REALTIME INTERACTIVE STRUCTURAL ANALYSIS AN OBJECT ORIENTED APPROACH

by

HOU-GION WUU

Master of Science in Structural Engineering

National Taiwan University, June 1994

Submitted to the Department of Civil and Environmental Engineering

in Partial Fulfillment of the Requirements for the Degree of

MASTER OF ENGINEERING

IN CIVIL AND ENVIRONMENTAL ENGINEERING

AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1998

Signature of Author ______________________________
Department of Civil and Environmental Engineering
May 7, 1998

Certified by ______________________________
Jerome J. Connor
Thesis Supervisor
Professor, Department of Civil and Environmental Engineering

Accepted by ______________________________
Joseph M. Sussman
Chairman, Department Committee on Graduate Studies

Realtime Interactive Structural Analysis
An Object Oriented Approach

by
Hou-Gion Wuu

Submitted to the Department of Civil and Environmental Engineering on May 7, 1998, in partial fulfillment of the requirements for the degree of Master of Engineering in Civil and Environmental Engineering at the Massachusetts Institute of Technology

Abstract

A realtime interactive structural analysis tool for educational purpose can be implemented due to the improvement of the computational ability of the microprocessor and software technology.

In general, a structural analysis tool should provide the required information for structural analysis. The information includes the geometry and material properties of the structure, the deformation and the forces in the member, and the validity of the assumptions. An interactive graphical user interface is the most effective way for users to understand the behavior of the structures.

Object oriented technology emphasizes the attributes and the behaviors of the object, and the relationships between objects. It can be used for modeling the structure and the architecture of the software application.

To meet different requirements, a layered architecture is adopted here. This architecture includes a finite element kernel, a basic interface, and a graphical user interface. The graphical user interface allows one to build the model of the structure by following the same approach he uses to build the real model. It provides the easiest way to build the model intuitively in a shortest time.

Thesis Supervisor: Professor Jerome J. Connor

Title: Professor of Civil and Environmental Engineering

Acknowledgments

I would like to thank Professor Jerome J. Connor, my thesis advisor, for his encouragement and guidance. I appreciate the enlightenment he has given to me in structural analysis and design.

Special thanks to Professor Hiroshi Sakuta for his crucial comments and directions in computer graphics and graphical user interface design.

I would like to thank Professor Hong-Ki Hong, Professor Jenn-Chuan Chen, and Professor Tai-Ming Parng for their recommendations for me to study in MIT.

I would like to express my gratitude to my sponsor, Mr. Tian-Sheng Yang, for giving me this scholarship to study abroad.

Thanks also to the members in my Smart Technologies Group, Bernard, Lisa, Maria, and Simonetta. I am very glad to work with them in the High Performance Structure project.

Finally, I would like to send my deepest thanks to my family for their encouragement and perpetual support.

Table of Contents

Abstract 3
Acknowledgments 5
Table of Contents 7
Table of Figures 9
Table of Code Examples 11
1. Introduction 13
  1.1 Realtime Interactivity 13
  1.2 Object Oriented Approach 14
  1.3 Requirements 15
  1.4 Layered Architecture 16
    1.4.1 Functionality of FEM Kernel 17
    1.4.2 Functionality of Basic Interface 17
    1.4.3 Functionality of Graphical User Interface 18
  1.5 Implementation 18
2. Finite Element Modeling of Structure 21
  2.1 Analysis 21
  2.2 Design 23
    2.2.1 Element 24
3. Basic Interface 27
  3.1 Analysis 27
  3.2 Design 28
4. Interactive Graphical User Interface 31
  4.1 Analysis 31
  4.2 Design 32
    4.2.1 Layout and Graphical Components 32
    4.2.2 Switching System 34
    4.2.3 Event Handling 35
    4.2.4 Data Transfer Mechanism 37

5. Conclusions 39
6. Future Work 41
7. References 43
8. Appendices 45
Appendix A 46
Appendix B 47

Table of Figures

Fig. 1: Interactivity 13
Fig. 2: Object and identity. 14
Fig. 3: Objects and class. 14
Fig. 4: Superclass and subclass. 15
Fig. 5: Layered architecture for users and developers. 16
Fig. 6: Structural frame. 21
Fig. 7: Local coordinate system for 1-D element 22
Fig. 8: Classification. 22
Fig. 9: Procedure of solving the problem by finite element method. 23
Fig. 10: Data access. 27
Fig. 11: Interface between objects. 28
Fig. 12: Relationships between basic interface and other interfaces. 29
Fig. 13: Complexity of user interface. 31
Fig. 14: Components in the graphical user interface. 33
Fig. 15: Switching system. 35
Fig. 16: Event handling. 36
Fig. 17: Choice and the corresponding process. 36
Fig. 18: Data transfer mechanism. 37
Fig. 19: Class relationships. 46

Table of Code Examples

C.E. 1: Class `Beam`, `Truss`, and `Element`. 24

C.E. 2: Procedures of calculating stiffness matrix. 25

C.E. 3: Class `Beam`, `Truss`, and `Element`. 25

C.E. 4: Association links between FEM kernel and its interface. 28

1. Introduction

There are many structural analysis tools, but generally it takes a long time for users to learn how to use the applications, and developers can not reuse the source codes when they want to extend and enhance the functions of the tool. Even if the developers have the source codes, it is difficult for them to understand and modify the codes. The other issue is that the tool is not sufficiently interactive for analysis. When users change the loads or makeup of the frame, they have to execute another command to rerun the program in order to obtain the solution. Our objective is to use an object oriented approach to build a structural analysis tool that provides realtime response, i.e., that computes the solution in a continuous manner as the input is being changed. An object oriented approach also enables developers to extend and enhance the application easily. Gajewski [1], Arruda, Landau, and Ebecken [2] discussed about the advantages of using object oriented approach and developed the graphical environment for structural analysis.

1.1 Realtime Interactivity

By "Realtime Interactivity", we mean the user can obtain the response immediately after a modification is made to the structure. A realtime interactive environment can help beginners to obtain a better understanding of the behavior of the structure. Users can immediately see the relationships between the behavior of the system and the imposed variations. For a simple linear static system, people can learn, imagine or even predict the behavior of the system after a few trials. For a complex, nonlinear, or dynamic system, it is difficult for people to obtain a sense of the behavior of the system without the aid of the tool.

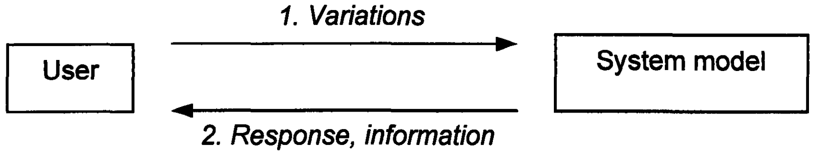


Fig. 1: Interactivity

Figure 1 shows the interaction between the user and the system model. An interactive environment not only can show the response to the variations quickly, it can also provide the necessary information to the users. For example, the process of design involves many assumptions, and sometimes an engineer may use the wrong assumption without perceiving it. In this case, an interactive analysis tool can generate a warning to the user.

1.2 Object Oriented Approach

As shown in Fig. 2, an object that is an identity in the world has attributes and operations. An object means that it is an identity in the world. Objects with the same attributes and operations can be grouped into a class [3], as shown in Fig. 3.

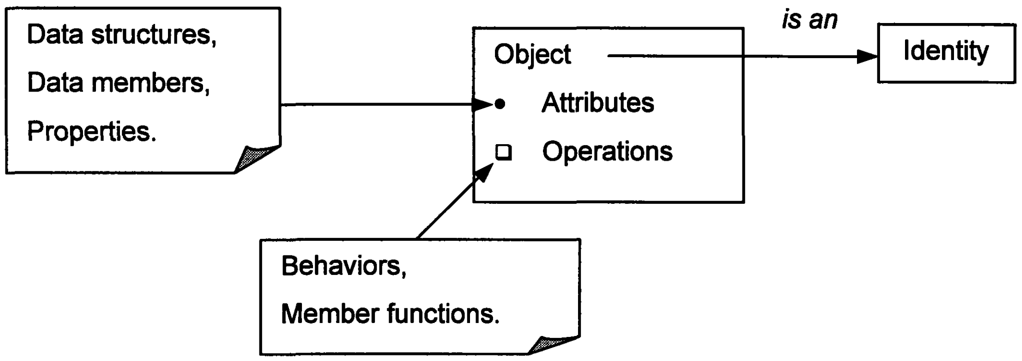


Fig. 2: Object and identity.

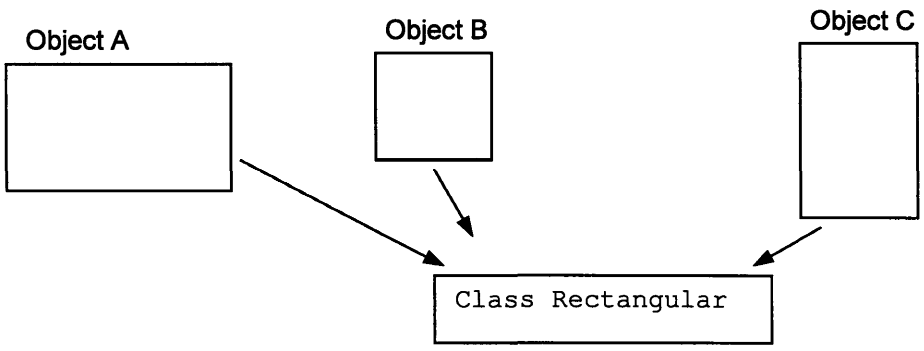


Fig. 3: Objects and class.

Classes with same attributes and behaviors that are shared can be grouped into another class that is called a superclass. The hierarchical relationship between superclasses and subclasses is the sharing of attributes and behaviors, which is called inheritance [3]. Figure 4 shows the relationships between classes `Square`, `Rectangular`, `Triangle`, and `Polygon`. We use the object model notation defined in [3].

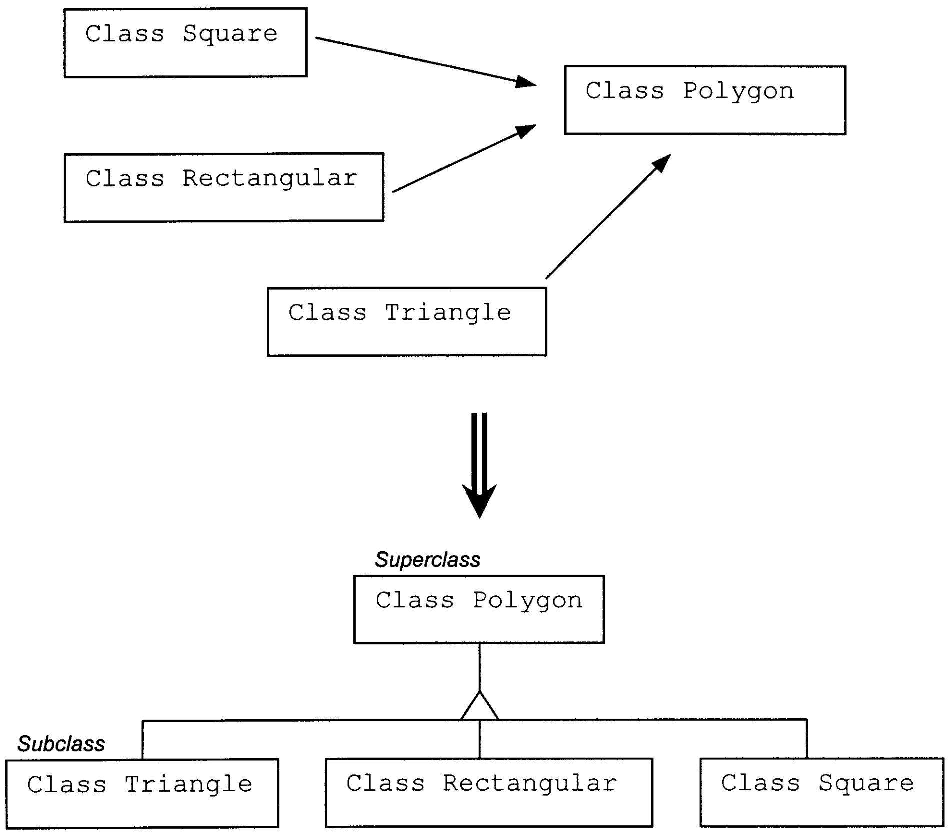


Fig. 4: Superclass and subclass.

1.3 Requirements

The structural analysis tool is designed for general users and developers. The basic needs of general users are as follows:

1. Easy to learn.
2. Easy and intuitive to use.
3. Provide an interactive environment.

The basic needs of developers are as follows:

1. Allow different levels of developers to modify the program.
2. Easy to maintain.
3. Easy to extend and enhance the functions, such as the type of elements.

Taking the procedure to build the model to be the same as the procedure people actually use to build the structure allows beginners to learn to build the model in a shortest time. Beginners can also understand the behavior of the structure through different model design. For the developers, they can add new types of elements in the finite element model. They can also add new functions or build their own graphical user interface. The idea is to use a layered design for the system to meet different requirements.

1.4 Layered Architecture

In a layered architecture, each layer provides only the functions needed for its own purpose. The three-layer design shown in Fig. 5 is adopted use. The first is the finite element kernel, the second is the basic interface, and the third is the graphical user interface.

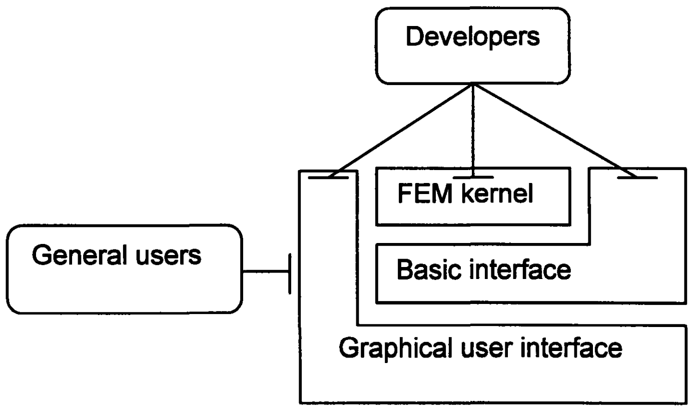


Fig. 5: Layered architecture for users and developers.

