# “GUI FOR FINEART SOFTWARE”

**Project submitted in partial fulfillment of the requirements for the**

**award of the Degree of**

# Bachelor of Computer Applications of

**SASTRA DEEMED UNIVERSITY**

# Submitted by “R.NITHYA”

**Register No: -------------**

# Under the Supervision and Guidance of

**--------------**



# School of Computing SASTRA Deemed University

**(Under section 3 of the UGC Act, 1956)**

# Thanjavur – 613 401

**January – 2024**

# School of Computing SASTRA University

**(U asnder section 3 of the UGC Act, 1956)**

# Thanjavur – 613 401



**Bonafide Certificate** Certified that this project report entitled **“GUI FOR FINEART SOFTWARE”**

Is a bonafide record of work done by

# “R.NITHYA”

**Register No. 010970889**

In partial fulfillment of the requirements for award of the Degree of **Bachelor of Computer Applications**

During the year 20-- -20--

# Project Guide

**Submitted for Project viva-voce examination held on --------**

# Examiner -I Examiner-II

Acknowledgements

I would like to thank our Honourable Chancellor Prof.R.Sethuraman for providing me with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

I would like to thank our Honourable Vice-Chancellor Dr.S.Vaidhyasubramaniam and Dr.S.Swaminathan, Dean, Planning & Development, for the encouragement and strategic support at every step of our college life.

I extend my sincere thanks to Dr.R.Chandramouli, Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

I extend my heartfelt thanks to Dr.V.S.Shankar Sriram, Dean, School of Computing, Dr.R.Muthaiah, Associate Dean-Research, School of Computing Dr.K.Ramkumar, Associate Dean-Academics, School of Computing Dr.D.Manivannan, Associate Dean-Intrastructure School of Computing and Dr.R.Alageswaran, Associate Dean-Student Welfare, School of Computing for their motivation and support offered in materializing this project.

My guide XXXXXXX (your guide’s name), School of Computing was the driving force behind this whole idea from the start. Her deep insight in the field and invaluable suggestions helped me in making progress throughout our project work. I also thank the project review panel members for their valuable comments and insights which made this project better.

I would like to extend my gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped me in the completion of the project.

I gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the successful completion of this project. I thank you all for providing me an opportunity to showcase my skills through project.

# SYNOPSIS

This project entitled with “GUI for FINEART (Finite element Engineering analysis using Adaptive Refinement Techniques) software”. This project simplifies the analysis and design of engineering structures such as bridges, offshore platforms, ship hulls and aircrafts using finite element method (FEM). In the analysis of such engineering structures accurate determination of responses is of considerable importance for economical and safe design. FEM offers solution to almost all structural analysis problems once a suitable formulation and computational model are adopted. Analysis of complex structures by using FEM requires large amount of input data to be prepared to represent the problem accurately. This gives ample scope for errors in input data and inefficient modeling. The focus will be on developing GUI and data handling required to represent geometry of engineering structures.

# Table of Contents

|  |  |  |
| --- | --- | --- |
| S.NO | CONTENT | PG NO |
| 1 | **Introduction**  Problem Statement |  |
| 2 | **Software Project Plan**  Time Schedule for Various phases |  |
| 3 | **Functional requirements Analysis**   * 1. Functional requirements   2. Non-functional requirements |  |
| 4 | **System Analysis**  Data Flow Diagram |  |
| 5 | **Design**   * 1. Front End Design   2. Back End Design |  |
| 6 | Coding |  |
| 7 | **Testing**   * 1. Unit Testing   2. Integration Testing   3. Validation Testing   4. Debugging |  |
| 8 | **Implementation**   * 1. Problem Faced   2. Lessons Learnt |  |
| 9 | **Future plans** |  |
| 10 | **Bibliography** |  |

***INTRODUCTION***

# Introduction Problem Statement

Finite element Engineering analysis using Adaptive Refinement Techniques (FINEART) is a software developed and copyright by SERC. The software has a back- end module with advanced analysis capabilities which is supported by front-end module for graphical pre-processor requirement. The software presently runs on Linux platform using FORTRAN and C compliers for back-end module while the graphical pre- processor module requires X-window/Motif/OpenGL libraries. Based on the feed back from the users, SERC has proposed to port this software to MS-Windows platform using compatible compilers and graphical libraries with improved GUI interface and graphical tools. Pre-processing software should be capable of building the complex geometry of a structure using simple geometry entities. The project aims to develop a pre-processor for FINEART software, graphical user interface (GUI) and associated graphics capabilities. The focus will be on developing graphical objects in terms of 2-D surfaces, 3-D volumes to model complex geometry of a given structure.

FINEART software analysis is concerned with finding the response of a structure for the given set of loads. The first task in carrying out a finite element analysis is to represent all the important characteristics of the structure in a format compatible with the software. To use a software, however, it is required to model the geometry and other details pertaining to the structure. Graphics based pre-processing helps an analyst to model the structure as well to visualize the major features of a structure. This includes geometry, load, boundary and other interface of the structure. Therefore, pre-processing plays an important role in the overall task of structural analysis.

## Objectives:

* To study and understand the basic requirements of a graphical pre-processor of a FE analysis software including the modeling capabilities of the MS-windows version of FINEART
  + To design and develop GUI and data handling related to the 3-D geometry and finite element modeling
  + To develop and implement program modules related to GUI

***SOFTWARE PROJECT PLAN***

# Software Project Plan

**Time Schedule for Various Phases**

## Project Starting Date:

**Project Finishing Date:**

|  |  |
| --- | --- |
| **Description of the Task** | **No. of days to complete** |
| **Software requirements gathering** |  |
| **Software Analysis** |  |
| **Design** |  |
| **Implementation** |  |
| **Testing** |  |
| **Reports** |  |
| **Deployment** |  |
| **Future plan** |  |

***SOFTWARE REQUIREMENTS SPECIFICATION***

different from a flowchart, which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances.

1. **Software Requirements specification**
   1. **Functional requirements (Data Flow Diagram)**

A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design).

On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink**,** via an internal process.

A DFD provides no information about the the timing or ordering of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite

There are different notations to draw data-flow diagrams, defining different visual representations for processes, data stores, data flow, and external entities

Dataflow Processes

Terminators Data store

* 1. **Processes** -- The only *active* elements. Processes cause something to happen. They have embedded descriptions, often in verb-object form. (Sometimes informally called "bubbles" because of their shape in an early version of SA.)
  2. **Terminators** -- Represent users or other systems, i.e. entities outside the boundary of the system being described.
  3. **Data flows** -- Composite data items (or objects) that pass either
     1. from any element to a process (input dataflow) or
     2. from a process to any element (output dataflow)
  4. **Data stores** -- Holding places for data flows; often implemented by databases.

# Non-functional requirements

## Resource requirements Hardware Requirements:

Machine :

RAM :

Video RAM :

Hard Disk :

Monitor :

Keyboard :

Mouse :

## Software Requirements:

Packages : MS ACCESS of MS Office XP

Operating System : windows XP

Graphical User Interface : Visual C++ 6.0 of Microsoft Visual

studio 6.0

## Performance Requirements

The process requirement is to get the value according to user’s requirement. Structural analysis using finite element methods requires that the software should be menu-driven. The menus should be very user-friendly and easily accessible with no additional learning about the system. Faster and accurate calculation is to be performed. Graphical plots are generated through graphs with high resolution and the system provides number of values as outputs including graphical outputs. The system should display the plot results and report generation of its configurations, which helps the end user to analyze the result within limited time for further iterations of its growth. Also,

the user input dialog boxes should be designed with care that helps the user to enter only appropriate values within the accepted range for proper computation of result and the retrieval of the values to the user can take place quickly. Finally the report should be generated based on the given input details and it greatly assists the user to know about the output obtained along with the input given within the acceptable ranges.

## Safety Requirements

The access to the system restricted with the privileges. Only authentication users can access the system. The user can use the exe of a project that makes safeguard, and the action to be performed to prevent the unnecessary works.

## Security Requirements

There is no confidential information required for structural analysis using finite element method. Hence, security requirements are not necessary. However, the people who are comfortable in working with the existing software VC++ should be made to work for finite elements without any security threat.

***SYSTEM ANALYSIS***

# System Analysis

## Data flow Diagram Level 0 DFD:

FE model

**Level**

**1 DFD:**

Geometry

## Level 2 DFD:

FINEART

Id, Coordinates

User

Create, Edit, Delete,

s

Generate Poi View

nt

User

Point

On Screen

Analysis

Display

Create, Edit, Delete,

Id, Coordinates

Create, Edit, Delete, GenPeoraintteinPfoirnmtsation

Generate Lines, Curves, Surfaces, Volumes

Data Store

## Level 3 DFD:

Point id, line id,

surface id, curv volume id

Line, curve, Volume & Surface

Point information

User

Option

D

Id, Coordinates

e id,

Create, Edit, Delete, Generate Lines,

Curves, Surfaces,

ata Store Create, Edit, Delete,

Volumes

**View options-**Zoom in, Zoom out Rotate, Pan

**Analysis options**-Static, Dynamic, vibration

**FE model-**Meshing, loading, element, boundary, stiffeners,

Generate Points

On Screen Display

Point id, line id, surface id, curve id, volume id

Point information Data Store

***DESIGN***

* 1. **Design**
  2. Front End Design

Figure 5.1(a) and 5.1(b) diagrammatically shows how the process works.

Fig.5.1 (a) Main menu

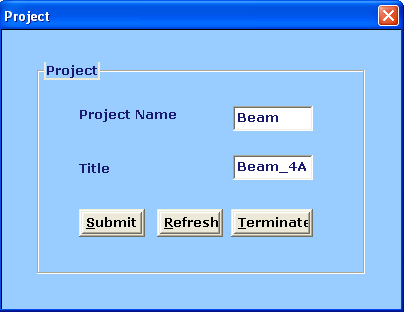


Fig.5.1 (b) Project name and title

## Pre-processor to FINEART

The pre-processor to FINEART consists of the major features of FINEART. It consists of view port, tree view and icons. It may be noted that in a numerical method such as finite element method. The main requirement for the preprocessor is to develop object in terms of 2-Dimensions surfaces and 3-Dimensions volumes. This is required to model complex geometry of the structure using simple geometrical objects. The overall part of the project will be taken in. That consists of geometry point, FE-Model Analysis, view model and the analysis model.

