밍하면 모든 OLE기능이 있는 프로그램과 연결이 가능하다. Fig. 2.13는 후처리기에서 OLE 기능을 이용하여 요소에 대한 각 방향의 응력을 그래프로 나타낸 것이다. Fig. 2.13에서는 특별한 의미을 갖는 그래프로 보여주는 것이 아니라 OLE기능을 이용하여 프로그램의 컴퍼넌트 구성이 이루어지는 것을 보여주기 위한 것이다.

V. 결 론

본 연구에서는 실질적인 프로그램 개발을 통하여 VB을 이용하여 윈도우즈용 구조해석 프로그램을 효율적으로 제작하는 방법을 제시하였다. 시스템 전체적으로 윈도우즈 응용 프로그램이 필수적으로 갖게되는 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 있슴을 알 수 있었다. 윈도우즈가 제공하는 DDE 및 OLE기능을 이용한 프로그램 설계는 프로그램의 컴포넌트(Component) 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 요소 라이브러리(Element Library)의 확장과 후처리기에 있어서 OLE 적용 범위를 넓히고 세부적인 기능의 추가가 필요하다고 사료된다.

참고문헌

1) Microsoft, Window 3.1 User Guide, 1993.
2) Clayton, W., Borland C++ for Windows Programming, 2Ed., Prentice - Hall, 1994.
3) Leestma, M., Object - Oreinted Programming Turbopascal, Merril - Macmillan, 1993.
4) James Martin, Principles of Object - Oriented Analysis and Design, Prentice - Hall, 1993.
5) Robert, W. Stewart, Graphic Programming with Visual Basic, SAMS, 1995.
6) Ori Gurewich & Nathan Gurewich, How to Creat Real - World Application with Visual Basic, SAMS, 1994.
7) Geary Entsminger, The Secrete of Visual Basic 3.0 for Windows, SAMS, 1994.
8) 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.
9) P. L. George, Automatic Mesh Generation Application to Finite Element Method, WILEY, 1991.
10) Daniel Appleman, Visual Basic Programmer Guide to the Windows API, Ziff - Davis, 1993.
11) Geary Entsminger, Visual Basic 3.0 Master, SAMS, 1994.
12) James L. Conger, Windows API Bible, SAMS, 1994.
13) Zane Thomas & Robert Arnson, Visual Basic How - To, 2Ed., Waite - Group, 1993.