With the graphical user interface, general users can build their desired structural frame without knowing anything about the details such as codes, data structures, functions, architecture of classes, and even the finite element theory. For the developers, they can extend and enhance the functionality of each layer. For example, they can build their own graphical user interface, add different types of elements in the finite element kernel, enhance the function of matrix manipulation, and add more dynamic functions to this tool. The functionality of each layer is discussed in the following sections.

1.4.1 Functionality of FEM Kernel

The functionality of the FEM kernel includes the following:

1. Initialize a finite element model.
2. Keep the information about the objects in the finite element model, such as number of nodes, number of beam elements, number of truss elements, and number of boundary conditions. The objects are nodal coordinates, beams, trusses, nodal nodes, boundary conditions, section properties, and material properties.
3. Assemble the stiffness matrix.
4. Solve the matrix equations.
5. Provide the information about nodal displacements.

1.4.2 Functionality of Basic Interface

The basic interface allows for the following input and output actions:

1. Set up nodal coordinates.
2. Set up elements.
3. Set up boundary conditions and elastic supports.
4. Set up external loads.
5. Set up cross section properties.
6. Set up material properties.
7. Output the nodal displacements.

8. Output the member forces.
9. Invoke functions of the FEM kernel.

1.4.3 Functionality of Graphical User Interface

The graphical user interface provides for the following operations:

1. Add elements with their section and materials properties.
2. Set boundary conditions.
3. Set nodal loads.
4. Show the responses of the structure.
5. Show warnings and errors.

As mentioned in section 1.4.2, the functionality of these two different interfaces is very similar. The graphical user interface allows users to build the structural frame by adding truss and beam members, setting up boundary conditions, and adding nodal loads in an intuitive way. Users can observe the deformation of the frame in the display, and retrieve the displacement data from the `TextField` box.

1.5 Implementation

For the implementation, there are a number of computer languages that one can employ. The requirements of the computer language that are relevant for this system are:

1 An object oriented language.
2 A good Integrated Development Environment.
3 Reasonable execution speed for an interactive application.
4 Support a well-developed (robust, easy-to-understand) class library which has the capability of dealing with graphical user interface and network.

Based on the above, Java was chosen as the developing language. In addition, Java has other beneficial aspects such as multithreading, security, and cross platform. In order to provide Internet accessibility and allow the user to run it in the web browser, the program is built as a Java Applet instead of a Java Application. The disadvantage of this choice is that it does not provide file input and output at the client's end, so that one cannot save the data of the models. This problem can be solved by using the network ability in the Java class, but was not attempted in this stage.

2. Finite Element Modeling of Structure

2.1 Analysis

In the finite element method, a structural frame is discretized as a collection of nodes and elements. External loads are imposed on the nodes. Figure 6 shows the relationships between the external loads, the structural frame, and its components- elements and boundary conditions. Each element is related to its end nodes, material, and section property.

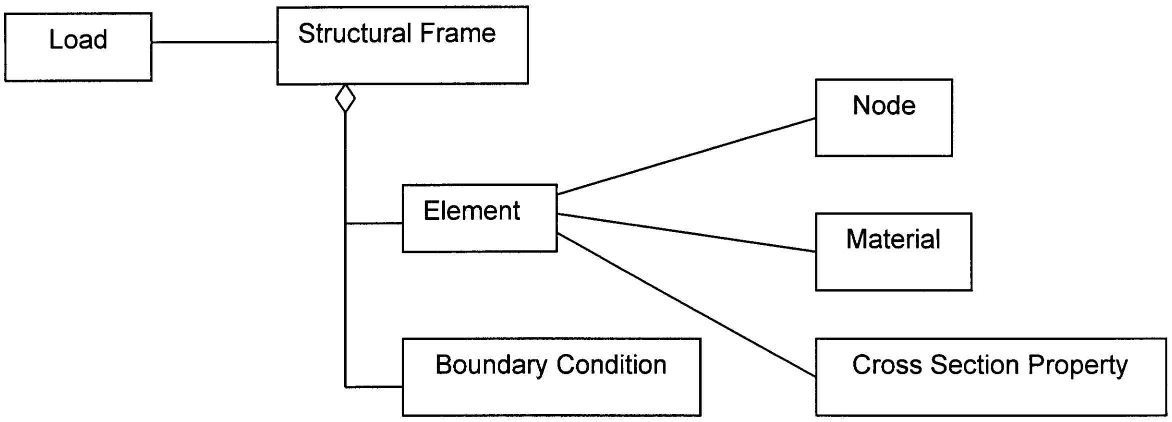


Fig. 6: Structural frame.

Data attributes of nodes are coordinates. Data attributes of materials are Young's modulus, poisson's ratio, etc. Data attributes of section property are cross-sectional area, moment of inertia along y-axis and z-axis. Figure 7 shows the local coordinate system of the 1-D element.

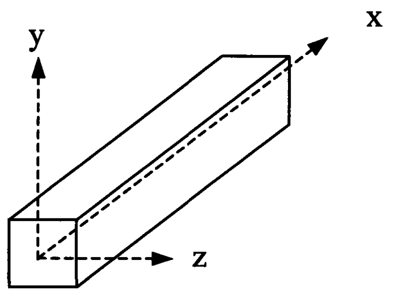


Fig. 7: Local coordinate system for 1-D element.

Figure 8 shows class `Beam`, `Truss`, and `ElasticSupport` can be classfied as class *Element*.

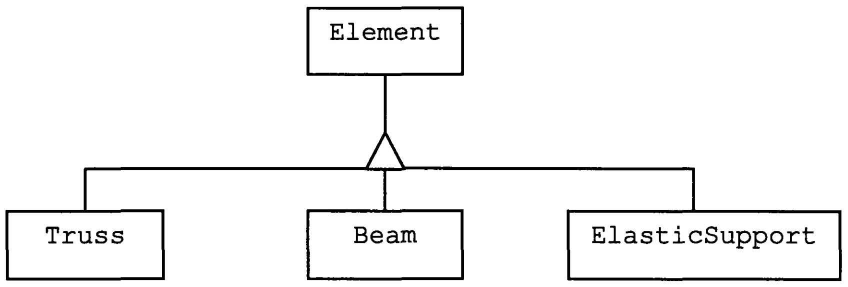


Fig. 8: Classification.

Figure 9 shows the procedures of solving the problem with the finite element method. The first step is to initialize the finite element model. We assign the required parameters, allocate the required memories, and make a connection to the basic interface. The second step is to set up the frame. All the data about the nodal coordinates, elements, materials, properties of cross sections will be connected to the finite element kernel. The third step is to assemble the stiffness matrix. The stiffness matrix of each element is added to the main stiffness matrix. The fourth step is to set up the nodal loads. Then we solve the equations by Gauss elimination and get the nodal displacements. We can also get the member forces after we know the nodal displacements.

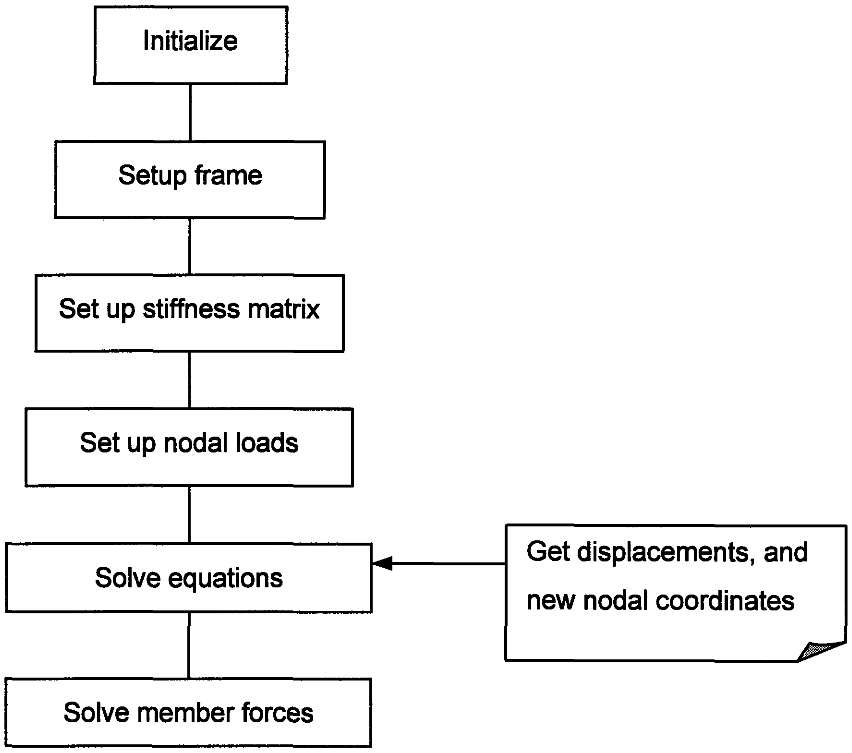


Fig. 9: Procedure of solving the problem by finite element method.

2.2 Design

The finite element model class `FEMAPP` implements all the functions we have mentioned above except the setup of the frame, the calculation of the stiffness matrix of each element and calculation of member forces. The procedure for setting up the frame is implemented in the main class `FemApplet`. Each type of element has its own stiffness matrix. Instead of implementing the calculation of the stiffness matrix of each element in the finite element kernel, we implemented the function of calculation in the class of each element. When we want to assemble the stiffness matrix, we pass the reference of stiffness array to the element's member function. The function will add the transformed stiffness matrix to the main stiffness matrix. The advantage of this method is that it is easy to add new types of elements to the finite element model. For the same reason, the calculation of member forces is implemented in its own class.

2.2.1 Element

We can find that they are some similarities of the functions of different types of elements. We set up an abstract class `Element` that declares the fundamental functions such as initialize the element, and calculate the stiffness matrix. We also define the functions as abstract functions. As beam and truss belong to a kind of element, we define class `Beam` and class `Truss` as the subclasses of class `Element`. Code example 1 shows the implementation of inheritance using "`extends`" in Java language [4].

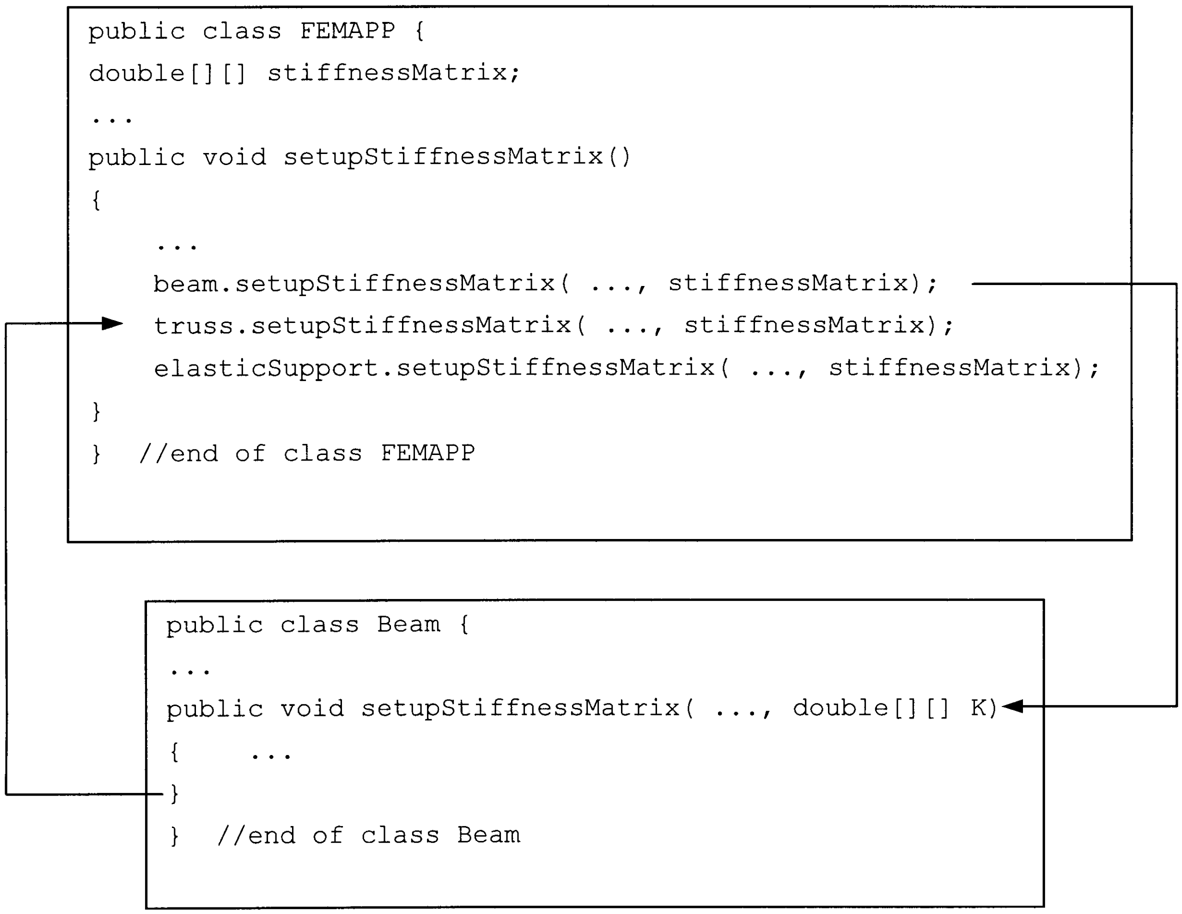
```
class Beam extends Element { ...}
class Truss extends Element { ...}
```

C.E. 1: Class `Beam`, `Truss`, and `Element`.

The subclass `Beam` and `Truss` implement their definitions for the abstract functions that are declared in class `Element` and also keep the data of the array of elements. Code example 2 in the next page shows the implementation of the procedures for calculating the stiffness matrix in Java language.

When we want to set up the main stiffness matrix of the structure, we pass a reference of the two dimension array `stiffnessMatrix` to the member function `setupStiffnessMatrix()` of the class `beam` and class `truss`. Then the function will calculate the transformed stiffness matrix of each member and add it to the main stiffness matrix.