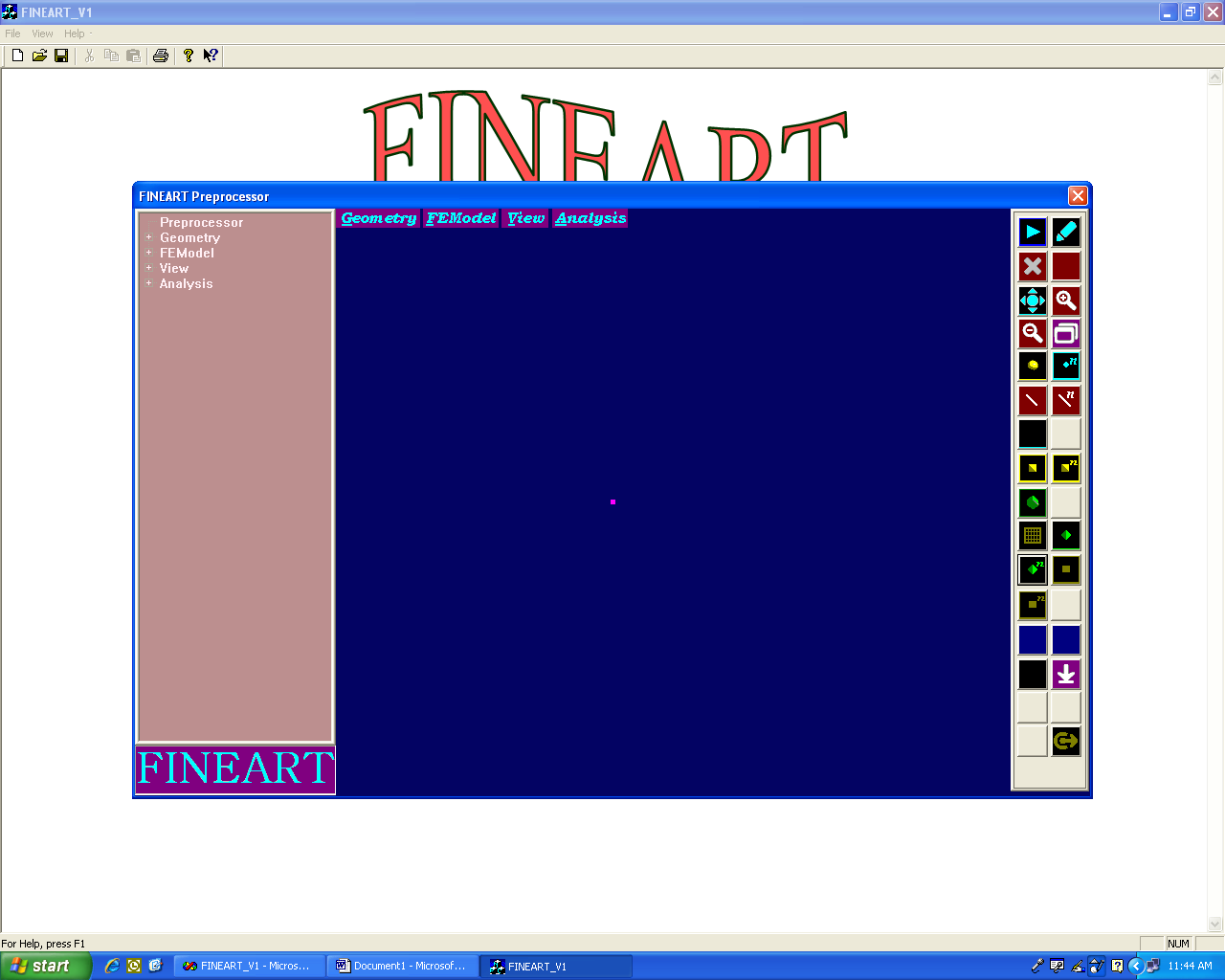


Fig 5.1[c] View port of the entire project

## Geometry-Point:

This module consists of point details. In this module there are four major operations that can be performed. They are create point, edit point, generate point and delete point. Each operation is performed for both 2-dimensions and 3-dimensions. When the user clicks for 2-dimensions the coordinates will be displayed correspondingly as X-Y, X-Z, and Y-Z, whereas for 3-dimension it will be performed for all the coordinates X, Y, Z.

All these operation deal with the point-id values and the coordinates values. Refer Figure 5.1(d) for details of the point module.

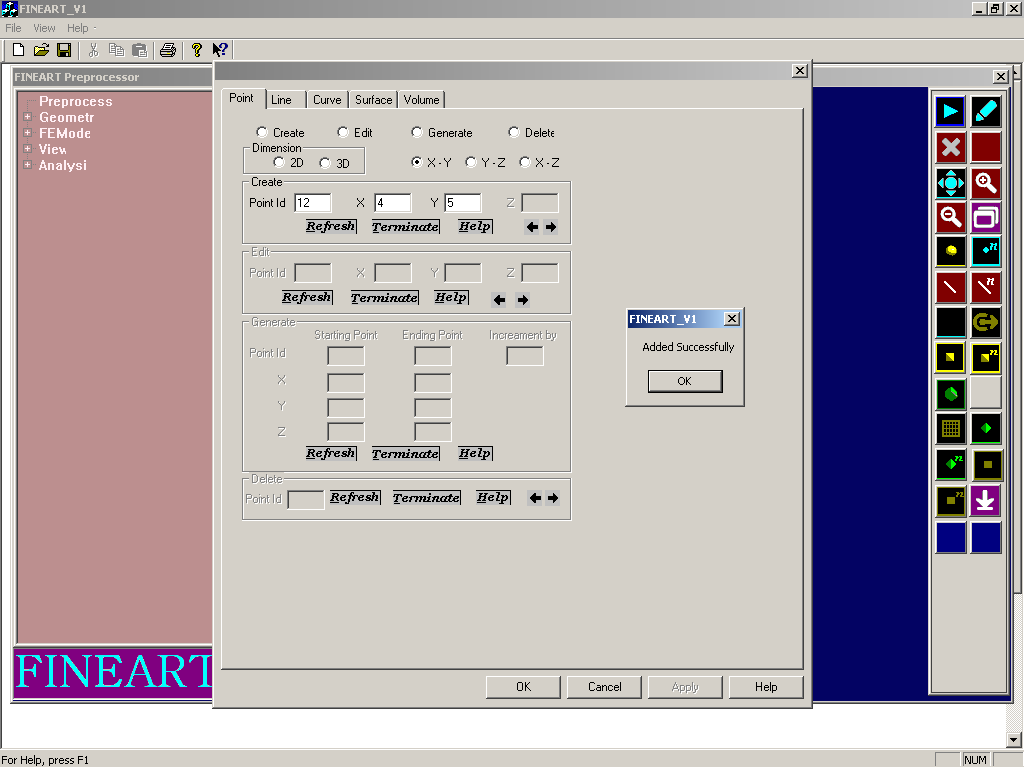


Fig 5.1[d] Geometry-Point

## Geometry-Line:

This module consists of line details. In this module there are four major operations that can be performed. They are create line, edit line, generate line and delete line. All these operations deal with line-id values and coordinate values. Refer Figure 5.1(e) for details of the for line module.

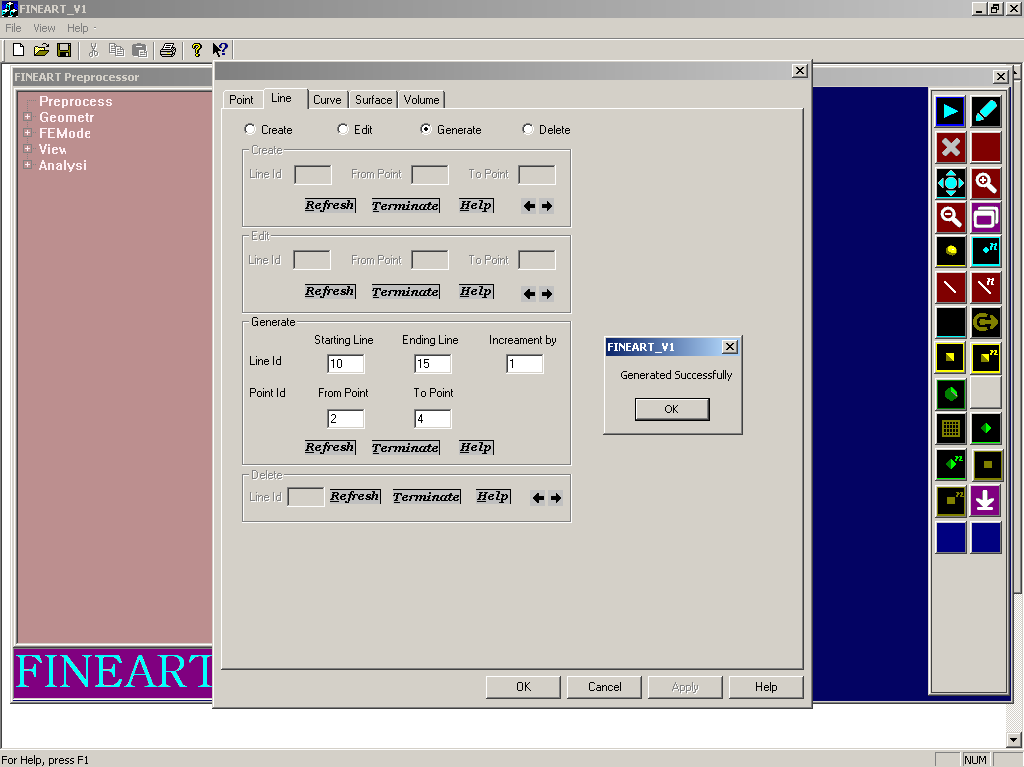


Fig 5.1[e] Geometry-Line Geometry-Curve:

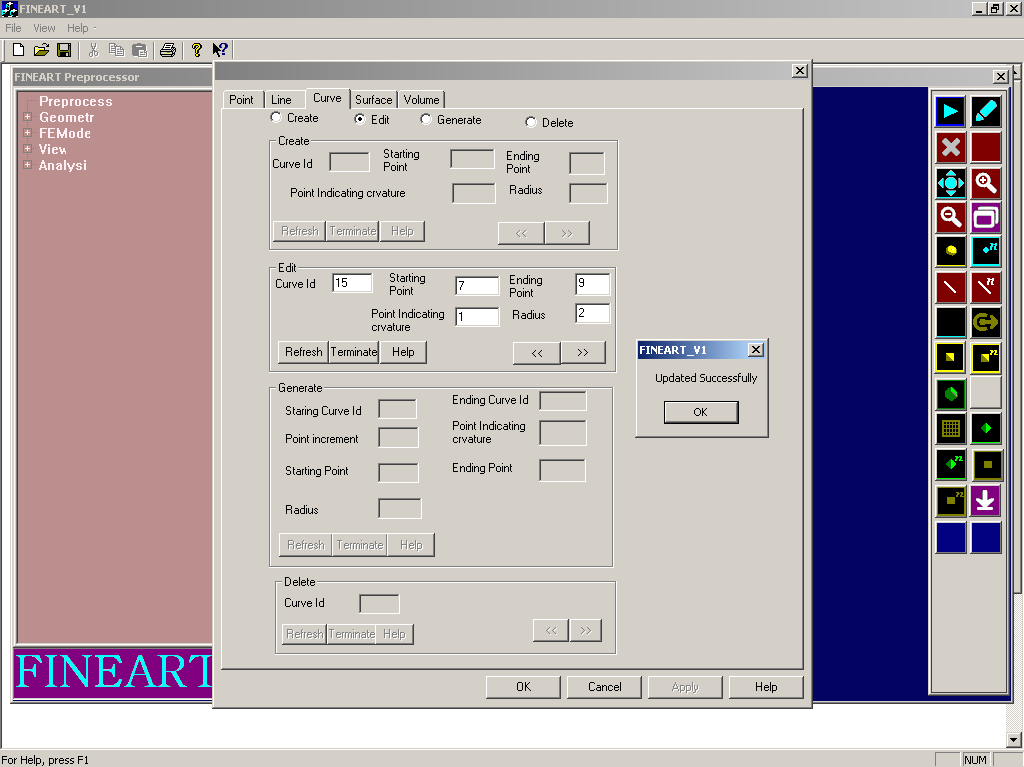


Fig 5.1[f] Geometry-Curve

This module consists of curve details. In this module there are four major operations that can be performed. They are create curve, edit curve, generate curve and delete curve. All these operations deal with line-id values and coordinate values. Refer Figure 5.1(f) for details of the curve module.

## Geometry-Surface:

This module consists of surface details. In this module there are four major operations that can be performed. They are create surface, edit surface, generate surface and delete surface. All these operation deals with the surface-id values and the coordinate values. Refer Figure 5.1(g) for details of the surface module.

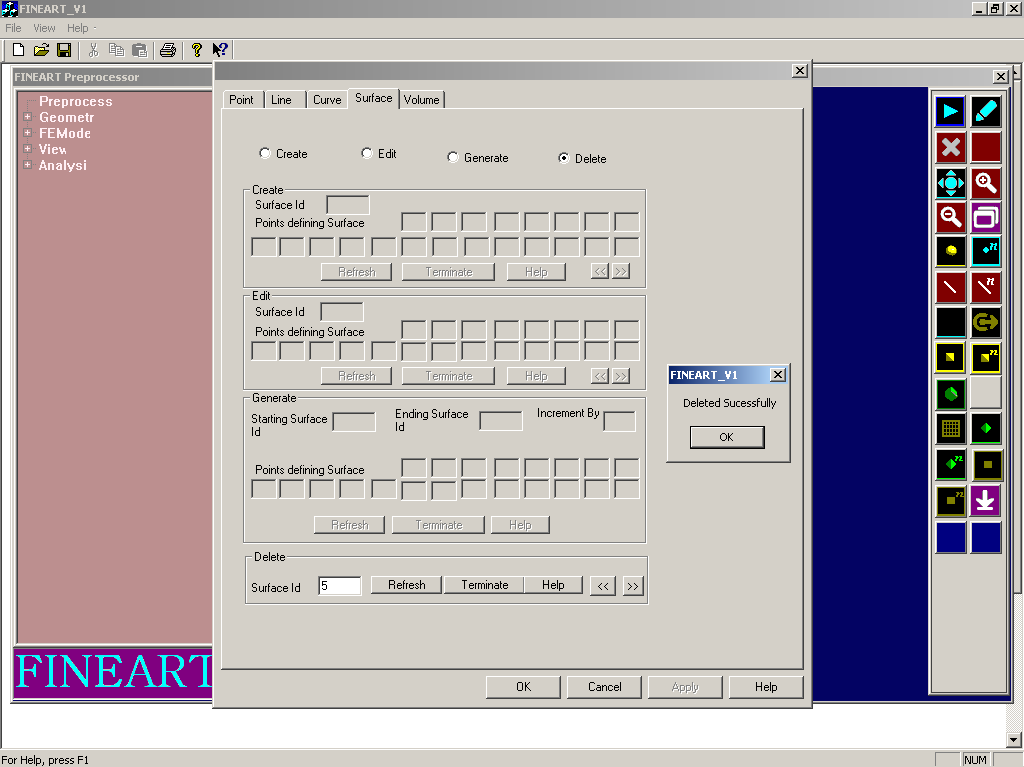


Fig 5.1[g] Geometry-Surface

## Geometry-Volume:

This module consists of volume details. In this module there are three major operations that can be performed. They are create volume, edit volume and delete volume. All these operation deals with the volume-id values and the coordinates values. Refer Figure 5.1(h) for details of the for volume module.