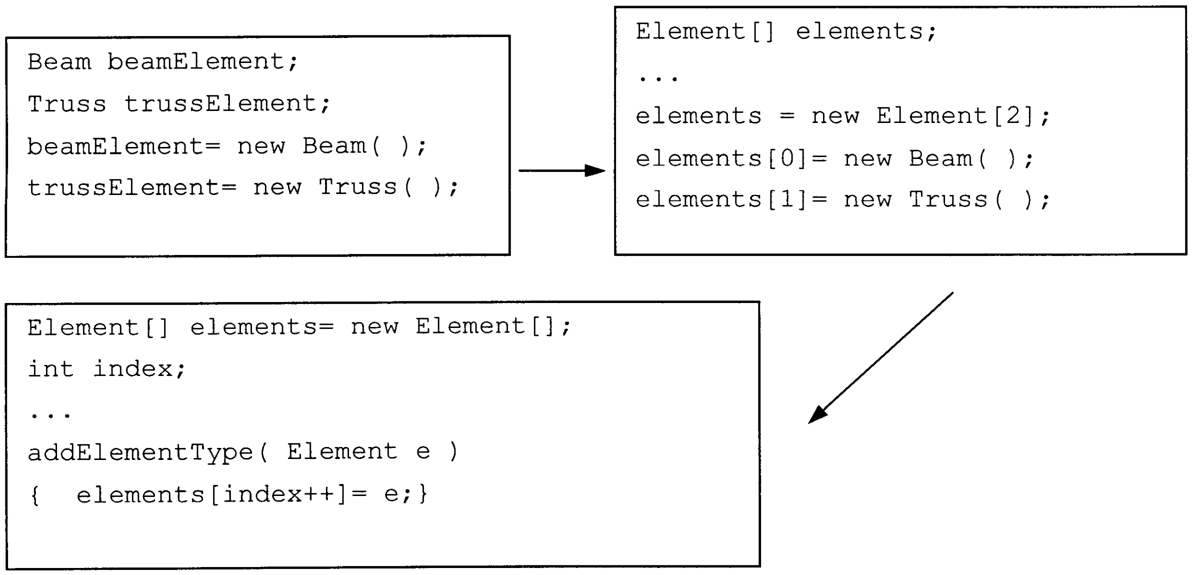
We can merge different types of elements such as beam and truss into an array to enhance the dynamic ability of the class `FEMAPP`. In this way, we can add a new type of element without modifying the original codes in the class `FEMAPP`. But sometimes we have to add a management class to handle it. Code example 3 shows the implementation in Java.



```
public class FEMAPP {
double[][] stiffnessMatrix;
...
public void setupStiffnessMatrix()
{
    ...
    beam.setupStiffnessMatrix( ..., stiffnessMatrix);
    truss.setupStiffnessMatrix( ..., stiffnessMatrix);
    elasticSupport.setupStiffnessMatrix( ..., stiffnessMatrix);
}
}  //end of class FEMAPP
```

```
public class Beam {
...
public void setupStiffnessMatrix( ..., double[][] K)
{     ...
}
}  //end of class Beam
```

C.E. 2: Procedures of calculating stiffness matrix.



```
Beam beamElement;
Truss trussElement;
beamElement= new Beam( );
trussElement= new Truss( );
```

```
Element[] elements;
...
elements = new Element[2];
elements[0]= new Beam( );
elements[1]= new Truss( );
```

```
Element[] elements= new Element[];
int index;
...
addElementType( Element e )
{  elements[index++]= e;}
```

C.E. 3: Class `Beam`, `Truss`, and `Element`.

The alternatives that allow users to extend the functionality of the original class are as follows:

1. Modify the codes of the original class.
2. Extend or inherit to the original class.
3. Enhance the dynamic ability of the original class.

For a well-structured and well-defined problem, one can use the second and the third schemes. For some computer languages, the third scheme may be a better choice.

3. Basic Interface

The basic interface provides data access and function invocation between the finite element kernel and the other modules. Instead of directly accessing the data in the finite element kernel, we can access the data through the interface. We implement the functions according to the interface. If there is no significant change to the finite element kernel, we only have to update the interface instead of modifying all the objects.

3.1 Analysis

The basic interface is a standard interface for programmers to access the data and invoke the functions in the finite element kernel. As shown in Fig 10, there are two ways of data access, one is direct data access and the other is indirect data access. The advantage of direct data access is time; the disadvantage is that it makes the program more vulnerable to changes. Accessing through the interface allows more dynamic implementation in the future.

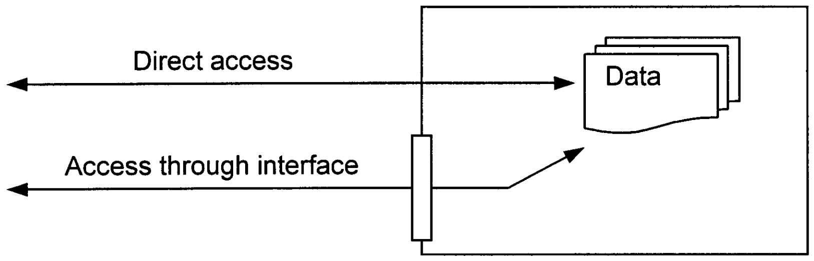


Fig. 10: Data access.

To allow access to the data in an object directly, the object has to expose its data by defining the data as public. For the same reason, the object has to define the interface functions as public. The interface between the kernel and the other classes can be a file input-output interface or a graphical user interface.

3.2 Design

As in the design of the integrated circuit, designers can design their innovative device according to the standards of the interface. Assuming that object A connects to object B through an interface (shown in Fig 11), by designing according to the interface we can change A without changing B or we can change B without changing A. This means that we can uncouple the system and decrease the dependency of the implementation.

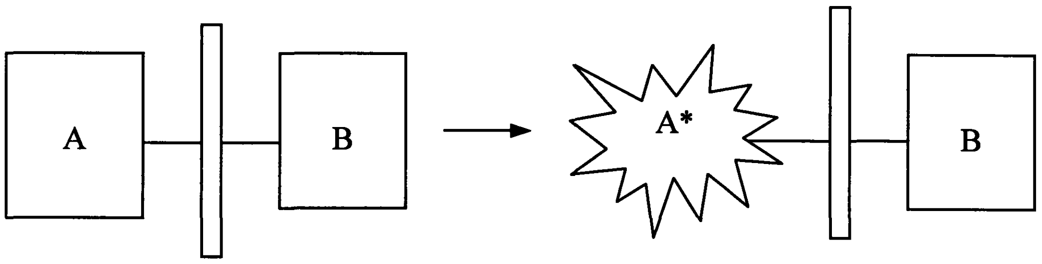


Fig. 11: Interface between objects.

As the interface is between object A and B, it does not belong to either A or B. We set the interface to be an independent class, and make association links to both the objects. Code example 4 shows the implementation in Java language. In the example, class `FEMIO` is the class of the basic interface to the finite element kernel, class `FEMAPP`.

```
public class FEMAPP {
...
FEMIO io;
...
}
```

```
public class FEMIO {
...
}
```

C.E. 4: Association links between FEM kernel and its interface.

For simplicity, the basic interface only provides the fundamental input and output functions. For files or graphics utilities, we can design their input-output interface based on this basic interface. Figure 12 shows the relationships between basic interface and other interfaces.

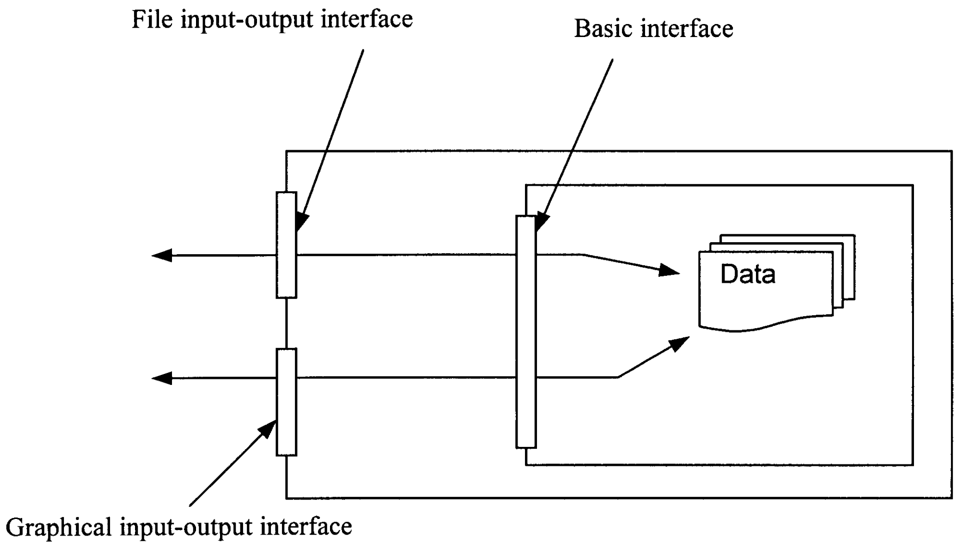


Fig. 12: Relationships between basic interface and other interfaces.

The basic interface implements all the functions mentioned in section 1.4.2 and also make a copy of the required parameters of the finite element model.

4. Interactive Graphical User Interface

An interactive graphical user interface can provide an intuitive way to build the structural frame, and the deformation of the structure will be showed right after the change of the frame and the loads. In this thesis, our goal is to develop a 2-D graphical user interface for the 2-D structural frame.

4.1 Analysis

A window-based system is chosen for our design of the graphical user interface. The mouse and keyboard are the basic and required input devices, and we focus on the mouse-driven interface. The ideas of the user interface are:

1. Easy to learn.
2. Intuitive to operate.
3. Informative.

The graphical user interface is a human-computer interface. The easiest way for a human to build the model is to follow the same approach a human uses to build the real object. A computer-aided tool can decrease the order of the complexity of building the model with a traditional user interface.

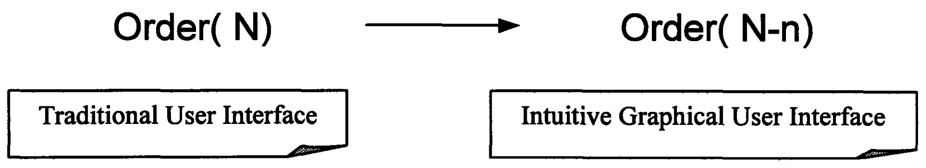

Order(N)
Order(N-n)
Traditional User Interface
Intuitive Graphical User Interface

Fig. 13: Complexity of user interface.

The functions for building the model are as follows:

1. Add elements.
2. Add nodal loads.
3. Set up boundary conditions.

In order to allow users to know the interaction between the imposed variation of the structural model and response of the structure, the following additional functions are required:

1. Allow a user to adjust the magnitude of the nodal loads.
2. Update the output of the screen automatically right after a user makes a change.

The response of the structure includes the followings:

1. Displacements of nodes.
2. Deformations of structure members.
3. Changes of member forces.

4.2 Design

As a window-based system is chosen for the design, at first we have to decide the layout of the graphical user interface. During the procedures of design and implementation of the graphical user interface, we found the graphical user interface includes the following systems:

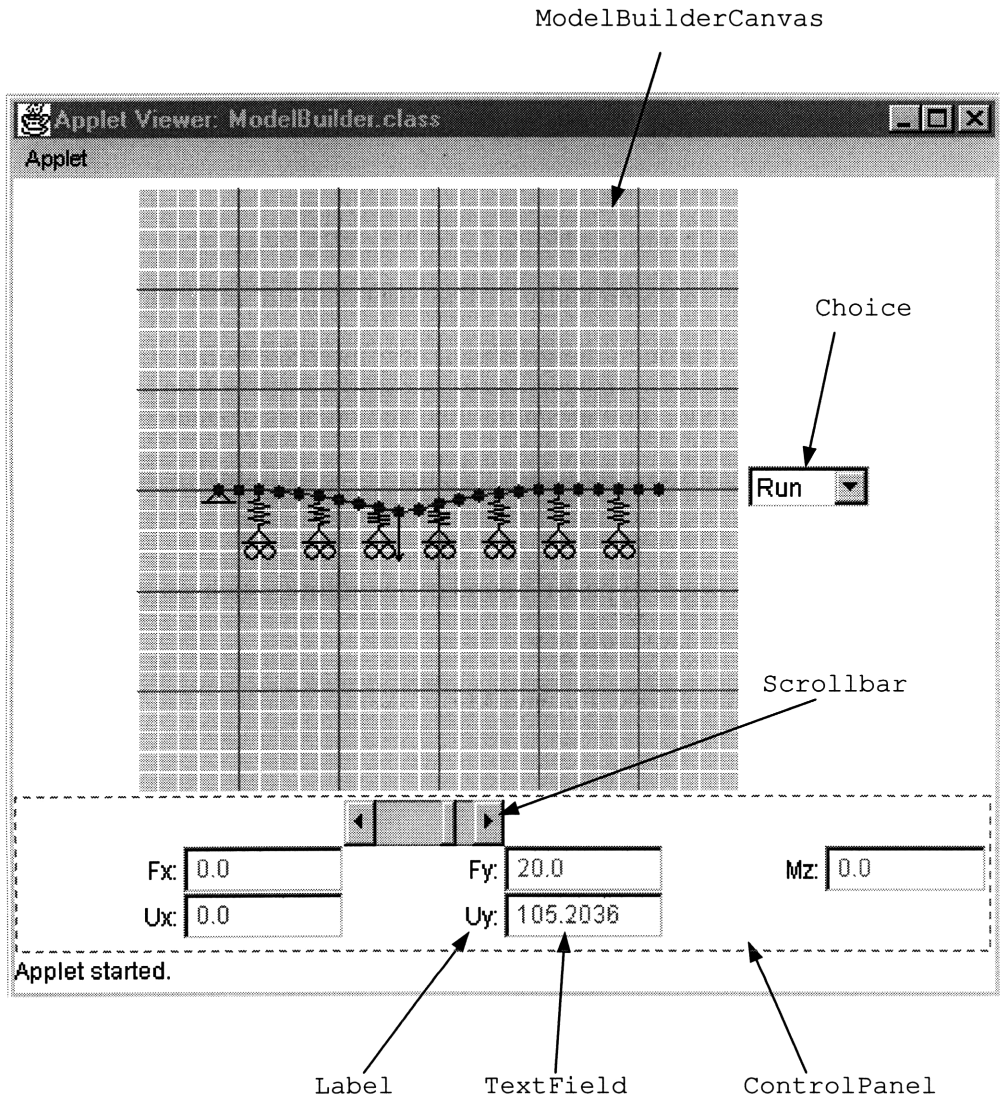
1. Layout and graphical components: `Canvas`, `Choice`, `Scrollbar`, `TextField`, `Label`, `ControlPanel`.
2. Switching system corresponding to the choice we have made.
3. Data transfer system between the components.
4. Event handling system.

4.2.1 Layout and Graphical Components

As shown in Fig 14, the main components of the graphical user interface are as follows:

1. `Canvas`, it allows users to build the structural frame in the canvas and shows the deformations of the structure.
2. `Choice`, it allows users to select the command, such as add beam elements, add truss elements, set boundary conditions, and perform analysis.

3. `Scrollbar`, it allows users to control the properties of the objects.
4. `TextField`, it allows users to access the data of the objects.
5. `Label`, it is used for labeling.
6. `ControlPanel`, it manages the components in the panel.



Fx: 0.0
Fy: 20.0
Mz: 0.0
Ux: 0.0
Uy: 105.2036

Fig. 14: Components in the graphical user interface.

Canvas

The class `ModelBuilderCanvas` reserves all the objects input by the user. Different kind of object is stored in different `Vector` object. Class `ModelBuilderCanvas` also handles all the functions for building the frame. Corresponding to different selections, it processes the event in different ways. When we add elements, boundary conditions, and nodal loads in the `ModelBuilderCanvas`, we have to notice that we do not want to generate duplicate objects of each node. We use "register" method to get the index of the object in the `Vector`. If the object is in the `Vector`, the register method will return the index of the object in the `Vector`. If the object is not in the Vector, the register method will add the object into the `Vector` and then return the index of the object. Each `Vector` has its own implementation of the register method.

When the user adds an object into the `ModelBuilderCanvas`, the related information will be link to the object automatically. The user should also input the following information for each object through the graphical user interface before he add the object:

1. Truss: start node, end node, Young's modulus, and cross-sectional area.
2. Beam: start node, end node, Young's modulus, cross-sectional area, and moment of inertia.
3. Boundary conditions: node, type of boundary condition such as roller-X, roller-Y, hinge-Z, fixed, spring-X, and spring-Y.
4. Load: combination of loads- Fx, Fy, and Mz.

4.2.2 Switching System

As the required functions mentioned in section 4.1, the following commands are added into the `Choice` object:

1. Add truss elements.
2. Add beam elements.
3. Set boundary conditions.
4. Set nodal loads.
5. Execute program.
6. Clear all objects.

As the users make their choices, the layout and the event handler will also change simultaneously. A switching system is adopted to make the entire system perform correspondingly to the choice. An "Index" is also needed for this switching system which is shown in Fig 15.

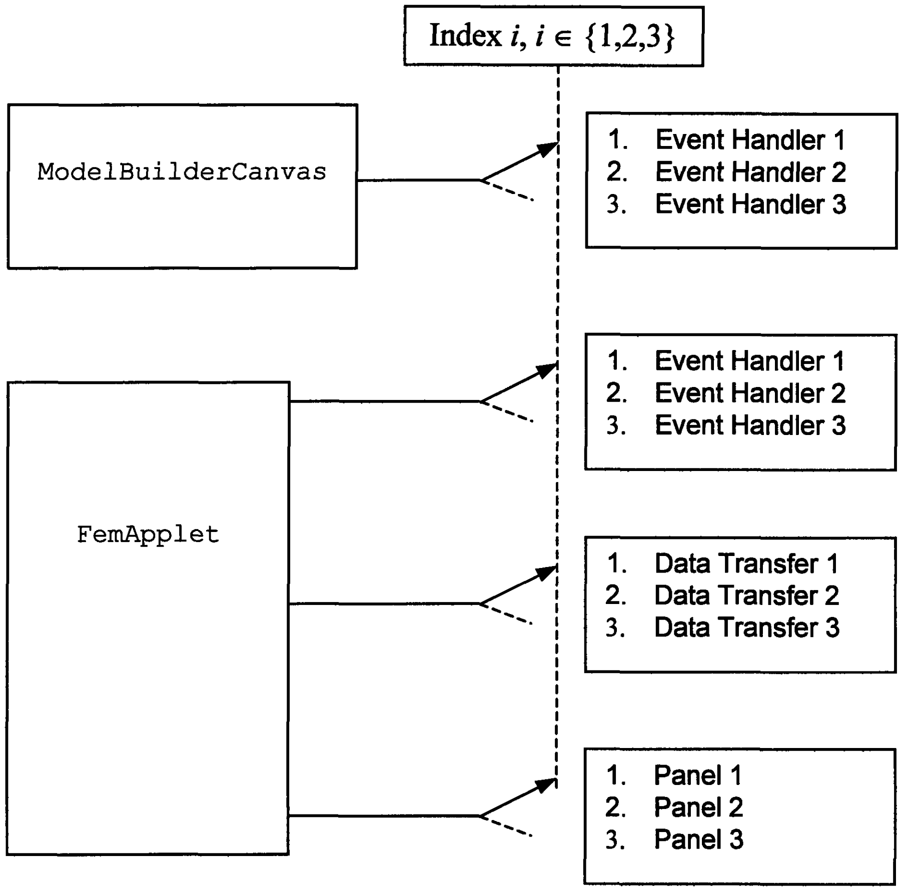


Fig. 15: Switching system.

4.2.3 Event Handling

A window system is a kind of the event-driven system. As the user selects an item in the list of `Choice`, it will generate a hardware event A. The event processing system will get a notification of event A from the operating system. Then the operating system creates an `Event` e object [5] and dispatches the event to its event handler. Figure 16 shows the flow of the event.

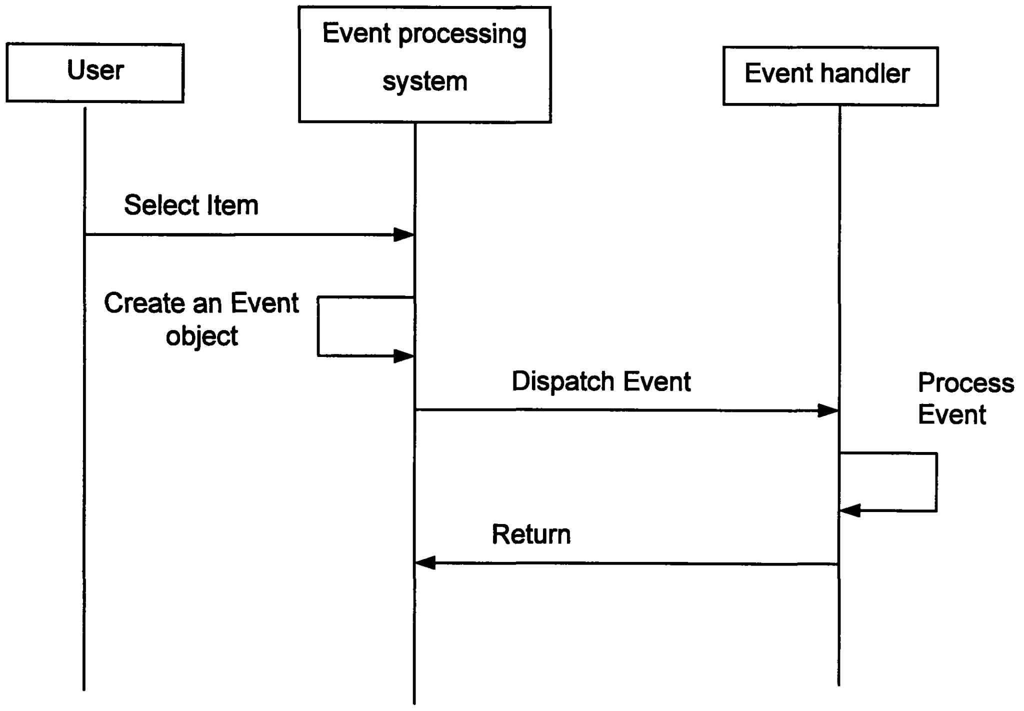


Fig. 16: Event handling.

When the user makes his choice, the corresponding event handler will change the layout in the `ControlPanel`, then the class `FemApplet` calls a `validate()` function to update the layout of the components in the window. Figure 17 shows the above process.

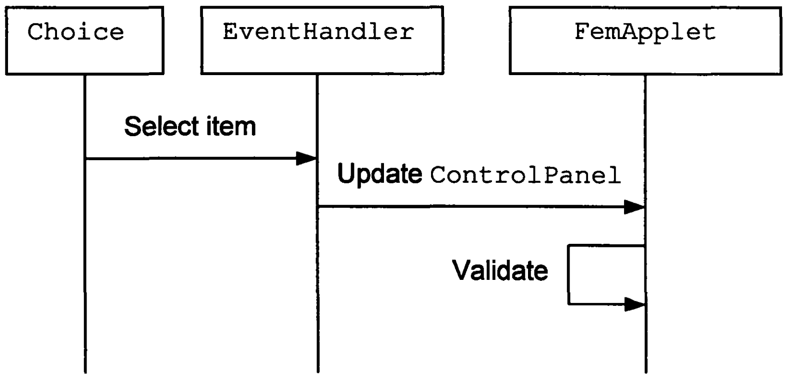


Fig. 17: Choice and the corresponding process.

4.2.4 Data Transfer Mechanism

The required information for each object is input through the `TextField` or `Scrollbar` which reside in the `ControlPanel`. As it is shown in the Fig. 18, we have to consider the data transfer between different components. Data buffers are used for transferring the data between the components. When the class `ModelBuilderCanvas` wants to update data, it invokes a data transfer function of the class `FemApplet`. The data buffers of the class `FemApplet` and the class `ModelBuilderCanvas` are mapped to the same memory. This can simplify the implementation of the data transfer system.

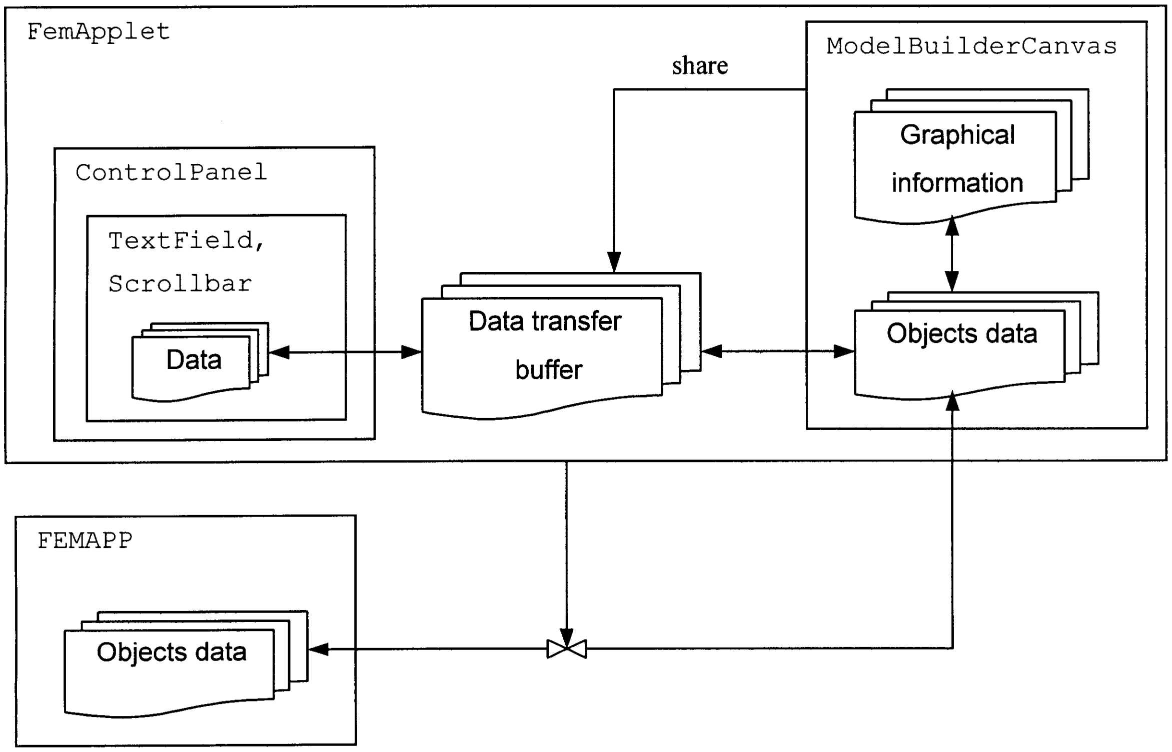


Fig. 18: Data transfer mechanism.

5. Conclusions

In order to provide a realtime response, an interactive analysis tool needs a lot of computation. Although the realtime response is not plausible for very large structures, it is a very important feature for structural analysis and design. It can help structural engineers make a prototype of a structure frame in a very short time and understand the behavior of the structure.

An object oriented approach is good for developers to design the architecture of the software. However usually it is difficult for general developers to understand the relationships between the classes and the functionality of each class. An easier way is to use component oriented approach. Components are self-contained elements of software that can be controlled dynamically and assembled to form applications [6].

The associations of the classes that we have mentioned in the previous chapters are shown in Appendix A. Appendix B shows the implementation using Java language.

6. Future Work

The objective of an interactive structural analysis tool is to help analyzers and designers to realize the structural behaviors. For the future work, we want to improve the analysis tool with the following functions:

1. New types of elements.
2. Nonlinear analysis.
3. Dynamics analysis.
4. 3-D user interface.
5. Information of the validity of assumptions.

7. References

1. R.R. Gajewski. "An Object Oriented Approach to finite element programming," *Artificial Intelligence and Object Oriented Approaches for Structural Engineering*, pp.107-113, 1994.
2. R.S. Arruda, L. Landau, and N.F.F. Ebecken, "Object-Oriented Structural Analysis in a Graphical Environment," *Artificial Intelligence and Object Oriented Approaches for Structural Engineering*, pp.129-138, 1994.
3. J. Runmbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modeling and Design*. Prentice-Hall, 1991.
4. P. Niemeyer and J. Peck, *Exploring Java*. O'Reilly & Associates, 1996.
5. Event Handling. http://java.sun.com/docs/books/tutorial/ui/overview/events.html
6. R. Englander. *Developing JAVA Beans*. O'Reilly & Associates, 1997.

8. Appendices

Notes

Professor Hiroshi Sakuta developed class `Engineer` and the drawing functions in class `FEMVIEW`. The files in the attached disk, which are related to class `Engineer`, are not printed in Appendix B.

The original program used in class `FEMAPP`, `Beam`, and `Truss` has been completed as a result of a series of assignments throughout the course of Finite Element Method instructed by Professor Yeong-Bin Yang, Spring 1993.

The attached disk includes the complete Java source codes and compiled byte-codes. Users can use web browser to open the HTML file, FemApplet.html. The browser will download the Applet automatically. In order to get a better performance, please make a copy to the hard disk and use web browser to open the copy in the hard disk.