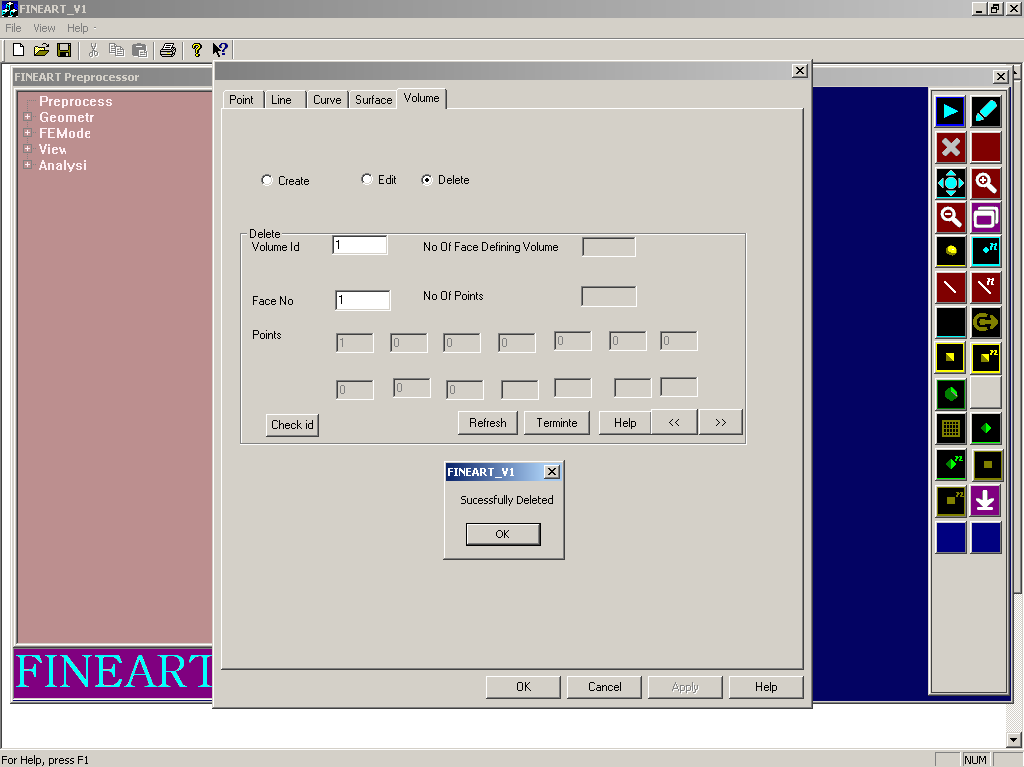


Fig 5.1[h] Geometry-Volume

FE-Model: Meshing

This module consists of meshing details. In this module add operations that can be performed. The operation is performed for 1-dimensions, 2-dimensions and 3- dimensions. When the user clicks for 2-dimensions the options will be displayed correspondingly as quadrilateral and triangular .For 1-D operation deal with the beam-id values and for 2-D operations deal with the surface- id values. Refer Figure 5.1(i) for details of the meshing module.

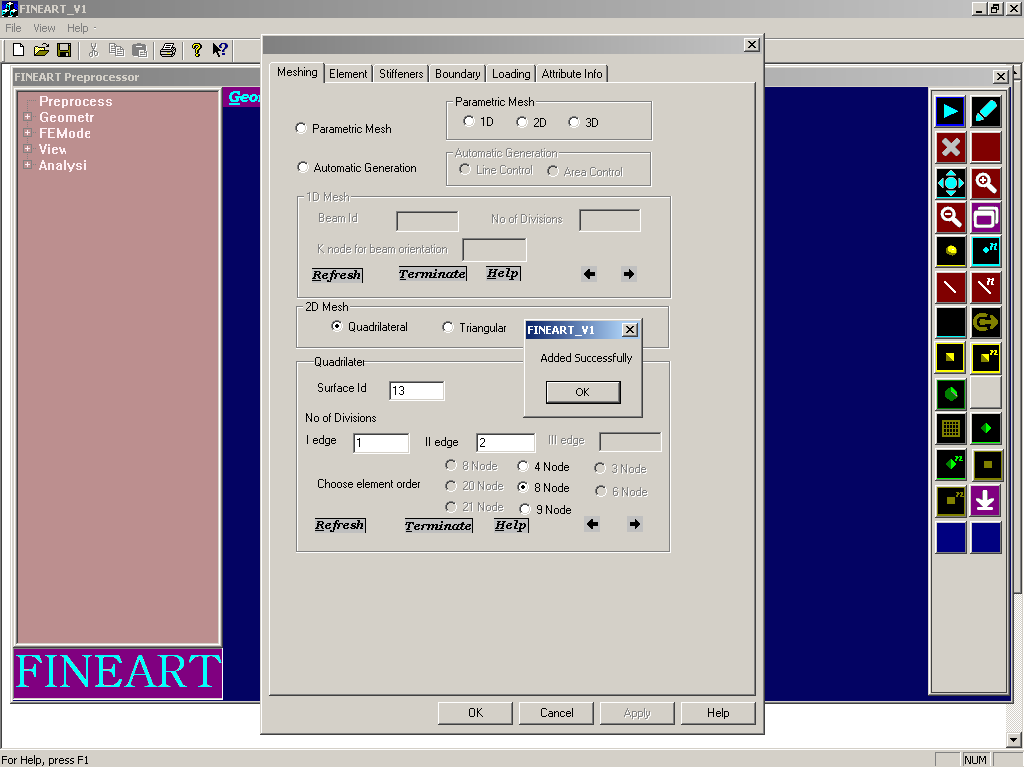


Fig 5.1[i] FE Model-Meshing FE-Model: Element

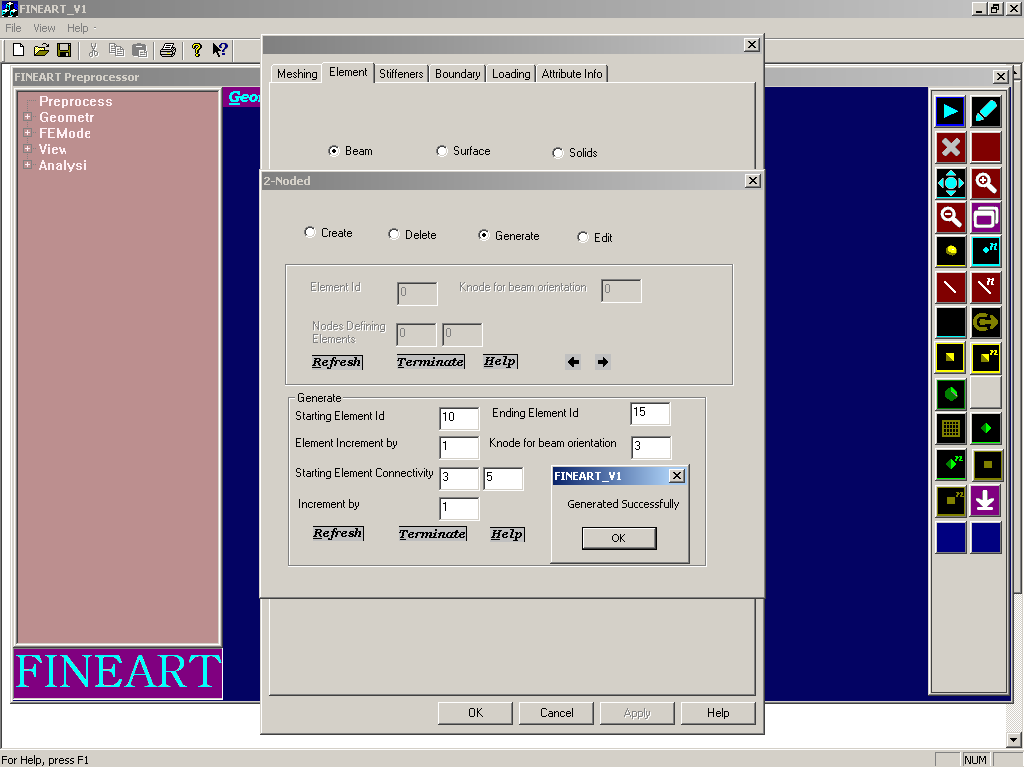


Fig 5.1(j) FE Model-Element

This module consists of element details. In this module there are three options such as beam, surface and solid that can be performed. When the user clicks for beam, the options will be displayed correspondingly as 2-node and 3-node. When the user clicks for surface the options will be displayed correspondingly as 4-node, 8-node and 9-node when the user clicks for solid the options will be displayed correspondingly as 20-node and 21-node .All these there are four major operations that can be performed. They are create, edit, generate and delete. All these operation deal with the element-id values. Refer Figure 5.1(j) for details of the element module.

FE Model-Boundary:

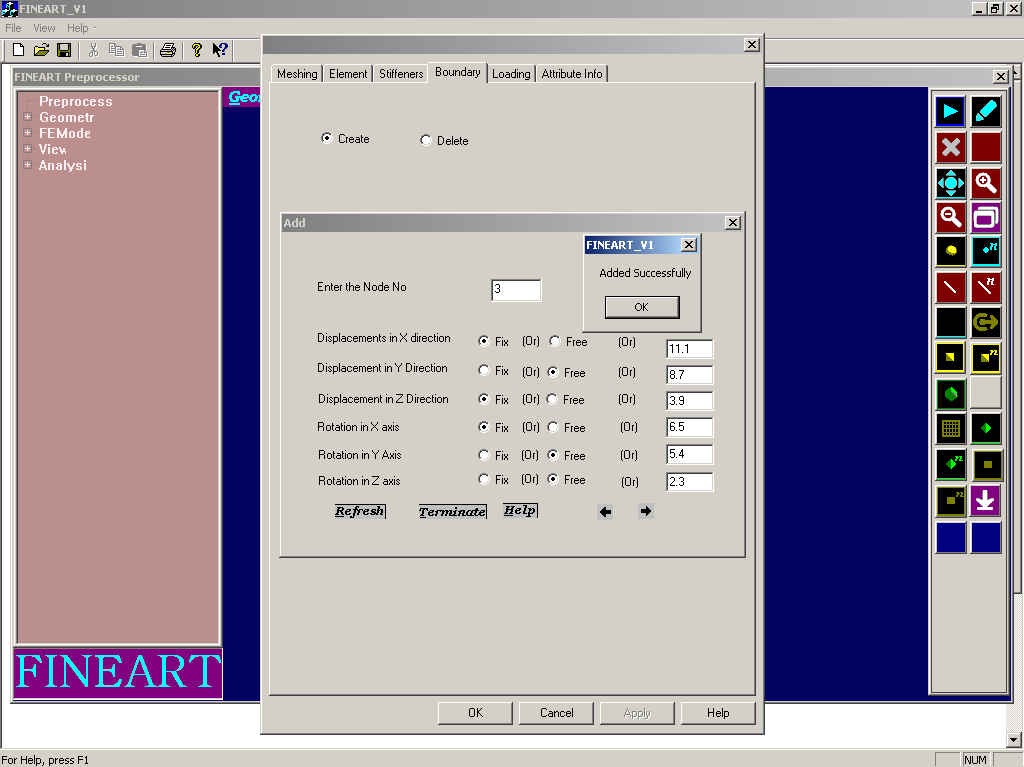


Fig 5.1(k) F E Model-Boundary

This module consists of boundary details. In this module there are two major operations that can be performed. They are create boundary and edit boundary. All these operation deal with the node number. Refer Figure 5.1(k) for details of the boundary module.

FE Model: Loading

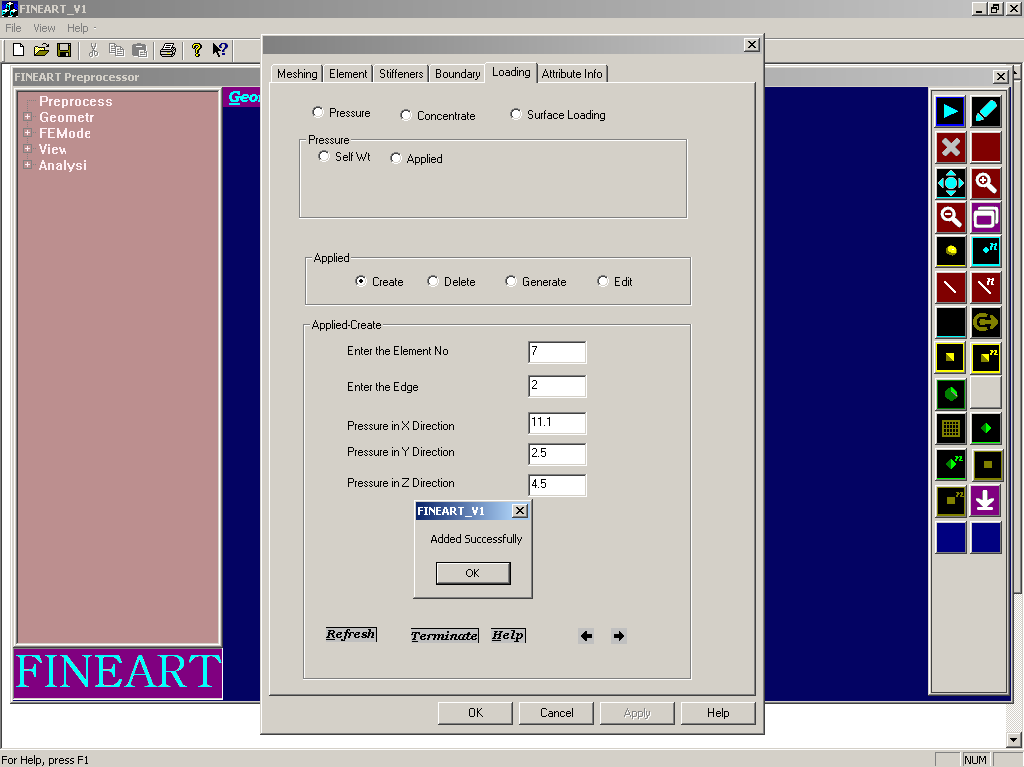


Figure 5.1(l) FE Model-Loading

This module consists of loading details. In this module there are three options

.They are pressure, concentrate and surface loading .two major operations that can be performed. When the user clicks for pressure the options will be displayed correspondingly as selfwt and applied .They are create boundary and edit boundary. All these there are four major operations that can be performed. They are create, edit, generate and delete. All these operation deal with the element number values. Refer Figure 5.1(l) for details of the loading module.

Analysis: Static

This module consists of static details. In this module add operations that can be performed .The static details that can be stored as either with or without error estimates. The values such as smoothing type and convergence tolerance that can be stored. Refer Figure 5.1(m) for details of the static module.

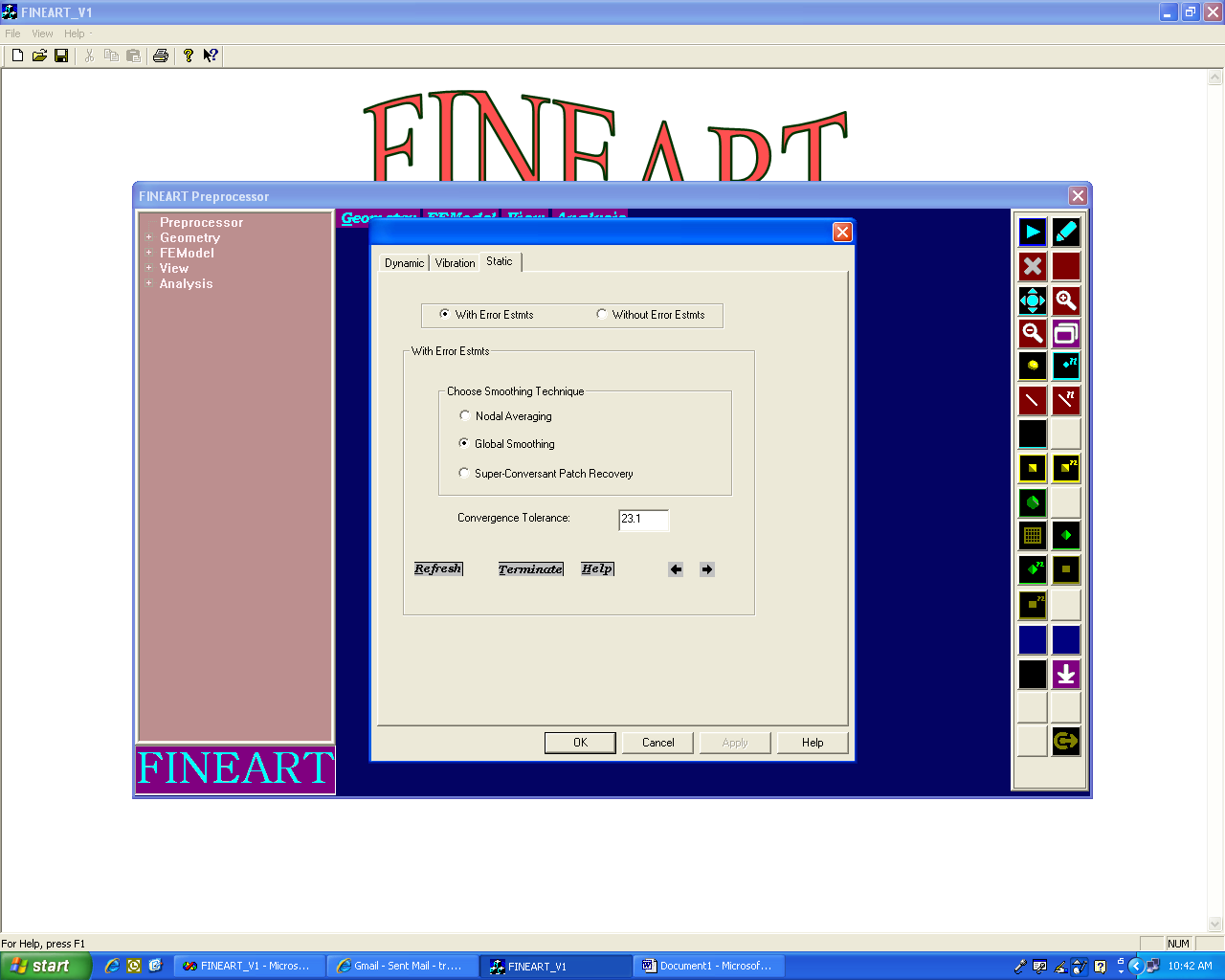


Figure 5.1(m) Analysis-Static Analysis: Dynamic

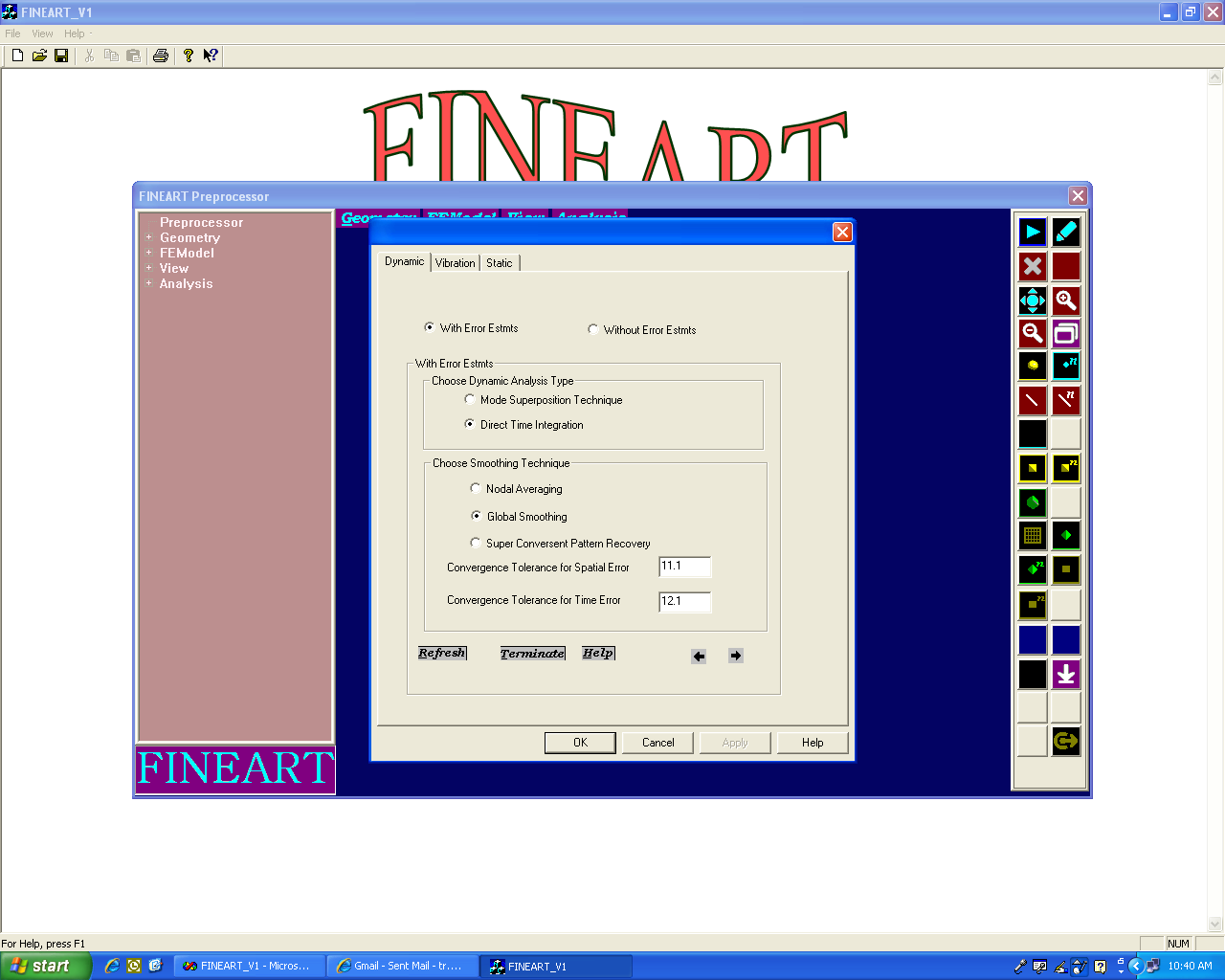


Figure 5.1[n] Analysis-Dynamic

This module consists of dynamic details. In this module add operations that can be performed. The dynamic details that can be stored as either with or without error estimates. The values such as dynamic analysis type, smoothing type, spatial error and time error that can be stored. Refer Figure 5.1(n) for details of the dynamic module.