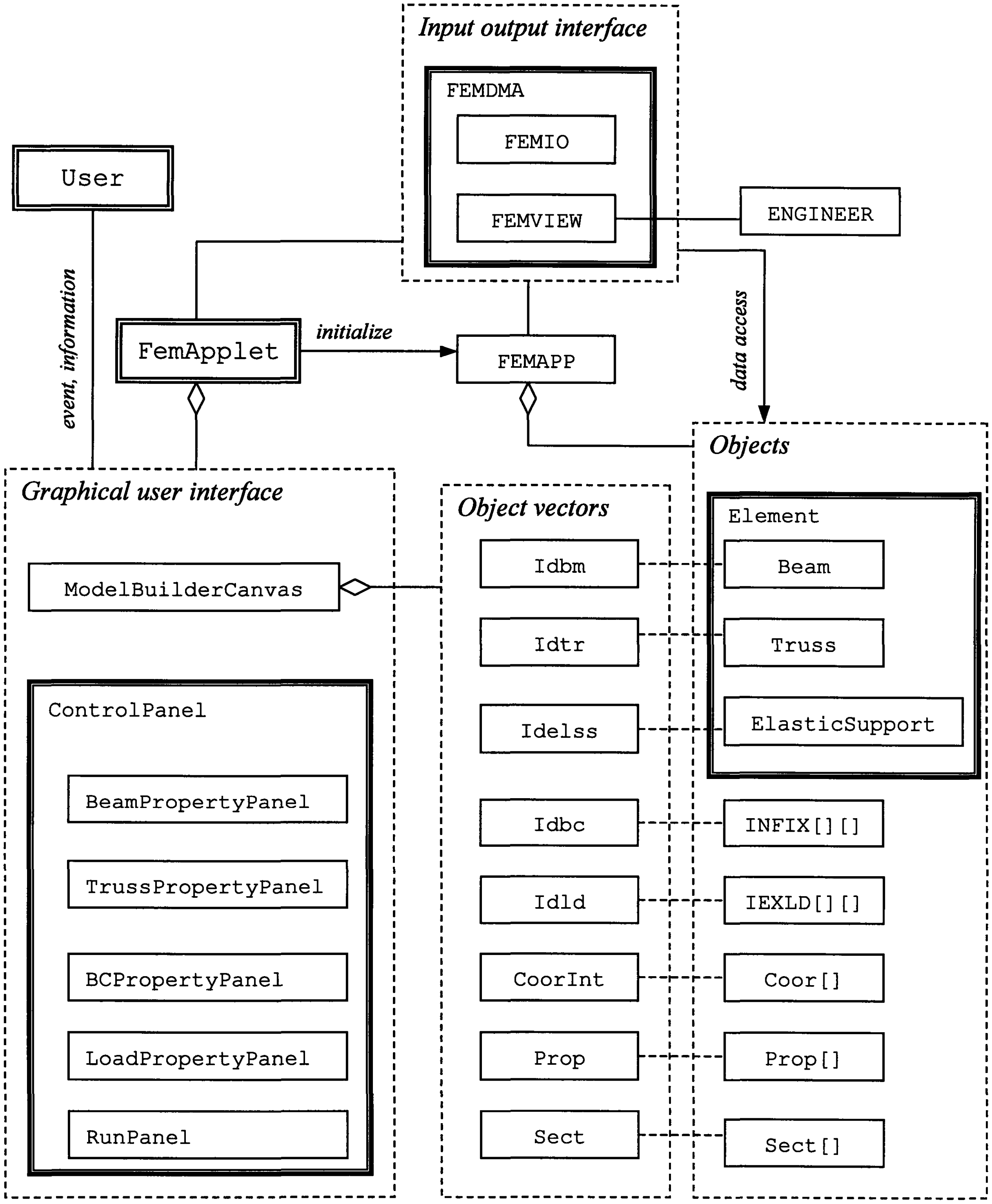


Fig. 19: Class relationships.

Appendix B

BCPropertyPanel.java

```
import java.awt.*;

public class BCPropertyPanel extends ControlPanel {

TextField tf1;
Choice ch1;

public BCPropertyPanel( Notifiable nf)
{  super( nf);}

public void setupComponents()
{
    setLayout( new GridLayout(1,3));

    ch1= new Choice();
    ch1.addItem("RollerX");
    ch1.addItem("RollerY");
    ch1.addItem("HingeZ");
    ch1.addItem("Fixed");
    ch1.addItem("Spring-X");
    ch1.addItem("Spring-Y");

    add( ch1);
    add( new Label("Stiffness: ",Label.RIGHT));
    add( tf1=new TextField("10.",8));

}

public void transferData( int[] bfi, double[] bfD, boolean f)
{
    if( f)
    { bfi[1]=ch1.getSelectedIndex();
      bfD[0]= Double.valueOf(tf1.getText()).doubleValue();
    }

}

}  //end of class BCPropertyPanel;
```

BeamPropertyPanel.java

```
import java.awt.*;

public class BeamPropertyPanel extends ControlPanel {

TextField tf1, tf2, tf3;

public BeamPropertyPanel( Notifiable nf)
{  super( nf);}

public void setupComponents()
{
    setLayout( new GridLayout(1,6));

    add( new Label("Young's: ",Label.RIGHT));
    add( tf1=new TextField("29000.",8));

    add( new Label("Area: ",Label.RIGHT));
    add( tf2=new TextField("10.",8));

    add( new Label("Iz: ",Label.RIGHT));
    add( tf3=new TextField("200",8));
}

public void transferData( int[] bfi, double[] bf, boolean f)
{
    if( f)
    { switch( bfi[2])
      { case 0:
          bf[0]= Double.valueOf(tf1.getText()).doubleValue()*1000.;
          //MPa->KPa
          bf[1]= Double.valueOf(tf2.getText()).doubleValue()/10000.;
          //cm^2->m^2
          bf[2]= Double.valueOf(tf3.getText()).doubleValue()/1.E8;
          //cm^4->m^4
          break;

        case 1:
          bf[0]= Double.valueOf(tf1.getText()).doubleValue()*144.;
          //Ksi->Kips/ft^2
          bf[1]= Double.valueOf(tf2.getText()).doubleValue()/144.;
          //in^2->ft^2
          bf[2]= Double.valueOf(tf3.getText()).doubleValue()/20736.;
          //in^4->ft^4
          break;
      }
    }

}

} //end of class BeamPropertyPanel
```

ControlPanel.java

```
import java.awt.*;

abstract public class ControlPanel extends Panel {
static boolean getValue=true;
static boolean setValue=false;
Notifiable nf;

public ControlPanel( Notifiable nf)
{ this.nf=nf;}

abstract public void setupComponents();

abstract public void transferData( int[] bfi, double[] bfD, boolean f);

} //end of class ControlPanel;
```

Coor.java

```
public class Coor{
public double x, y, z;
public Coor( double x, double y, double z) {
    this.x=x;
    this.y=y;
    this.z=z;
}

public void newPosition( double x, double y) {
    this.x= x;
    this.y= y;
}

} //end of class Coor;
```

CoorInt.java

```
public class CoorInt{
int x, y, z;

public CoorInt( int x, int y, int z) {
    this.x=x;
    this.y=y;
    this.z=z;
}

} //end of class CoorInt;
```

DistributedLoad.java

```
public class DistributedLoad {

public int ib;

public double vx, vy, vz;

public DistributedLoad( int i, double x, double y, double z)
{
    ib=i;
    vx=x;
    vy=y;
    vz=z;
}

} // end of class DistributedLoad;
```

FemApplet.java

```
import java.applet.*;
import java.awt.*;
import fem.core.*;

public class FemApplet extends Applet implements Notifiable
{

    ModelBuilderCanvas mCanvas;
    Choice sel;
    Choice sel2;
    Scrollbar mScrollbar;
    java.util.Vector comps;

    int[] bufferInt;
    double[] bufferDouble;


    FEMAPP frame;

    int NI;

    ControlPanel cPanel;

    Container parent;

    FEMIO femio;

public FemApplet()
{   //setup screen coordinate system

    comps= new java.util.Vector(12);

    bufferInt=    new int[6];
    bufferDouble= new double[6];

}

public void init()
{
    parent = getParent();
    while (parent != null &&  !(parent instanceof Frame)) {
           parent = parent.getParent();
       }

    sel2 = new Choice();
    sel2.addItem("S.I.");
    sel2.addItem("Imperial");
    add(sel2);

}


void removeComponentGroup()
{
    for( int i=comps.size()-1; i>=0; i--)
       {
         remove( (Component)comps.elementAt(i));
       }
    comps.removeAllElements();
}


Component addComponentGroup( Component comp)
{
    add( comp);
    comps.addElement( comp);
    return comp;
}

public void transferData()
{
    cPanel.transferData( bufferInt, bufferDouble,
ControlPanel.getValue);
}

public boolean action(Event e, Object arg) {
    int i, j;

    if( e.target== sel)
      { mCanvas.repaint();

        i=sel.getSelectedIndex();

        bufferInt[0]= i;

        switch(i) {
            case 0:
                removeComponentGroup();
                addComponentGroup( cPanel=new TrussPropertyPanel(this));
                cPanel.setupComponents();
                break;

            case 1:
                removeComponentGroup();
                addComponentGroup( cPanel=new BeamPropertyPanel(this));
                cPanel.setupComponents();
                break;

            case 2:
                removeComponentGroup();
                addComponentGroup( cPanel=new BCPropertyPanel(this));
                cPanel.setupComponents();
                break;

            case 3:
                removeComponentGroup();
                addComponentGroup( cPanel=new LoadPropertyPanel(this));
                cPanel.setupComponents();
                break;

            case 4: // RUN
                removeComponentGroup();

                if( initializeFEM())
                  { input();
                    femio.setupMatrix();
                    setupLoads();
                    femio.solveMatrix();

                    mCanvas.updateCoordinates( femio.ICOORN);

                    Idld ld=(Idld)mCanvas.loads.elementAt(0);

                    bufferDouble[1]=ld.Px;
                    bufferDouble[2]=ld.Py;
                    bufferDouble[3]=ld.Mz;

                    mCanvas.repaint();

                    NI=0;

                    addComponentGroup( cPanel=new RunPanel( this));
                cPanel.setupComponents();

                updateValue();
              }

            break;

        case 5: //None
            removeComponentGroup();
            break;

        case 6: //Clear
            removeComponentGroup();
            mCanvas.clearAll();
            //mCanvas.repaint();

            remove( mCanvas);
            remove( sel);
            sel= null;

            sel2 = new Choice();
            sel2.addItem("S.I.");
            sel2.addItem("Imperial");
            add(sel2);

            System.gc();
            break;

    }

    validate();

    return true;

  }

if( e.target== sel2)
  {
    i=sel2.getSelectedIndex();

    bufferInt[2]= i;

    remove( sel2);

    mCanvas= new ModelBuilderCanvas( this, (Frame)parent,
                                     bufferInt, bufferDouble);
    mCanvas.reshape( 10,10,300,300);
    mCanvas.setBackground( Color.lightGray);
    add( mCanvas);

    sel= new Choice();
    sel.addItem("Truss");
    sel.addItem("Beam");
    sel.addItem("B.C.");
    sel.addItem("Load");
    sel.addItem("Run");
    sel.addItem("-----");
    sel.addItem("Clear");
    add(sel);

    bufferInt[0]= 0;   // Choice=Member
```

```
            removeComponentGroup();

            addComponentGroup( cPanel=new TrussPropertyPanel( this));

            cPanel.setupComponents();

            validate();

            return true;
          }

        return false;

    }

    void updateValue()
    {
        Coor p;

        p=femio.getNodalDisplacement(NI);

        bufferDouble[4]= p.x;

        bufferDouble[5]= p.y;

        cPanel.transferData( bufferInt,bufferDouble,ControlPanel.setValue);
    }

    public void mNotify()
    {
        int ni, n, i;
        double s;

        transferData();

        if( (n=mCanvas.loads.size())<=0) return;

        s=bufferDouble[0];

        Idld ld=(Idld)mCanvas.loads.elementAt(0);

        femio.setExld( ni=ld.Ni, bufferDouble[1]=ld.Px*s,
                       bufferDouble[2]=ld.Py*s, bufferDouble[3]=ld.Mz*s);

        for( i=1; i<n; i++)
           { ld=(Idld)mCanvas.loads.elementAt(i);
             if( ld.Ni== ni)
               { femio.addExld( ni, ld.Px, ld.Py, ld.Mz);
               }
           }

        femio.solveMatrix();            //solve Matrix

        mCanvas.updateCoordinates( femio.ICOORN);

        mCanvas.repaint();

        updateValue();

    }
```

```
boolean initializeFEM() {

    int NNODE= mCanvas.coordinates.size();
    int NPROP= mCanvas.props.size();
    int NSECT= mCanvas.sects.size();
    int NBEAM= mCanvas.beams.size();
    int NTRUS= mCanvas.truss.size();
    int NCONS= 0;                               //constraints
    int NELAS= mCanvas.elasticSupport.size();   //elastic support
    int NDISL= 0;                               //distributed load

    if( NBEAM==0 && NTRUS==0) return false;

    // node, mat, sec, beam, truss, constr, elastics support, distl
    frame= new FEMAPP( NNODE, NPROP, NSECT, NBEAM, NTRUS, NCONS, NELAS,
                       NDISL);

    femio=frame.io;

    return true;

}

void input()
{
    femio.setNode( mCanvas.coordinates);

    femio.setIdbm(   mCanvas.beams);  //2D
    femio.setIdtr(   mCanvas.truss);
    femio.setIdelss( mCanvas.elasticSupport);

    femio.setBoundaryConditions( mCanvas.boundaryConditions);  //2D

    femio.setProp( mCanvas.props);
    femio.setSect( mCanvas.sects);

} // end of input

void setupLoads()
{
    java.util.Vector v=mCanvas.loads;

    Idld ld;

    femio.clearLoad();
    for( int i= v.size()-1; i>=0; i--)
       { ld=(Idld)v.elementAt(i);
         femio.addExld( ld.Ni, ld.Px, ld.Py, ld.Mz);
       }

} // end of setupLoads

} // end of class FemApplet;
```

Idbc.java

```
public class Idbc {

public int Ni;
public int type;

public static final int rollerX= 0;
public static final int rollerY= 1;
public static final int hingeZ=  2;
public static final int fixed=   3;

public Idbc( int n, int t)
{
    Ni=n; type=t;
}

} //end of class Idbc;
```

Idbm.java

```
public class Idbm {
public int Ni, Nj, Mat, Sec, b1, b2;

public Idbm( int NI, int NJ, int MAT, int SEC, int B1, int B2)
{ Ni=NI+1; Nj=NJ+1; Mat=MAT; Sec=SEC; b1=B1; b2=B2; }

public Idbm( int NI, int NJ, int MAT, int SEC)
{ Ni=NI+1; Nj=NJ+1; Mat=MAT; Sec=SEC; }

} //end of Idbm;
```

Idelss.java

```
public class Idelss {

public double k;
public int Ni;
public int type;

public static final int springX= 0;
public static final int springY= 1;

public Idelss( int n, int t, double K)
{   Ni= n;
    type= t;
    k= K;
}

} //end of class Idelss;
```

Idld.java

```
public class Idld {

public int Ni;
public double Px;
public double Py;
public double Mz;

public Idld( int Ni, double Px, double Py, double Mz)
{
    this.Ni= Ni;
    this.Px= Px;
    this.Py= Py;
    this.Mz= Mz;
}

} //end of class Idld;
```

Idtr.java

```
public class Idtr {
public int Ni, Nj, Mat, Sec, b1, b2;

public Idtr( int NI, int NJ, int MAT, int SEC, int B1, int B2)
{ Ni=NI+1; Nj=NJ+1; Mat=MAT; Sec=SEC; b1=B1; b2=B2; }

public Idtr( int NI, int NJ, int MAT, int SEC)
{ Ni=NI+1; Nj=NJ+1; Mat=MAT; Sec=SEC; }

} //end of class Idtr;
```

LoadPropertyPanel.java

```
import java.awt.*;

public class LoadPropertyPanel extends ControlPanel {

TextField tf1, tf2, tf3;

public LoadPropertyPanel( Notifiable nf)
{  super( nf);}

public void setupComponents()
{
    setLayout( new GridLayout(1,6));

    add( new Label("Px: ",Label.RIGHT));
    add( tf1=new TextField("0.",8));

    add( new Label("Py: ",Label.RIGHT));
    add( tf2=new TextField("0.",8));

    add( new Label("Mz: ",Label.RIGHT));
    add( tf3=new TextField("0.",8));
}

public void transferData( int[] bfi, double[] bf, boolean f)
{
    if( f)
    { bf[0]= Double.valueOf(tf1.getText()).doubleValue();
      bf[1]= Double.valueOf(tf2.getText()).doubleValue();
      bf[2]= Double.valueOf(tf3.getText()).doubleValue();
    }
}

} //end of class LoadPropertyPanel;
```

ModelBuilderCanvas.java

```
import java.awt.*;

public class ModelBuilderCanvas extends Canvas {

Image offScrImg;

boolean mDownFlag;

public java.util.Vector beams;
public java.util.Vector truss;
public java.util.Vector elasticSupport;
public java.util.Vector coordinatesInt;
public java.util.Vector coordinates;
public java.util.Vector boundaryConditions;
public java.util.Vector loads;
public java.util.Vector props;
public java.util.Vector sects;

public java.util.Vector coordinatesIntTmp;
public java.util.Vector coordinatesIntNew;

int iX, iY, jX, jY;
static double scale= 0.1, invScale=10.;

int select;
int[] bInt;
double[] bDouble;

FemApplet fa;

Frame frame;

static double colorFactor;

ModelBuilderCanvas( FemApplet fa, Frame frame, int[] bufferInt, double[]
bufferDouble)
{
    select= 0;
    bInt= bufferInt;
    bDouble= bufferDouble;
    this.fa= fa;
    this.frame= frame;

    colorFactor= Math.PI/200;

    coordinates=        new java.util.Vector(20);
    beams=              new java.util.Vector(20);
    truss=              new java.util.Vector(20);
    elasticSupport=     new java.util.Vector(20);
    props=              new java.util.Vector(20);
    sects=              new java.util.Vector(20);
    loads=              new java.util.Vector(20);
    boundaryConditions= new java.util.Vector(20);
    coordinatesInt=     new java.util.Vector(20);
    coordinatesIntNew=  new java.util.Vector(20);
    coordinatesIntTmp= coordinatesInt;

}
```

```
public void clearAll()
{

    beams.removeAllElements();
    truss.removeAllElements();
    elasticSupport.removeAllElements();
    coordinatesInt.removeAllElements();
    coordinates.removeAllElements();
    boundaryConditions.removeAllElements();
    loads.removeAllElements();
    props.removeAllElements();
    sects.removeAllElements();
    coordinatesIntNew.removeAllElements();
    coordinatesIntTmp.removeAllElements();

}

public void updateCoordinates( Coor[] ICOORN)
{
    int i=coordinatesIntNew.size()-1;
    CoorInt ci;

    for( ; i>=0; i--)
       { ci=(CoorInt)coordinatesIntNew.elementAt(i);
         ci.x= (int)Math.round(ICOORN[i].x*invScale);
         ci.y= (int)Math.round(ICOORN[i].y*invScale);
         ci.z= (int)Math.round(ICOORN[i].z*invScale);
       }

}

public void update( Graphics g)
{

    if( offScrImg== null)
        offScrImg= createImage( size().width, size().height);

    Graphics og= offScrImg.getGraphics();

    og.clearRect( 0,0,size().width,size().height);
    og.setColor( Color.lightGray);
    og.fillRect( 0,0,size().width,size().height);

    paint(og);

    g.drawImage( offScrImg, 0, 0, this);

    og.dispose();

}

public void paint( Graphics g)
{   int i, ni, nj, x, y, t, x1, y1;
    double px, py, mz, s;

    g.setColor( Color.white);

    for( x=0; x<300; x+=10)
       { g.drawLine( x, 0, x, 299);
         g.drawLine( 0, x, 299, x);}
```

```
g.setColor( Color.gray);

for( x=50; x<300; x+=50)
   { g.drawLine( x, 0, x, 299);
     g.drawLine( 0, x, 299, x);}

if( bInt[0]==4) { coordinatesInt= coordinatesIntNew;}
else { coordinatesInt= coordinatesIntTmp;}

// Draw Boundary Conditions
g.setColor( Color.blue);

i= boundaryConditions.size()-1;

for( ;i>=0; i--)
   {
     ni= ((Idbc)boundaryConditions.elementAt(i)).Ni;
     t= ((Idbc)boundaryConditions.elementAt(i)).type;

     //System.out.println( ni);
     //System.out.println( t);

     x= ((CoorInt)coordinatesInt.elementAt(ni)).x;
     y= ((CoorInt)coordinatesInt.elementAt(ni)).y;

     switch(t) {
     case Idbc.rollerX:
          g.drawLine( x,y,x+6,y+6);
          g.drawLine( x,y,x-6,y+6);
          g.drawLine( x-8,y+6,x+8,y+6);

          g.drawOval( x-7,y+7,7,7);
          g.drawOval( x+1,y+7,7,7);
          break;

      case Idbc.rollerY:
          g.drawLine( x,y,x+6,y+6);
          g.drawLine( x,y,x+6,y-6);
          g.drawLine( x+6,y-8,x+6,y+8);

          g.drawOval( x+7,y-7,7,7);
          g.drawOval( x+7,y+1,7,7);
          break;

       case Idbc.hingeZ:
          g.drawLine( x,y,x+6,y+6);
          g.drawLine( x,y,x-6,y+6);
          g.drawLine( x-8,y+6,x+8,y+6);
          break;

      case Idbc.fixed:
          g.drawLine( x-8,y,x+8,y);
          g.drawLine( x-4,y,x-7,y+9);
          g.drawLine( x,y,x-3,y+9);
          g.drawLine( x+4,y,x+1,y+9);
          break;

     }

     }
```

```
// Draw Elastic Supports
    g.setColor( Color.blue);

    i= elasticSupport.size()-1;

    for( ;i>=0; i--)
       {
         ni= ((Idelss)elasticSupport.elementAt(i)).Ni;
         t= ((Idelss)elasticSupport.elementAt(i)).type;

         //System.out.println( ni);
         //System.out.println( t);

         x= ((CoorInt)coordinatesInt.elementAt(ni)).x;
         y= ((CoorInt)coordinatesInt.elementAt(ni)).y;

         x1=((CoorInt)coordinatesIntTmp.elementAt(ni)).x;
         y1=((CoorInt)coordinatesIntTmp.elementAt(ni)).y;

         switch(t) {
            case Idelss.springX:
                s=1.-(double)(x-x1)/18.;

                g.drawLine( x,y,x+(int)(5*s+0.5),y);

                g.drawLine( x+(int)(5*s+0.5),y,   x+(int)(6*s+0.5),
                            y-5);
                g.drawLine( x+(int)(6*s+0.5),y-5, x+(int)(8*s+0.5),
                            y+5);
                g.drawLine( x+(int)(8*s+0.5),y+5, x+(int)(10*s+0.5),
                            y-5);
                g.drawLine( x+(int)(10*s+0.5),y-5, x+(int)(12*s+0.5),
                            y+5);
                g.drawLine( x+(int)(12*s+0.5),y+5,x+(int)(14*s+0.5),
                            y-5);
                g.drawLine( x+(int)(14*s+0.5),y-5, x+(int)(16*s+0.5),
                            y+5);
                g.drawLine( x+(int)(16*s+0.5),y+5,x+(int)(17*s+0.5),y);

                g.drawLine( x+(int)(17*s+0.5),y,x1+20,y);

                x=x1+20;

                g.drawLine( x,y,x+6,y+6);
                g.drawLine( x,y,x+6,y-6);
                g.drawLine( x+6,y-8,x+6,y+8);

                g.drawOval( x+7,y-7,7,7);
                g.drawOval( x+7,y+1,7,7);
                break;

            case Idelss.springY:
                s=1.-(double)(y-y1)/18.;

                g.drawLine( x,y,x,y+(int)(5*s+0.5));

                g.drawLine( x  ,y+(int)(5*s+0.5), x-5,y+(int)(6*s+0.5));
                g.drawLine( x-5,y+(int)(6*s+0.5), x+5,y+(int)(8*s+0.5));
                g.drawLine( x+5,y+(int)(8*s+0.5), x-5,
                            y+(int)(10*s+0.5));
                g.drawLine( x-5, y+(int)(10*s+0.5), x+5,
                        y+(int)(12*s+0.5));
                g.drawLine( x+5,y+(int)(12*s+0.5),x-5,
                            y+(int)(14*s+0.5));
                g.drawLine( x-5, +(int)(14*s+0.5),
                            x+5,y+(int)(16*s+0.5));
                g.drawLine( x+5,y+(int)(16*s+0.5),x,
                            y+(int)(17*s+0.5));
                g.drawLine( x,y+(int)(17*s+0.5),x,y1+20);

                y=y1+20;

                g.drawLine( x,y,x+6,y+6);
                g.drawLine( x,y,x-6,y+6);
                g.drawLine( x-8,y+6,x+8,y+6);

                g.drawOval( x-7,y+7,7,7);
                g.drawOval( x+1,y+7,7,7);
                break;


          }
        }

// Draw Nodes
    g.setColor( Color.red);

      i= coordinates.size()-1;

      for( ;i>=0; i--)
         {
             x= ((CoorInt)coordinatesInt.elementAt(i)).x;
             y= ((CoorInt)coordinatesInt.elementAt(i)).y;

             g.fillOval( x-3,y-3,7,7);

         }

// Draw Truss
      i= truss.size()-1;

      for( ;i>=0; i--)
         {
             ni= ((Idtr)truss.elementAt(i)).Ni-1;
             nj= ((Idtr)truss.elementAt(i)).Nj-1;

         if( bInt[0]==4) g.setColor( colorOfForce(
                         fa.femio.truss.ElementForce(i,fa.femio.IPP)));

             g.drawLine( ((CoorInt)coordinatesInt.elementAt(ni)).x,
                         ((CoorInt)coordinatesInt.elementAt(ni)).y,
                         ((CoorInt)coordinatesInt.elementAt(nj)).x,
                         ((CoorInt)coordinatesInt.elementAt(nj)).y );

         }

// Draw Beams
    g.setColor( Color.magenta);
      i= beams.size()-1;

      for( ;i>=0; i--)
         {
            ni= ((Idbm)beams.elementAt(i)).Ni-1;
            nj= ((Idbm)beams.elementAt(i)).Nj-1;

            g.drawLine( ((CoorInt)coordinatesInt.elementAt(ni)).x,
                        ((CoorInt)coordinatesInt.elementAt(ni)).y,
                        ((CoorInt)coordinatesInt.elementAt(nj)).x,
                        ((CoorInt)coordinatesInt.elementAt(nj)).y );

        }

// Draw Loads
    g.setColor( Color.black);

    i= loads.size()-1;

      for( ;i>=0; i--)
        {
            ni= ((Idld)loads.elementAt(i)).Ni;
        if( bInt[0]==4 && i==0)
          { px= bDouble[1];
            py= bDouble[2];
            mz= bDouble[3];
          }
        else
              { px= ((Idld)loads.elementAt(i)).Px;
                py= ((Idld)loads.elementAt(i)).Py;
                mz= ((Idld)loads.elementAt(i)).Mz;
          }

            x= ((CoorInt)coordinatesInt.elementAt(ni)).x;
            y= ((CoorInt)coordinatesInt.elementAt(ni)).y;

        g.setColor( colorOfForce(px));
            if( px< -1.0E-6)
          {
            g.drawLine( x-24, y, x, y);
            g.drawLine( x-24, y, x-21, y-2);
            g.drawLine( x-24, y, x-21, y+2);
          }
        else if( px> 1.0E-6)
               {
                 g.drawLine( x+24, y, x, y);
                 g.drawLine( x+24, y, x+21, y-2);
                 g.drawLine( x+24, y, x+21, y+2);
               }

        g.setColor( colorOfForce(py));
            if( py< -1.0E-6)
          {
            g.drawLine( x, y-24, x, y);
            g.drawLine( x, y-24, x-2, y-21);
            g.drawLine( x, y-24, x+2, y-21);
          }
        else if( py> 1.0E-6)
               {
                 g.drawLine( x, y+24, x, y);
                 g.drawLine( x, y+24, x-2, y+21);
                 g.drawLine( x, y+24, x+2, y+21);
               }
```

```
// drawArc(int  x, int  y, int  width, int  height, int  startAngle, int
arcAngle)
         g.setColor( colorOfForce(mz));
             if( mz< -1.0E-6)
           {
             g.drawArc( x-11, y-11, 23, 23, 15, 210);
             g.drawLine( x-8, y+8, x-8, y+5);
             g.drawLine( x-8, y+8, x-11, y+7);
           }
         else if( mz> 1.0E-6)
                {
                  g.drawArc( x-11, y-11, 23, 23, 165, -210);
                  g.drawLine( x+8, y+8, x+8, y+5);
                  g.drawLine( x+8, y+8, x+11, y+7);
                }


         }

}

public boolean mouseEnter(Event evt, int x, int y) {
    frame.setCursor( Frame.CROSSHAIR_CURSOR);
    return true;
}

public boolean mouseExit(Event evt, int x, int y) {
    frame.setCursor( Frame.DEFAULT_CURSOR);
    return true;
}

public boolean mouseDown(Event evt, int x, int y)
{
    mDownFlag= true;

    switch( select= bInt[0]) {
        case 0:
        case 1:
            iX=(x+2)/5*5;
            iY=(y+2)/5*5;
            getGraphics().fillOval( iX-3,iY-3,7,7);
            return true;

        case 2:
            return true;

        case 3:
            return true;

        case 4:
            return true;
     }

    return true;
}

public boolean mouseUp(Event evt, int x, int y)
{
    if( mDownFlag== true && select== bInt[0])
      { jX=(x+2)/5*5;
        jY=(y+2)/5*5;
        fa.transferData();

        switch( select) {
            case 0:
                if( Math.abs(jX-iX)<4 && Math.abs(jY-iY)<4 ) break;
                addTrussElement();
                break;

            case 1:
                if( Math.abs(jX-iX)<4 && Math.abs(jY-iY)<4 ) break;
                addBeamElement();
                break;

            case 2:
                addBoundaryCondition();
                break;

            case 3:
                addLoad();
                break;

            case 4:
                showNodalDisplacement();
                break;
        }

        repaint();

      }

      mDownFlag= false;

    return true;
}

void addTrussElement()
{

      truss.addElement( new Idtr( registerNode( iX, iY),
                                  registerNode( jX, jY),
                                  registerProp( bDouble[0])+1,
                                                        //Material
                                  registerSect( bDouble[1])+1
                                                //Section Property
                              ));

}

void addBeamElement()
{   int p, s, N;
    double sX, sY, dX, dY;

    p= registerProp( bDouble[0])+1;
    s= registerSect( bDouble[1], bDouble[2])+1;

    N=Math.max( Math.max( Math.abs(iY-jY)/10, Math.abs(iX-jX)/10), 1);

    dX= (double)(jX-iX)/(double)N;
    dY= (double)(jY-iY)/(double)N;
    sX= iX;
    sY= iY;

    for( int i=0; i<N; i++)
       { beams.addElement( new Idbm( registerNode( (int)Math.round(sX),
                                                   (int)Math.round(sY)),
                                  registerNode(
                         (int)Math.round(sX+dX), (int)Math.round(sY+dY)),
                                  p, s
                                 ));
         sX+= dX;
         sY+= dY;
       }

}

void addBoundaryCondition()
{
    if( bInt[1]<4)
      { boundaryConditions.addElement( new Idbc( registerNode( jX, jY),
                                                  bInt[1]
//Type of Boundary Condition
                                                ) ); }
    else
      { elasticSupport.addElement( new Idelss( registerNode( jX, jY),
                                                bInt[1]-4,
//Type of Boundary Condition
                                                bDouble[0]
                                                )); }
}

void addLoad()
{
    if( Math.abs(bDouble[0])+Math.abs(bDouble[1])+Math.abs(bDouble[2])<
1.0e-6) return;

    loads.addElement( new Idld( registerNode( jX, jY),
                                bDouble[0], bDouble[1], bDouble[2]
//Loads
                               ));

}

void showNodalDisplacement()
{   int i, n;
    CoorInt iCoorInt;

    n= coordinatesInt.size();

    for( i=0; i<n; i++)
       { iCoorInt=(CoorInt)coordinatesInt.elementAt(i);
         if( (iCoorInt.x-jX)*(iCoorInt.x-jX)+ (iCoorInt.y-jY)*
             (iCoorInt.y-jY)<=9)
           { fa.NI=i;
             fa.updateValue();
             return;
           }
       }
}

int registerNode( int x, int y)
{
    int n, i;
    CoorInt iCoorInt;

    n= coordinatesInt.size();

    for( i=0; i<n; i++)
       { iCoorInt=(CoorInt)coordinatesInt.elementAt(i);
         if( (iCoorInt.x-x)*(iCoorInt.x-x)+ (iCoorInt.y-y)*
              (iCoorInt.y-y)<=9) return i;
       }

    coordinatesInt.addElement( new CoorInt( x, y, 0));
    coordinatesIntNew.addElement( new CoorInt( x, y, 0));
    coordinates.addElement( new Coor( scale*(double)x, scale*(double)y,
                                      0.));

    return n;

}


int registerSect( double A)
{
    return registerSect( A, 0.);
}


int registerSect( double A, double Iy)
{
    int n, i;
    Sect iSect;

    n= sects.size();

    for( i=0; i<n; i++)
       { iSect=(Sect)sects.elementAt(i);
         if( Math.abs(iSect.A-A) + Math.abs(iSect.Iy-Iy)<=1.0E-6)
             return i;
       }

    sects.addElement( new Sect( A, Iy));

    return n;
}

int registerProp( double E)
{
    int n, i;
    Prop iProp;

    n= props.size();

    for( i=0; i<n; i++)
       { iProp=(Prop)props.elementAt(i);
         if( Math.abs(iProp.mE-E)<=1.0E-6) return i;
       }

    props.addElement( new Prop( E));

    return n;
```

```
}

Color colorOfForce( double f)
{
    if( f<0)
        return( new Color( (float)Math.sin(Math.min(-f,100.)
                               *colorFactor), 0.0f, 0.0f));   // (r,g,b)
    else
        return( new Color( 0.0f, 0.0f, (float)Math.sin(
                                        Math.min(f,100.)*colorFactor)));

}

} // end of class ModelBuilderCanvas;
```