Analysis: Vibration

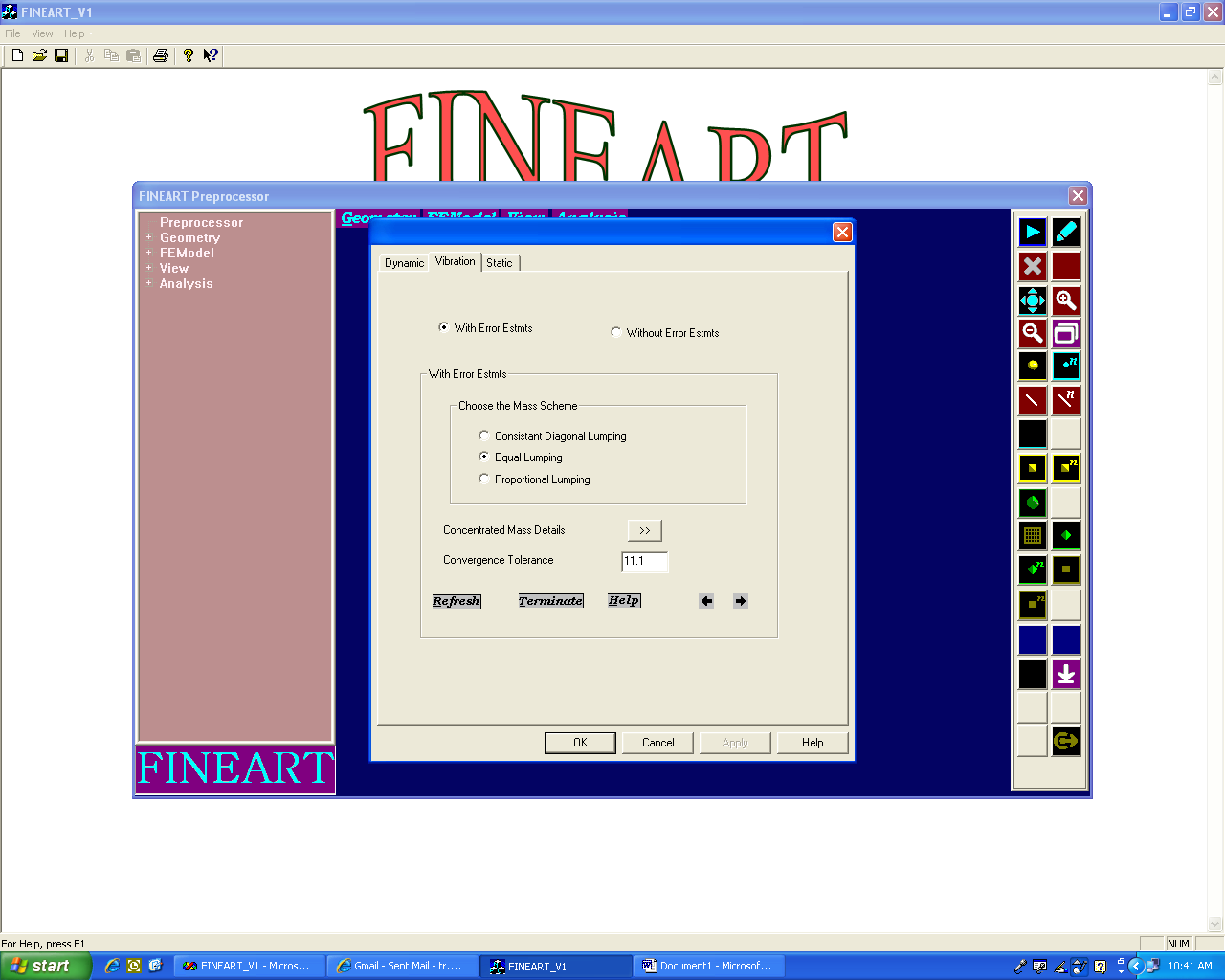


Figure 5.1(o) Analysis-Vibration

This module consists of vibration details. In this module add operations that can be performed. The vibration details that can be stored as either with or without error estimates. The values such as mass scheme and convergence tolerance that can be stored. Refer Figure 5.1(o) for details of the vibration module.

# Back End Design

## LOGIN TABLE

This table consists of login table. In this table there are two values that can be stored. They are User Name and Password. The User Name can be set as Primary Key.

|  |  |  |  |
| --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | PRIMARY KEY | DESCRIPTION |
| User Name | Text | Yes | Username given for the  administrator |
| Password | Text | Yes | Password given for the administrator |

## POINT TABLE

|  |  |  |  |
| --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | PRIMARY KEY | DESCRIPTION |
| TIhdis table consist  They are id, x, y, | sNoufmpboeirnt table. In th  and z. The id can be se | iYs etsable there are fou  t as Primary Key*.* | rIdvagliuveesntfhoart tchaenpobientstored. |
| X | Number | No | x-coordinate given for the  point |
| Y | Number | No | y-coordinate given for the  point |
| Z | Number | No | z-coordinate given for the  point |

**LINE TABLE**

This table consists of line table. In this table there are three values that can be stored. They are id, spid and epid. The id can be set as Primary Key.

|  |  |  |  |
| --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | PRIMARY KEY | DESCRIPTION |
| Id | Number | Yes | Id given for the line |
| spid | Number | No | Starting point given for the line |
| epid | Number | No | Ending point given for the  line |

## CURVE TABLE

This table consists of curve table. In this table there are five values that can be stored. They are id, spid, epid, curvature and radius. The id can be set as Primary Key.

|  |  |  |  |
| --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | PRIMARY KEY | DESCRIPTION |
| id | Number | Yes | Id given for the curve |
| spid | Number | No | Starting point given for the Curve |
| epid | Number | No | Ending point given for the  Curve |
| curvature | Number | No | Curvature value given for the curve |
| radius | Number | No | radius value given for the curve |

## SURFACE TABLE:

This table consists of surface table. In this table there are twenty two values that can be stored. They are id,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,P12, p13, p14, p15, p16, p17, p18, p19, and p20 and p21.The id can be set as Primary Key.

|  |  |  |  |
| --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | PRIMARY KEY | DESCRIPTION |
| Id | Number | Yes | Id given for the surface |
| P1 | Number | No | point given for the surface |
| P2 | Number | No | point given for the surface |
| P3 | Number | No | point given for the surface |
| P4 | Number | No | point given for the surface |
| P5 | Number | No | point given for the surface |
| P6 | Number | No | point given for the surface |
| P7 | Number | No | point given for the surface |
| P8 | Number | No | point given for the surface |
| P9 | Number | No | point given for the surface |
| P10 | Number | No | point given for the surface |
| P11 | Number | No | point given for the surface |
| P12 | Number | No | point given for the surface |
| P13 | Number | No | point given for the surface |
| P14 | Number | No | point given for the surface |
| P15 | Number | No | point given for the surface |
| P16 | Number | No | point given for the surface |
| P17 | Number | No | point given for the surface |
| P18 | Number | No | point given for the surface |
| P19 | Number | No | point given for the surface |
| P20 | Number | No | point given for the surface |
| P21 | Number | No | point given for the surface |

|  |  |  |  |
| --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | PRIMARY KEY | DESCRIBTION |
| id | Number | Yes | Id given for volume |
| fno | Number | No | Face no given for volume |
| P1 | Number | No | point given for the Volume |
| P2 | Number | No | point given for the Volume |
| P3 | Number | No | point given for the Volume |
| P4 | Number | No | point given for the Volume |
| P5 | Number | No | point given for the Volume |
| P6 | Number | No | point given for the Volume |
| P7 | Number | No | point given for the Volume |
| P8 | Number | No | point given for the Volume |
| P9 | Number | No | point given for the Volume |
| P10 | Number | No | point given for the Volume |

## VOLUME TABLE:

This table consists of volume table. In this table there are twelve values that can be stored. They are id,fno,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10.The id can be set as Primary Key.

## ELEMENT TABLE:

|  |  |  |  |
| --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | Primary key | DESCRIBTION |
| id | Number | Yes | Id given for 3node |
| k-node | Number | No | K-node for beam  orientation |
| **2**N**N**o**o**d**d**e**e**1**:** | Number | No | Node defining elements |
| Node2 | Number | No | Node defining elements |
| Node3This table con | sNisutsmobfer2-node table. | INnothis table the | rNe oadree fdoeufrinvinagluelsemtheant tcsan be |

stored. They are id, k-node, node1 and node2. The id can be set as Primary Key.

|  |  |  |  |
| --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | PRIMARY KEY | DESCRIBTION |
| id | Number | Yes | Id given for 2node |
| k-node | Number | No | K-node for beam orientation |
| Node1 | Number | No | Node defining elements |
| Node2 | Number | No | Node defining elements |

## 3Node:

This table consists of 3-node table. In this table there are five values that can be stored. They are id, k-node, node1, node2 and node3. The id can be set as Primary Key.

## 4Node:

This table consists of 4-node table. In this table there are six values that can be stored. They are id, k-node, node1, node2, node3 and node4. The id can be set as Primary Key.

|  |  |  |  |
| --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | Primary key | DESCRIBTION |
| id | Number | Yes | Id given for 4node |
| k-node | Number | No | K-node for beam orientation |
| Node1 | Number | No | Node defining elements |
| Node2 | Number | No | Node defining elements |
| Node3 | Number | No | Node defining elements |
| Node4 | Number | No | Node defining elements |

## DYNAM IC TABLE:

This table consists of dynamic table. In this table there are five values that can be stored. They are dynamic type, analysis type, smoothing type, time error and spatial error.

|  |  |  |
| --- | --- | --- |
| FIELD NAME | DATA TYPE | DESCRIBTION |
| dt | Text | Dynamic type |
| at | Text | Analysis type |
| st | Text | Smoothing type |
| se | Number | Spatial error |
| te | Number | Time error |

## STATIC TABLE:

This table consists of static table. In this table there are three values that can be stored. They are static type, smoothing type and convergence tolerance.

|  |  |  |
| --- | --- | --- |
| FIELD NAME | DATA TYPE | DESCRIBTION |
| static | Text | static type |
| st | Text | Smoothing type |
| ct | Number | Convergence tolerance |

## VIBRATION TABLE:

This table consists of vibration table. In this table there are three values that can be stored. They are vibration type, mass scheme and convergence tolerance.

|  |  |  |
| --- | --- | --- |
| FIELD NAME | DATA TYPE | DESCRIBTION |

|  |  |  |
| --- | --- | --- |
| vt | Text | vibration type |
| ms | Text | Mass scheme |
| ct | Number | Convergence tolerance |

## LOADING: APPLIED TABLE

This table consists of applied table. In this table there are five values that can be stored. They are element number, node number and pressure in x, y, z direction.

|  |  |  |
| --- | --- | --- |
| FIELD NAME | DATA TYPE | DESCRIBTION |
| eno | Text | Element number |
| nno | Text | Node number |
| px | Number | Pressure in x-direction |
| py | Number | Pressure in y-direction |
| pz | Number | Pressure in z-direction |

***CODING***

# Coding

## Sample codes

// Point\_page.cpp: implementation file #include "stdafx.h"

#include "FINEART\_V1.h" #include "Point\_page.h" #include "math.h"

#include "pdata.h" #ifdef \_DEBUG

#define new DEBUG\_NEW #undef THIS\_FILE

static char THIS\_FILE [] = FILE ; #endif

void Point\_page::OnBtnPcrefresh()

{

// TODO: Add your control notification handler code here

int id,i;

double dx,dy,dz; bool found=false;

CString name,sx,sy,sz; GetDlgItemText(IDC\_EDIT\_PCID,name); GetDlgItemText(IDC\_EDIT\_PCX,sx); GetDlgItemText(IDC\_EDIT\_PCY,sy);

GetDlgItemText(IDC\_EDIT\_PCY,sz); id=atoi(name);

if(name!="" && sx!="" && sy!="" && sz!="")

{

if(name!="0" && sx!="0" && sy!="0" && sz!="0")

{

if(id>=0)

{

while(!rs->IsEOF())

{

if(rs->m\_id==id)

{

MessageBox("id already exist"); i=0;

SetDlgItemText(IDC\_EDIT\_PCID,""); found=true;

}

if(found) break;

rs->MoveNext();

}

if(i!=0)

{

dx=atof(sx); dy=atof(sy); dz=atoi(sz);

rs->AddNew(); UpdateData(true); if(rs->CanUpdate())

{

rs->m\_id=id; rs->m\_x=dx; rs->m\_y=dy; rs->m\_z=dz; rs->Update();

}

rs->Requery();

UpdateData(false);

MessageBox("Added Successfully"); GetDlgItem(IDC\_EDIT\_PCID)->SetWindowText(""); GetDlgItem(IDC\_EDIT\_PCX)->SetWindowText(""); GetDlgItem(IDC\_EDIT\_PCY)->SetWindowText(""); GetDlgItem(IDC\_EDIT\_PCZ)->SetWindowText("");

found=true;

}

}

}

else

MessageBox("Not Acceptable Value");

}

else

{

MessageBox("Check Your Value");

}

}

void Point\_page::OnBtnPcback()

{

// TODO: Add your control notification handler code here int id;

double x,y; if(!rs->IsBOF())

{

rs->MovePrev(); id=rs->m\_id; x=rs->m\_x; y=rs->m\_y;

CString sid,sx,sy; sid.Format("%ld",id);

sx.Format("%lf",x);

sy.Format("%lf",y); SetDlgItemText(IDC\_EDIT\_PCID,sid); SetDlgItemText(IDC\_EDIT\_PCX,sx); SetDlgItemText(IDC\_EDIT\_PCY,sy);

}

else

MessageBox("No Record Found");

}

void Point\_page::OnBtnPcnext()

{

// TODO: Add your control notification handler code here int id;

double x,y; if(!rs->IsEOF())

{

rs->MoveNext(); id=rs->m\_id; x=rs->m\_x;

y=rs->m\_y;

CString sid,sx,sy;

sid.Format("%ld",id);

sx.Format("%lf",x);

sy.Format("%lf",y); SetDlgItemText(IDC\_EDIT\_PCID,sid); SetDlgItemText(IDC\_EDIT\_PCX,sx); SetDlgItemText(IDC\_EDIT\_PCY,sy);

}

else

MessageBox("No Record Found");

}

void Point\_page::OnBtnPdrefresh()

{

CString name; int id;

bool found=false; GetDlgItemText(IDC\_EDIT\_PD\_ID,name); if(name!="")

{

if(name!="0")

{

id=atoi(name); while(!rs->IsEOF())

{

if(rs->m\_id==id)

{

rs->Delete(); found=true;

}

if(found) break;

rs->MoveNext();

}

MessageBox("Record is sucessfully deleted"); GetDlgItem(IDC\_EDIT\_PD\_ID)->SetWindowText("");

}

else

MessageBox("Not Acceptable Value");

}

else

MessageBox("Check Your Value");

// TODO: Add your control notification handler code here

}

void Point\_page::OnBtnPgrefresh()

{

// TODO: Add your control notification handler code here bool found=false;

CString sid,eid,sx1,sx2,sy1,sy2,sz1,sz2,si,s,sx,sy,sz; CString s1,s2,s3;

int siid,eiid,ii,i;

double ix1,ix2,iy1,iy2,iz1,iz2,px,py,pz; int j=0,n=1,div;

GetDlgItemText(IDC\_EDIT\_PG\_START,sid); GetDlgItemText(IDC\_EDIT\_PG\_END,eid); GetDlgItemText(IDC\_EDIT\_GPSTART\_X,sx1); GetDlgItemText(IDC\_EDIT\_GPSTART\_X,sx); GetDlgItemText(IDC\_EDIT\_GPEND\_X,sx2); GetDlgItemText(IDC\_EDIT\_GPSTART\_Y,sy1); GetDlgItemText(IDC\_EDIT\_GPSTART\_Y,sy); GetDlgItemText(IDC\_EDIT\_GPEND\_Y,sy2); GetDlgItemText(IDC\_EDIT\_PG\_INC,si); GetDlgItemText(IDC\_EDIT\_GPSTART\_Z,sz1); GetDlgItemText(IDC\_EDIT\_GPEND\_Z,sz2); GetDlgItemText(IDC\_EDIT\_GPSTART\_Z,sz);

if((sid!="") && (eid!="") && (sx1!="") && (sy1!="") && (sx2!="") &&(sy2!="") && (sz1!="") && (sz2!="") && (si!=""))

{

if((sid!="0") && (eid!="0") && (sx1!="0") && (sy1!="0") && (sx2!="0") &&

(sy2!="0") && (sz1!="0") && (sz2!="0") && (si!="0"))

{ j++;

siid=atoi(sid); eiid=atoi(eid); ix1=atof(sx1); ix2=atof(sx2); iy1=atof(sy1); iy2=atof(sy2); iz1=atof(sz1); iz2=atof(sz2); ii=atoi(si); px=atof(sx);

py=atof(sy); pz=atof(sz); n=(eiid-siid)/ii+n; div=(eiid-siid)/ii;

while(!rs->IsEOF())

{

if(rs->m\_id==eiid || rs->m\_id==siid)

{

MessageBox("Id Already Exist"); found=true; SetDlgItemText(IDC\_EDIT\_PG\_START,""); SetDlgItemText(IDC\_EDIT\_PG\_END,""); i=0;

}

if(found) break;

rs->MoveNext();

}

if(i!=0)

{

for(j=1;j<=n;j++)

{

rs->AddNew(); if(rs->CanUpdate())

{

rs->m\_id=siid;

rs->m\_x=fabs(ix1); rs->m\_y=fabs(iy1); rs->m\_z=fabs(iz1); rs->Update();

}

rs->Requery(); UpdateData(false); found=true; siid=siid+ii; if(j==(n-1))

{

ix1=ix2; iy1=iy2; iz1=iz2;

}

else

{

ix1=fabsl((ix2-px)/div\*j+px); iy1=fabsl((iy2-py)/div\*j+py); iz1=fabsl((iz2-pz)/div\*j+pz);

}

}

MessageBox(" Generated successfully"); SetDlgItemText(IDC\_EDIT\_PG\_START,sid); SetDlgItemText(IDC\_EDIT\_PG\_END,eid); SetDlgItemText(IDC\_EDIT\_GPSTART\_X,sx1); SetDlgItemText(IDC\_EDIT\_GPSTART\_X,sx); SetDlgItemText(IDC\_EDIT\_GPEND\_X,sx2); SetDlgItemText(IDC\_EDIT\_GPSTART\_Y,sy1); SetDlgItemText(IDC\_EDIT\_GPSTART\_Y,sy); SetDlgItemText(IDC\_EDIT\_GPEND\_Y,sy2); SetDlgItemText(IDC\_EDIT\_PG\_INC,si); SetDlgItemText(IDC\_EDIT\_GPSTART\_Z,sz1); SetDlgItemText(IDC\_EDIT\_GPEND\_Z,sz2); SetDlgItemText(IDC\_EDIT\_GPSTART\_Z,sz);

}

}

else

MessageBox("Not Acceptable Value");

}

else

MessageBox("Check Your Value");

}

void Point\_page::OnBtnPgback()

{

if(!rs->IsBOF())

{

int rid; double xx,yy;

rs->MovePrev(); rid=rs->m\_id; xx=rs->m\_x; yy=rs->m\_y; CString pid,px,py;

pid.Format("%ld",rid);

px.Format("%lf",xx);

py.Format("%lf",yy); SetDlgItemText(IDC\_EDIT\_PCID,pid); SetDlgItemText(IDC\_EDIT\_PCX,px); SetDlgItemText(IDC\_EDIT\_PCY,py);

}

else

MessageBox("No Record Found");

// TODO: Add your control notification handler code here

}

void Point\_page::OnBtnPgnext()

{

// TODO: Add your control notification handler code here int rid;

double xx,yy; if(!rs->IsEOF())

{

rs->MoveNext(); rid=rs->m\_id; xx=rs->m\_x; yy=rs->m\_y; CString pid,px,py;

pid.Format("%ld",rid);

px.Format("%lf",xx);

py.Format("%lf",yy); SetDlgItemText(IDC\_EDIT\_PCID,pid); SetDlgItemText(IDC\_EDIT\_PCX,px); SetDlgItemText(IDC\_EDIT\_PCY,py);

}

else

MessageBox("No Record Found");

}

void Point\_page::OnBtnPdback()

{

// TODO: Add your control notification handler code here int id;

if(!rs->IsBOF())

{

rs->MovePrev(); id=rs->m\_id; CString sid;

sid.Format("%ld",id); SetDlgItemText(IDC\_EDIT\_PD\_ID,sid);

}

else

MessageBox("No Record Found");

}

void Point\_page::OnBtnPdnext()

{

int id;

if(!rs->IsEOF())

{

rs->MoveNext(); id=rs->m\_id; CString sid;

sid.Format("%ld",id); SetDlgItemText(IDC\_EDIT\_PD\_ID,sid);

}

else

MessageBox("No Record Found");

}

void Point\_page::OnBtnPeback()

{

// TODO: Add your control notification handler code here int id;

double x,y; if(!rs->IsBOF())

{

rs->MovePrev(); id=rs->m\_id; x=rs->m\_x; y=rs->m\_y;

CString sid,sx,sy; sid.Format("%ld",id);

sx.Format("%lf",x);

sy.Format("%lf",y); SetDlgItemText(IDC\_EDIT\_PEID,sid); SetDlgItemText(IDC\_EDIT\_EPX,sx); SetDlgItemText(IDC\_EDIT\_EPY,sy);

}

else

MessageBox("No Record Found");

}

void Point\_page::OnBtnPenext()

{

// TODO: Add your control notification handler code here int id;

double x,y; if(!rs->IsEOF())

{

rs->MoveNext(); id=rs->m\_id; x=rs->m\_x;

y=rs->m\_y;

CString sid,sx,sy;

sid.Format("%ld",id);

sx.Format("%lf",x);

sy.Format("%lf",y); SetDlgItemText(IDC\_EDIT\_PEID,sid); SetDlgItemText(IDC\_EDIT\_EPX,sx); SetDlgItemText(IDC\_EDIT\_EPY,sy);

}

else

MessageBox("No Record Found");

}

void Point\_page::OnBtnPcrefresh2()

{

// TODO: Add your control notification handler code here bool found=false;

int id;

double dx,dy,dz;

CString name,sx,sy,sz;

GetDlgItemText(IDC\_EDIT\_PEID,name); GetDlgItemText(IDC\_EDIT\_EPX,sx); GetDlgItemText(IDC\_EDIT\_EPY,sy); GetDlgItemText(IDC\_EDIT\_EPZ,sz); if(name!="" && sx!="" && sy!="")

{

if(name!="0" && sx!="0" && sy!="0")

{

id=atoi(name); dx=atof(sx); dy=atof(sy); dz=atof(sz); while(!rs->IsEOF())

{

if(rs->m\_id==id)

{

rs->Edit();

rs->m\_x=dx; rs->m\_y=dy; rs->m\_z=dz;

UpdateData(TRUE); found=true;

if(rs->CanUpdate())

{

rs->Update();

}

}

if(found) break;

rs->MoveNext();

}

GetDlgItem(IDC\_EDIT\_PEID)->SetWindowText(""); GetDlgItem(IDC\_EDIT\_EPX)->SetWindowText("");

***TESTING***

## TESTING

Testing has become an integral part of any system or the project. Software testing is a critical element of software quality assurance and represents the ultimate of specification, design and coding. When software is developed, before it is given to the user it must be tested whether it is solving the purpose for which it is developed. This testing involved various types through which one can ensure the reliability of the software.

## Unit Testing

**Test Co**A**n**l**d**l**it**m**io**o**n**dules are put **T**th**e**r**s**o**t**u**i**g**n**h**pu**fo**t**rmal unit tests. **E**T**x**h**p**e **e**u**c**n**t**i**e**t**d**te**o**s**u**ts**tp**f**u**ol**t**low a valid**R**U**es**n**u**i**l**t**t Ch**T**e**e**c**s**k**t **f**P**o**l**r**an**lo**t**g**h**i**a**n**t designed toCocrorveecrt fuusnecrtionnamaleanadndstruMctauirnalscteresetsn to provide maxSimucucmess path coverage. Unit TestpaPslsawnsorhdave been created only for modules with business

functionality. The devel GUI check list prior to r

Empty

**b) Integration Testing**

Project name and title Fail required-warning

message

Integration testing is applied after the completion of unit testing. In this testing,

nFgaitlhe

thIneviralriedspeincptiuvte s–cwreaernisngusi

nmg.essage

oIpnerscohrarvecet beuesnervenraifmieed

elaenadsinpgastshweosrcdreen for testi

the interfaces between the modules are tested. This testing process should therefore concentrate on the detection of module interface errors by rigorously exercising these

interfaces. Integration testing is used in all modules.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Condition** | **Test input** | **Expected output** | **Result** |
| **Check for Geometry** Point,Line,Curve Surface and Volume **FE model**  Meshing ,Element Stiffeners , Boundary Loading and Attribute  **Analysis**  **c) Validation Testing**  Static, Dynamic and  Validation testing Vibration  not. Suppose, by mistake if  "Please check your user nam | New id | Added and generated  successfully | Success |
| Exist id(Add and  generate) | Id already exist-warning  message | Fail |
| Exist id(Delete and  update) | Deleted and updated  successfully | Success |
| Empty | Check your value-  warning message | Fail |
| Mismatch data type  is the process the inputs, | Not acceptable value-  warning message  whether the given input is va | Fail  lid or |
| Previous, next record  the user enters the inco  e and password" displaye | Display record  rrect data then warning me  d. This testing is used to va | success ssage  lidate |

the user entries, until correct data is entered.