Notifiable.java

```
interface Notifiable {

void mNotify();
} //end of interface Notifiable;
```

Prop.java

```
public class Prop {
public double mE, mP, b1, b2;

public Prop( double ME, double MP, double B1, double B2)
{ mE=ME; mP=MP; b1=B1; b2=B2;}

public Prop( double ME, double MP)
{ mE=ME; mP=MP;}

public Prop( double ME)
{ mE=ME; mP=0.;}

} //end of class Prop;
```

RunPanel.java

```
import java.awt.*;

public class RunPanel extends ControlPanel {

TextField tf1, tf2, tf3, tf4, tf5;
Scrollbar sb1;

GridLayout gl;

static boolean Forward=true;

public RunPanel( Notifiable nf)
{   super( nf);}

public void setupComponents()
{

    gl=new GridLayout(3,6);
    setLayout( gl);

    sb1= new Scrollbar( Scrollbar.HORIZONTAL, 5, 1, -10, 10);
    sb1.setPageIncrement(1);

    add( new Label(" "));
    add( new Label(" "));
    add( sb1);
    add( new Label(" "));
    add( new Label(" "));
    add( new Label(" "));

    add( new Label("Fx: ",Label.RIGHT));
    add( tf1=new TextField("0.",8));

    add( new Label("Fy: ",Label.RIGHT));
    add( tf2=new TextField("0.",8));

    add( new Label("Mz: ",Label.RIGHT));
    add( tf3=new TextField("0.",8));

    add( new Label("Ux: ",Label.RIGHT));
    add( tf4=new TextField("0.",8));

    add( new Label("Uy: ",Label.RIGHT));
    add( tf5=new TextField("0.",8));

    tf1.disable();
    tf2.disable();
    tf3.disable();
    tf4.disable();
    tf5.disable();
}

public void transferData( int[] bfi, double[] bfD, boolean f)
{
    if( f)
      { bfD[0]= (double)sb1.getValue()/5.; }
    else
      {
        tf1.setText( String.valueOf((float)bfD[1]));
        tf2.setText( String.valueOf((float)bfD[2]));
        tf3.setText( String.valueOf((float)bfD[3]));

        switch( bfi[2])
        {
            case 0: tf4.setText( String.valueOf((float)(bfD[4]*100.)));
                    //m->cm
                    tf5.setText( String.valueOf((float)(bfD[5]*100.)));
                    //m->cm
                    break;

            case 1: tf4.setText( String.valueOf((float)(bfD[4]*12.)));
                    //ft->in
                    tf5.setText( String.valueOf((float)(bfD[5]*12.)));
                    //ft->in
                    break;
        }
      }
}
```

```
public boolean handleEvent( Event event )
{
    if( event.target == sbl)
      { if( event.id == Event.SCROLL_ABSOLUTE  ||
            event.id == Event.SCROLL_LINE_DOWN ||
            event.id == Event.SCROLL_LINE_UP   ||
            event.id == Event.SCROLL_PAGE_DOWN ||
            event.id == Event.SCROLL_PAGE_UP )
          {
            nf.mNotify();
            return true;
          }
       }

    return super.handleEvent( event );
  }

} //end of RunPanel;
```

Sect.java

```
public class Sect {
public double A, Iy, Iz, J, b1, b2;

public Sect( double a, double IY, double IZ, double j, double B1,
             double B2)
{ A=a; Iy=IY; Iz=IZ; J=j; b1=B1; b2=B2;}

public Sect( double a, double IY, double IZ)
{ A=a; Iy=IY; Iz=IZ; J=0.;}

public Sect( double a, double IZ)
{ A=a; Iy=0.; Iz=IZ; J=0.;}

} //end of class Sect;
```

TrussPropertyPanel.java

```
import java.awt.*;

public class TrussPropertyPanel extends ControlPanel {

TextField tf1, tf2;

public TrussPropertyPanel( Notifiable nf)
{  super( nf);}

public void setupComponents()
{
    setLayout( new GridLayout(1,4));

    add( new Label("Young's: ",Label.RIGHT));
    add( tf1=new TextField("29000.",8));

    add( new Label("Area: ",Label.RIGHT));
    add( tf2=new TextField("10.",8));
}

public void transferData( int[] bfi, double[] bf, boolean f)
{
    if( f)
    { switch( bfi[2])
      { case 0:
          bf[0]= Double.valueOf(tf1.getText()).doubleValue()*1000.;
          //MPa->KPa
          bf[1]= Double.valueOf(tf2.getText()).doubleValue()/10000.;
          //cm^2->m^2
          break;

        case 1:
          bf[0]= Double.valueOf(tf1.getText()).doubleValue()*144.;
          //Ksi->Kips/ft^2
          bf[1]= Double.valueOf(tf2.getText()).doubleValue()/144.;
          //in^2->ft^2
          break;
      }
    }
}

} //end of class TrussPropertyPanel;
```