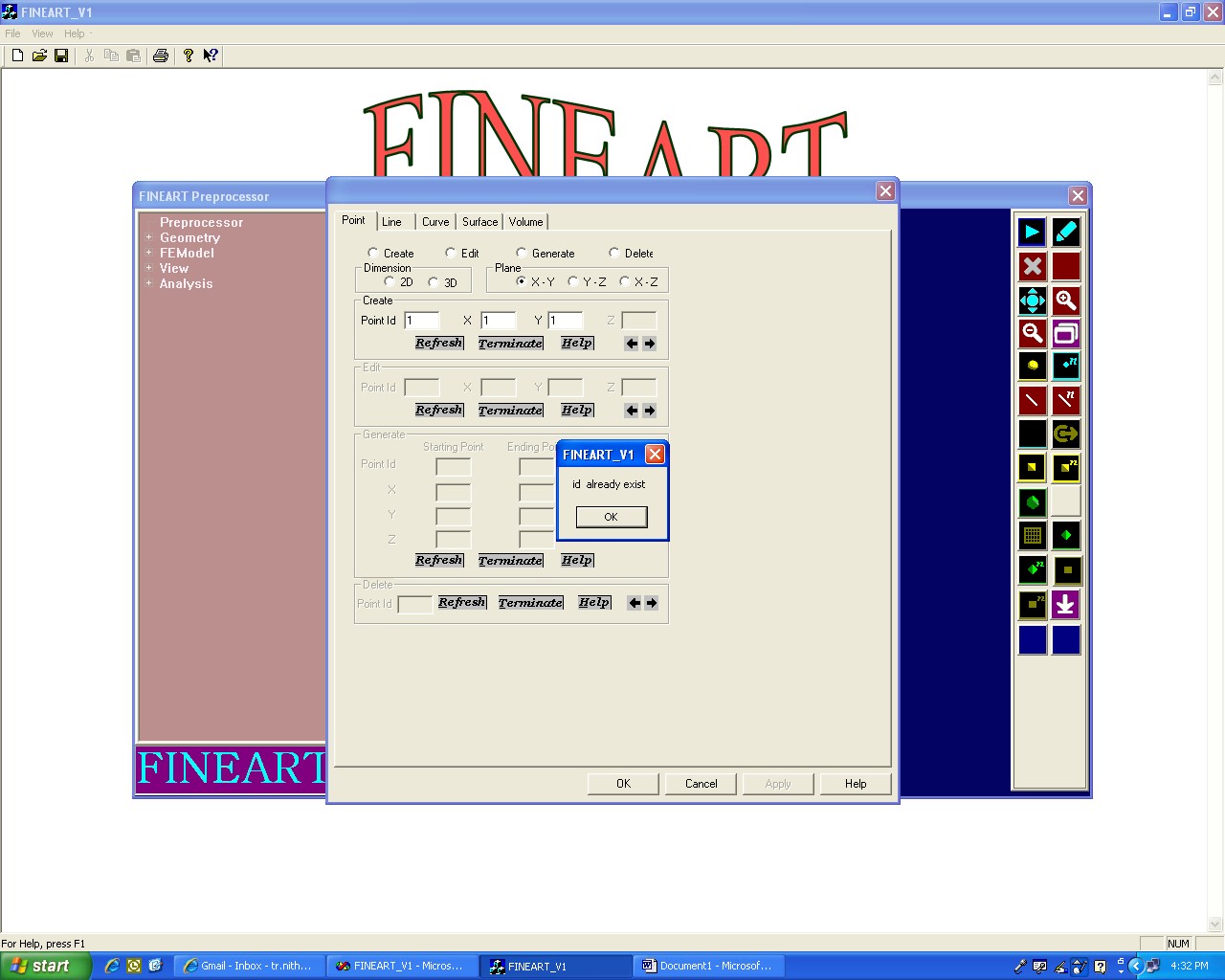


Figure 7.1

If the user enter already exist id, the message will be displayed as “id already exist”. Refer Fig 7.1

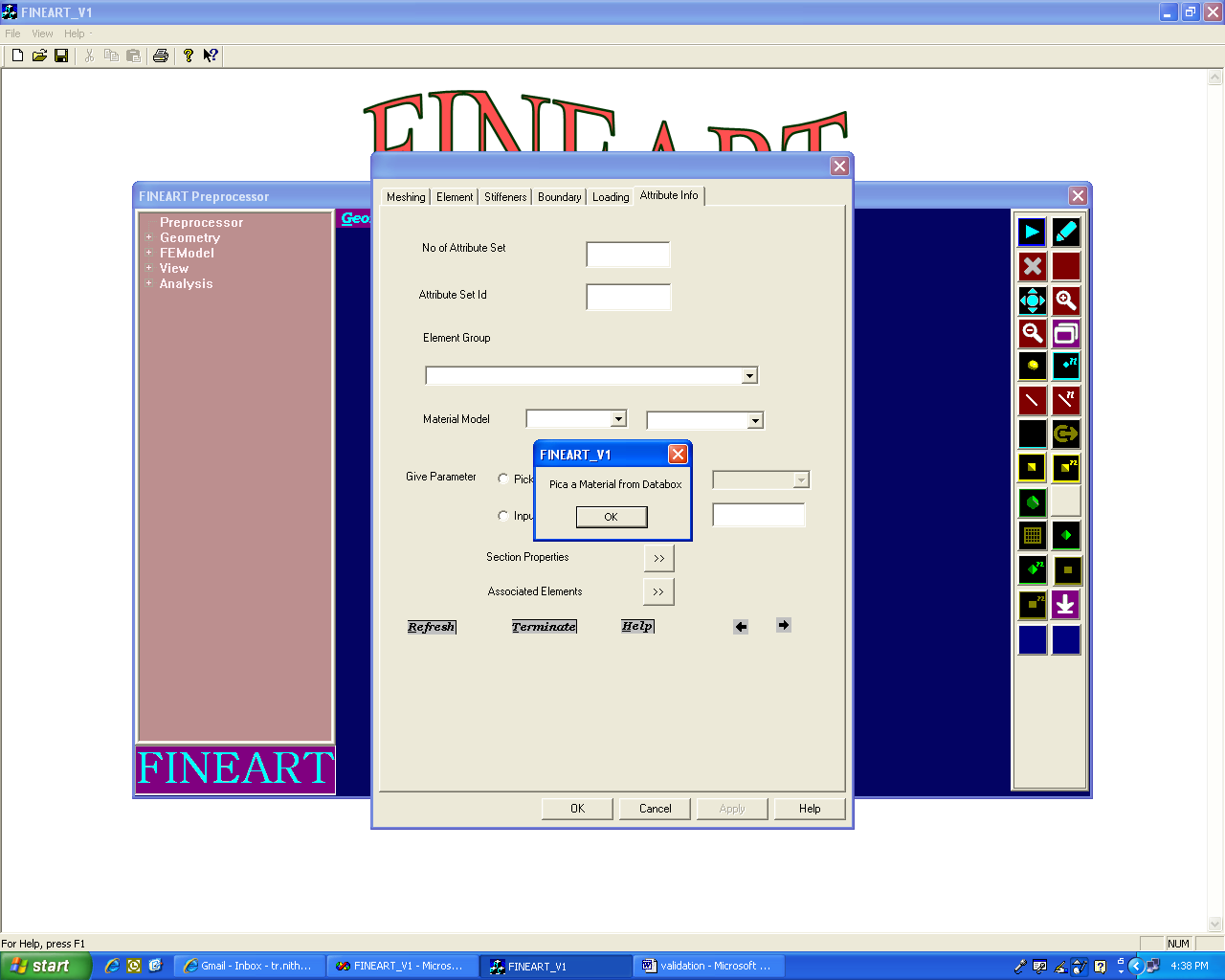


Figure 7.2

If the user discards input value, the message will be displayed as “pick a material from databox”.Refer Fig 7.2

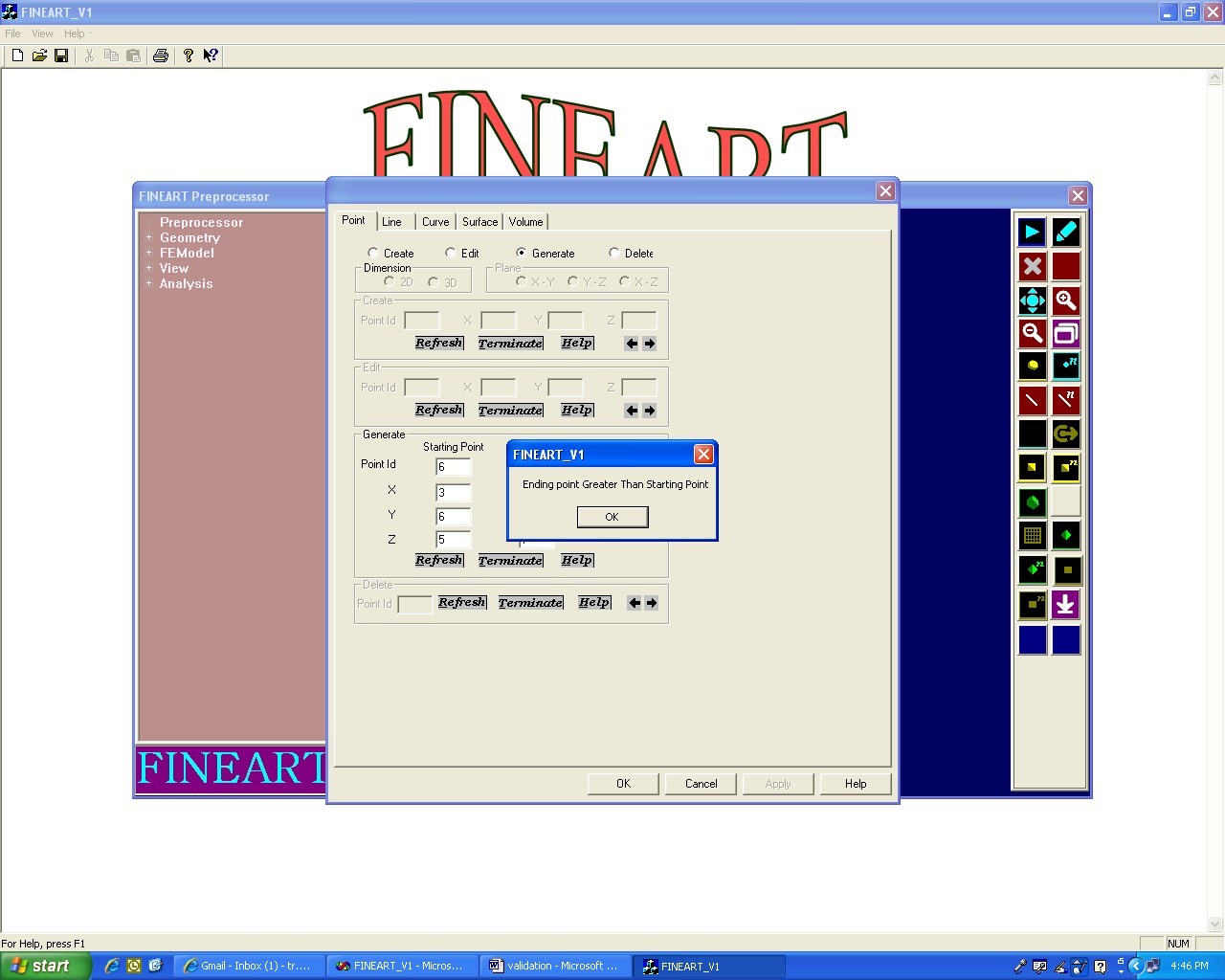


Figure 7.3

If the user enter starting id is greater than ending id, the message will be displayed as “Ending id is greater than starting id”.Refer Fig 7.3

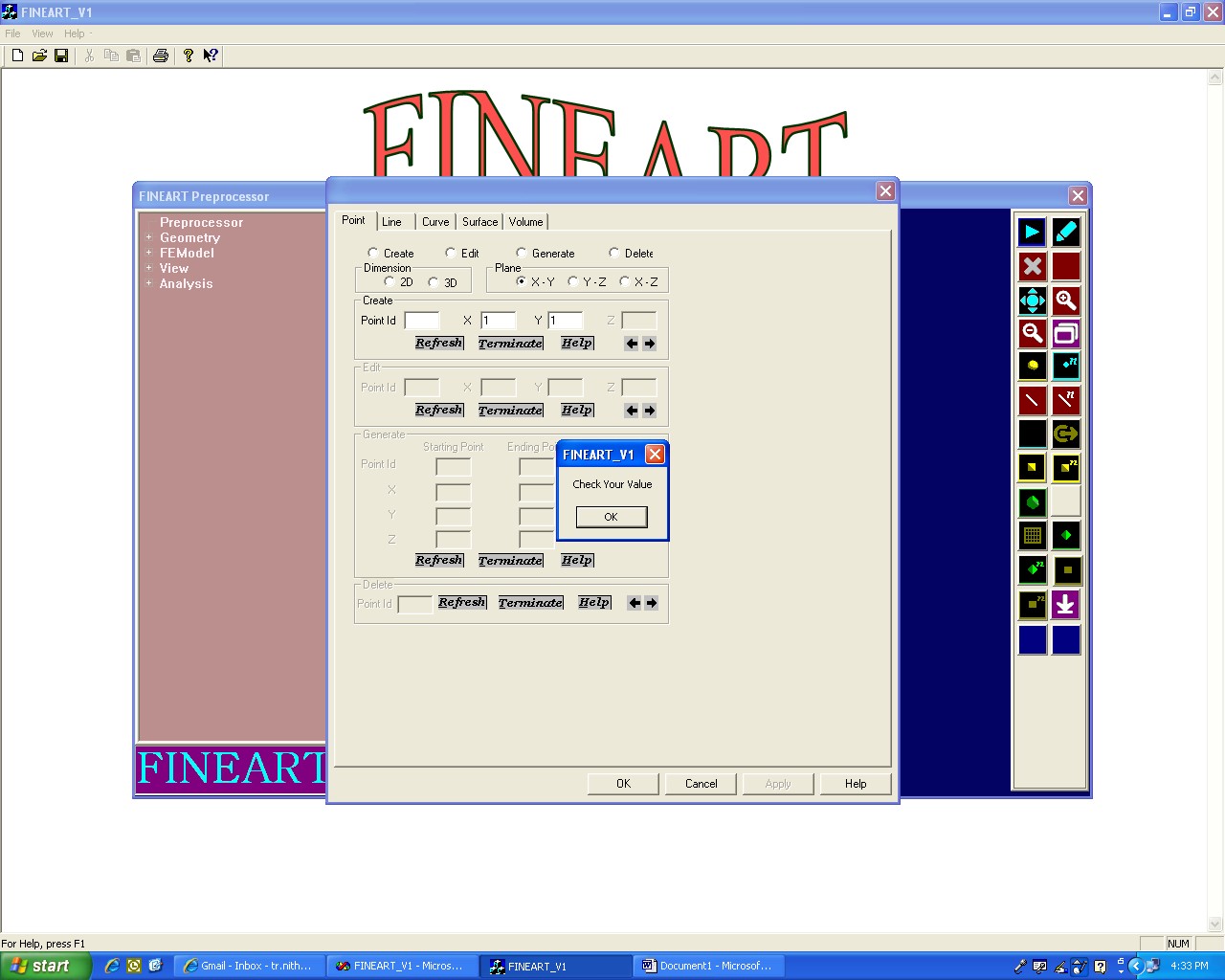


Figure 7.4

If the user discards input value, the message will be displayed as “Check your value”. Refer Fig 7.4

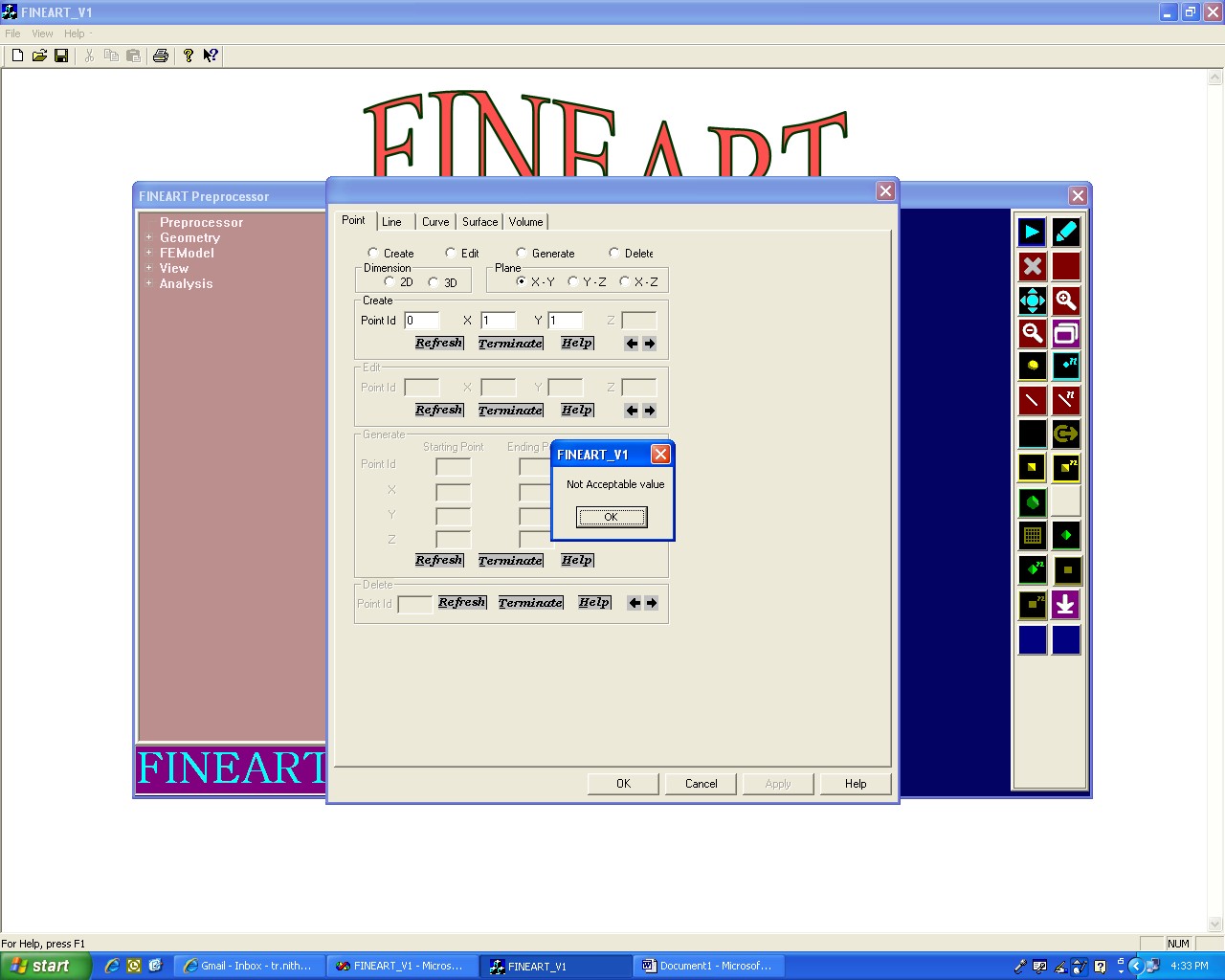


Figure 7.5

If the user enters mismatch data type, the message will be displayed as “Not acceptable value”. Refer Fig 7.5

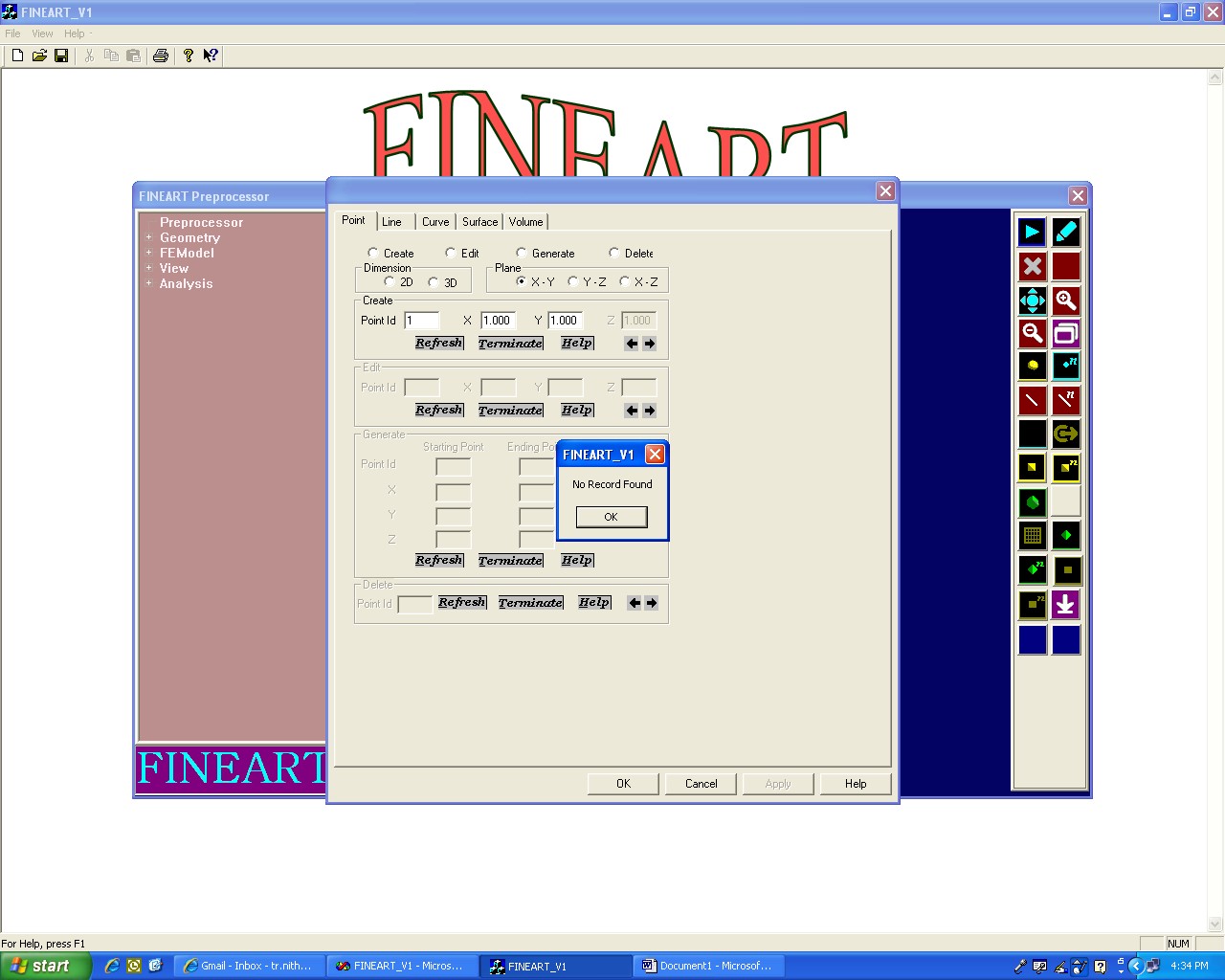


Figure 7.6

If the user click back or forward button when there is no record, the message will be displayed as ‘No Record Found”. Refer Fig 7.6

***IMPLEMENTATION***

## IMPLEMENTATION

* 1. Problems faced

When executing the queries such as update, insert I faced some problems with data type compatibility. After referring certain visual c++ books, the problem is resolved.

* 1. Lesson learnt

Through this project I learnt the following

* + - How to use database connections in efficient manner
    - Several operations in FINEART Software

***FUTURE PLAN***

Future Plan

In future I am planning to develop the project further by using open GL to draw geometric entities.

***BIBLIOGRAPHY***

10. BIBLIOGRAPHY

## Text Books:

1. David J.Krunglinkski, George Shepherd, and Scot Wingo “Programming Microsoft Visual C++”, Fifth Edition, Microsoft Corporation, USA
2. “Visual C++ 5”, Second Edition, BPB publications, Delhi

## Website Reference

* 1. <http://www.codeguru.com/cpp/controls/buttonctrl/> Article.php/c5191
  2. <http://msdn2.microsoft.com/>

En-us/library/t9fb9hww (VS.71).aspx

* 1. <http://www.codersource.net/mfc_ccombobox_example.html>
  2. <http://www.functionx.com/visualc/controls/combobox.htm>
  3. <http://www.functionx.com/visualc/collections/cobarray.htm>
  4. <http://www.functionx.com/visualc/>