fem.core.Beam.java

```
package fem.core;

public class Beam extends Element{

int NBEAM;

Idbm   IIDBM[];
double IVECT[][];
int    LM1[][];

// external link to the node coordinate
Coor ICOOR[];
Prop IPROP[];
Sect ISECT[];

int  NADD[];
double _ISS[];

fem.core.FEMAPP femapp;

public Beam( fem.core.FEMAPP fa, int n)
{
    femapp= fa;
    NBEAM=n;
}

public void INIT()
{
    IVECT= new double[NBEAM][3];
    IIDBM= new Idbm[NBEAM];
    LM1= new int[NBEAM][12];
}

public void SKYLINE( int iidnd[][], int nonz[], int NEQ)
{   int i, j, ii, jj, n, Imin;
```

```
    for( i=0; i<NBEAM; i++)
       { ii=IIDBM[i].Ni-1;
         jj=IIDBM[i].Nj-1;
         Imin= NEQ;
         for( j=0; j<6; j++)
            { LM1[i][j]=   iidnd[ii][j];
              LM1[i][j+6]= iidnd[jj][j];
            }
         for( j=0; j<12; j++)
            if( (n= LM1[i][j])!=0 && n< Imin) Imin=n;
         for( j=0; j<12; j++)
            { if( (n=LM1[i][j])!=0 && nonz[n-1]> Imin) nonz[n-1]=Imin;}
       }
}

//******************** STIFF ******************
public void STIFF( Coor icoor[], Prop iprop[], Sect isect[], int nadd[],
double _iss[])
{   int IB,i,j;

    double ESL[][], RSR[][];
    double ROT[][], TRN[][];
    double RL;

    ICOOR= icoor;
    IPROP= iprop;
    ISECT= isect;
    NADD=  nadd;
    _ISS=  _iss;

    ESL= new double[12][12];
    RSR= new double[12][12];
    ROT= new double[3][3];
    TRN= new double[3][3];

    for( i=0; i<12; i++) for( j=0; j<12; j++) ESL[i][j]=0.;
      for( IB=0; IB<NBEAM;  IB++)
         { RL= ROTATE( IB, ROT, TRN);
           ELEMENT( IB, RL, ESL);
           TRANS( ROT, TRN, ESL, RSR );

           ASSEM( LM1[IB], RSR);}
      return;}


double ROTATE( int ib, double rot[][], double trn[][])
{     int Ni, Nj;
      double RL;
      int i,j;

      Ni= IIDBM[ib].Ni-1;
      Nj= IIDBM[ib].Nj-1;

      rot[0][0]= ICOOR[Nj].x- ICOOR[Ni].x;
      rot[0][1]= ICOOR[Nj].y- ICOOR[Ni].y;
      rot[0][2]= ICOOR[Nj].z- ICOOR[Ni].z;

      RL= FEMAPP.Length( rot[0]);        //length of beam
      FEMAPP.Normalize(  rot[0]);        //vector of local-x

      FEMAPP.Crossproduct( rot[2], rot[0], IVECT[ib]);
      FEMAPP.Normalize( rot[2]);         //vector of local-z

      FEMAPP.Crossproduct( rot[1], rot[2], rot[0]);
      FEMAPP.Normalize( rot[1]);         //vector of local-y


      for( i=0; i<3; i++) for( j=0; j<3; j++) trn[i][j]= rot[j][i];

      return RL;
}

void TRANS( double rot[][], double trn[][], double esl[][],
            double rsr[][])
{   int i, j;
      for( i=0; i<12; i+=3)
         for( j=0; j<=i; j+=3)
            Trans( rot, trn, esl, rsr, i, j);
      return;
}

void Trans( double rot[][], double trn[][], double esl[][],
            double rsr[][],
            int m, int n)
{   int i,j,k;
    double sum;
    double tmp[][];
    tmp= new double[3][3];

    for( i=0; i<3; i++)
       for( j=0;  j<3; j++)
          { sum=0.;
            for( k=0; k<3; k++) sum += trn[i][k]*esl[k+m][j+n];
            tmp[i][j]= sum;
          }

    for( i=0; i<3; i++)
       for( j=0;  j<3; j++)
          { sum=0.;
            for( k=0; k<3; k++) sum += tmp[i][k]*rot[k][j];
            rsr[i+m][j+n]= sum;
          }

    return;
}


void ASSEM( int lm[], double rsr[][])
{     int i,j,Ni,Nj;
      for( i=0; i<12; i++)
         for( j=0; j<=i; j++)
            { if( lm[i]==0 || lm[j]==0 ) continue;
              Ni=lm[i]-1; Nj=lm[j]-1;
              // add stiffness matrix[Ni][Nj];
              if( Ni>= Nj ) _ISS[ NADD[Ni]+Nj] += rsr[i][j];
              else  _ISS[ NADD[Nj]+Ni] += rsr[i][j];
            }
      return;
}

void ELEMENT( int ib, double rl, double esl[][])
{
```

```
int i,j;
double EE, POIS, AREA, ERIZ, ERIY, RJ;
int mt, st;
double L, L2, L3;

L=  rl;
L2= L*L;
L3= L2*L;

mt= IIDBM[ib].Mat-1;
st= IIDBM[ib].Sec-1;

EE=    IPROP[mt].mE;
POIS=  IPROP[mt].mP;

AREA=  ISECT[st].A;
ERIZ=  EE* ISECT[st].Iz;
ERIY=  EE* ISECT[st].Iy;
RJ=    ISECT[st].J;

esl[ 0][ 0]=  EE * AREA / L;
esl[ 0][ 6]= -esl[0][0];
esl[ 1][ 1]=  12.* ERIZ / L3;
esl[ 1][ 5]=  6. * ERIZ / L2;
esl[ 1][ 7]= -esl[1][1];
esl[ 1][11]=  esl[1][5];
esl[ 2][ 2]=  12.* ERIY / L3;
esl[ 2][ 4]= -6. * ERIY / L2;
esl[ 2][ 8]= -esl[2][2];
esl[ 2][10]=  esl[2][4];

esl[ 3][ 3]=  EE * RJ/(2.*(1.+POIS)*L);
esl[ 3][ 9]= -esl[3][3];
esl[ 4][ 2]=  esl[2][4];
esl[ 4][ 4]=  4. * ERIY / L;
esl[ 4][ 8]= -esl[4][2];
esl[ 4][10]=  2. * ERIY / L;
esl[ 5][ 1]=  esl[1][5];
esl[ 5][ 5]=  4. * ERIZ / L;
esl[ 5][ 7]= -esl[5][1];
esl[ 5][11]=  2. * ERIZ / L;

esl[ 6][ 0]= -esl[0][ 0];
esl[ 6][ 6]= -esl[0][ 6];
esl[ 7][ 1]= -esl[1][ 1];
esl[ 7][ 5]= -esl[1][ 5];
esl[ 7][ 7]= -esl[1][ 7];
esl[ 7][11]= -esl[1][11];
esl[ 8][ 2]= -esl[2][ 2];
esl[ 8][ 4]= -esl[2][ 4];
esl[ 8][ 8]= -esl[2][ 8];
esl[ 8][10]= -esl[2][10];

esl[ 9][ 3]= -esl[3][ 3];
esl[ 9][ 9]=  esl[3][ 3];
esl[10][ 2]=  esl[4][ 2];
esl[10][ 4]=  esl[4][10];
esl[10][ 8]=  esl[4][ 8];
esl[10][10]=  esl[4][ 4];
esl[11][ 1]=  esl[5][ 1];
esl[11][ 5]=  esl[5][11];
```

```
      esl[11][ 7]=  esl[5][ 7];
      esl[11][11]=  esl[5][ 5];

      return;
}


// *********** FORCE ***********
void FORCE( double IPP[])
{   int IB,i,j;
    double ESL[][];
    double ROT[][], TRN[][];
    double EDG[], EDL[], FF[];
    double RL, sum;

    ESL= new double[12][12];
    ROT= new double[3][3];
    TRN= new double[3][3];
    EDG= new double[12];
    EDL= new double[12];
    FF=  new double[12];

    for( i=0; i<12; i++) for( j=0; j<12; j++) ESL[i][j]=0.;

    //if( sysout) System.out.println("\n     BEAM ELEMENT FORCES\n");

    for( IB=0; IB<NBEAM;  IB++)
       { RL= ROTATE( IB, ROT, TRN);
         for( i=0; i<12; i++)
              { if( (j=LM1[IB][i]) == 0) { EDG[i]= 0; continue;}
                  EDG[i]= IPP[j-1];
             }

            for( i=0; i<3; i++)
                 { EDL[i]  = ROT[i][0]*EDG[0]+ ROT[i][1]*EDG[1]+
ROT[i][2]*EDG[2];
                   EDL[i+3]= ROT[i][0]*EDG[3]+ ROT[i][1]*EDG[4]+
ROT[i][2]*EDG[5];
                   EDL[i+6]= ROT[i][0]*EDG[6]+ ROT[i][1]*EDG[7]+
ROT[i][2]*EDG[8];
                   EDL[i+9]= ROT[i][0]*EDG[9]+ ROT[i][1]*EDG[10]+
ROT[i][2]*EDG[11];
             }

            ELEMENT( IB, RL, ESL);
            for( i=0; i<12; i++)
               { sum=0;
                 for( j=0; j<12; j++) sum += ESL[i][j] * EDL[j];
                 FF[i]= sum;
               }
       }
      return;
}

public double ElementForce( int IB, int dof, double IPP[])
{     int i,j;
      double ESL[][];
      double ROT[][], TRN[][];
      double EDG[], EDL[], FF[];
      double RL, sum;
```

```
       ESL= new double[12][12];
       ROT= new double[3][3];
       TRN= new double[3][3];
       EDG= new double[12];
       EDL= new double[12];
       FF=  new double[12];

       for( i=0; i<12; i++) for( j=0; j<12; j++) ESL[i][j]=0.;

       RL= ROTATE( IB, ROT, TRN);
       for( i=0; i<12; i++)
          { if( (j=LM1[IB][i]) == 0) { EDG[i]= 0; continue;}
            EDG[i]= IPP[j-1];
          }

       for( i=0; i<3; i++)
          { EDL[i]   = ROT[i][0]*EDG[0]+ ROT[i][1]*EDG[1]+
ROT[i][2]*EDG[2];
              EDL[i+3]= ROT[i][0]*EDG[3]+ ROT[i][1]*EDG[4]+
ROT[i][2]*EDG[5];
              EDL[i+6]= ROT[i][0]*EDG[6]+ ROT[i][1]*EDG[7]+
ROT[i][2]*EDG[8];
              EDL[i+9]= ROT[i][0]*EDG[9]+ ROT[i][1]*EDG[10]+
ROT[i][2]*EDG[11];
        }

       ELEMENT( IB, RL, ESL);

       sum=0.;
       for( j=0; j<12; j++) sum += ESL[dof][j] * EDL[j];
       return sum;
}

public double valueOf( int n, int d)
{
    return ElementForce( n, d, femapp.IPP);
}

} //end of class Beam;
```

fem.core.ElasticSupport.java

```
package fem.core;

public class ElasticSupport extends Element{

int NELSS;
Idelss IDELSS[];
int LM[];

fem.core.FEMAPP femapp;

public ElasticSupport( fem.core.FEMAPP fa, int n)
{
    femapp= fa;
    NELSS=n;
}

public void INIT()
{
    IDELSS= new Idelss[NELSS];
    LM= new int[NELSS];
}

public void SKYLINE( int iidnd[][], int nonz[], int NEQ)
{
    int i;
    for( i=0; i<NELSS; i++)
       { LM[i]= iidnd[IDELSS[i].Ni][IDELSS[i].type];
       }

}

public void STIFF( Coor icoor[], Prop iprop[], Sect isect[], int nadd[],
double _iss[])
{   int i, n;

    for( i=0; i<NELSS; i++)
       { if( (n=LM[i])==0) continue;
            _iss[ nadd[n-1]+n-1] += IDELSS[i].k;
       }
}

public double ElementForce( int i, double IPP[])
{   int n;

    if( (n=LM[i])==0) return 0.;

    return IPP[n-1]*IDELSS[i].k;
}

public double valueOf( int n, int d)
{
    return ElementForce( n, femapp.IPP);
}

} //end of class ElasticSupport;
```

fem.core.Element.java

```
package fem.core;

public abstract class Element {

abstract public void INIT();

abstract public void SKYLINE( int iidnd[][], int nonz[], int NEQ);

abstract public void STIFF( Coor icoor[], Prop iprop[], Sect isect[],
int nadd[], double _iss[]);

abstract public double valueOf( int n, int d);

} // end of class Element;
```

fem.core.FEMAPP.java

```
package fem.core;
```

```
public class FEMAPP {

static double EXPL=1.e-6;

int NNOD, NMAT, NSEC, NBEAM, NTRUS, NCST, NELSS, NDISTL;

Coor   ICOOR[], ICOORN[];
double IEXLD[][], IFELD[][];
Prop   IPROP[];
Sect   ISECT[];

DistributedLoad distributedLoad[];

Beam beam;
Truss truss;
ElasticSupport elasticSupport;

int INFIX[][];
int IIDND[][];

int NONZ[];
int NADD[];

int MHB, MSS;           // MHB: max bandwidth  MSS: max storage size

int NEQ, NLC;
int INDEX, ISIGN;

double ISS[], _ISS[], IPP[];

boolean sysout=false;

public FEMIO io;

public FEMVIEW view;

public FEMAPP( int nnod, int nmat, int nsec, int nbeam, int ntrus,
               int ncst,
               int nelss, int ndistl)
{
     NNOD=  nnod;         //number of nodes
     NMAT=  nmat;         //number of materials
     NSEC=  nsec;         //number of sections
     NBEAM= nbeam;        //number of beams
     NTRUS= ntrus;        //number of truss
     NCST=  ncst;         //number of constraints
     NELSS= nelss;        //number of elastic support
     NDISTL= ndistl;      //number of distributed load

     io= new FEMIO( this);
     io.INIT1( nnod, nmat, nsec, nbeam, ntrus, ncst, nelss, ndistl);

     view= null;

     INIT();
     if( sysout) System.out.println( "%INIT() is OK!");
}

public FEMAPP( Engineer jcc, int nnod, int nmat, int nsec, int nbeam,
               int ntrus, int ncst,
               int nelss, int ndistl)
```

```
{
     NNOD=  nnod;          //number of nodes
     NMAT=  nmat;          //number of materials
     NSEC=  nsec;          //number of sections
     NBEAM= nbeam;         //number of beams
     NTRUS= ntrus;         //number of truss
     NCST=  ncst;          //number of constraints
     NELSS= nelss;         //number of elastic support
     NDISTL= ndistl;       //number of distributed load

     io= new FEMIO( this);
     io.INIT1( nnod, nmat, nsec, nbeam, ntrus, ncst, nelss, ndistl);

     view= new FEMVIEW( this, jcc);
     view.INIT1( nnod, nmat, nsec, nbeam, ntrus, ncst, nelss, ndistl);

     INIT();
     if( sysout) System.out.println( "%INIT() is OK!");
}

void setupMatrix()
{    int i;

     NEQ=IDMAT();  // Get the total DOF and number the non-fixed DOF
     if( sysout) System.out.println( "%IDMAT() is OK!");
     IIDND= INFIX;

     SKYLINE();
     if( sysout) System.out.println( "%SKYLINE() is OK!");

     ISS=  new double[MSS];
     _ISS= new double[MSS];
     IPP=  new double[NEQ];

     io.INIT3( _ISS, IPP, NEQ);
     if( view!=null) view.INIT3( _ISS, IPP, NEQ);

     // this process is necessary, because STIFFx() use += to add the
elements of
     // the stiffness matrix
     CLEAR( _ISS, MSS);
     if( sysout) System.out.println( "%CLEAR() is OK!");

     if( beam!=null)
       { beam.STIFF( ICOOR, IPROP, ISECT, NADD, _ISS);
         if( sysout)
           { System.out.println( "%STIFF1:ISS - - - - - -");
             for( i=0; i<MSS; i++) System.out.println( _ISS[i]/12.);
           }
       }

     if( truss!=null)
       { truss.STIFF( ICOOR, IPROP, ISECT, NADD, _ISS);
         if( sysout)
           { System.out.println( "%STIFF2: ISS - - - - - -");
             for( i=0; i<MSS; i++) System.out.println( _ISS[i]/12.);
           }
       }

     if( elasticSupport!=null)
       { elasticSupport.STIFF( ICOOR, IPROP, ISECT, NADD, _ISS);
```

```
        }

}

void setupMatrix2()
{    int i;

     CLEAR( _ISS, MSS);

     if( beam!=null) beam.STIFF( ICOOR, IPROP, ISECT, NADD, _ISS);

     if( truss!=null) truss.STIFF( ICOOR, IPROP, ISECT, NADD, _ISS);

     if( elasticSupport!=null) elasticSupport.STIFF( ICOOR, IPROP,
ISECT, NADD, _ISS);

}

void solveMatrix()
{    int i;
     LOAD();
     if( sysout) System.out.println( "%LOAD() is OK!");
     for( i=0; i<MSS; i++) ISS[i]= _ISS[i];
     DECOM();
     if( sysout) System.out.println( "%DECOM() is OK!");
     NLC= 1;
     SOLVER();
     if( sysout) System.out.println( "%SOLVER() is OK!");
     UpdateCoorN();
     if( sysout) System.out.println( "%UpdateCoorN() is OK!");

     return;
}

void INIT()
{   int i;
    ICOOR= new Coor[NNOD];
    ICOORN= new Coor[NNOD];

    for( i=0; i<NNOD; i++) ICOORN[i]= new Coor( 0.,0.,0.);

    INFIX= new int[NNOD][6];                //boundary conditions
    IEXLD= new double[NNOD][6];             //external nodal loads
    IFELD= new double[NNOD][6];             //fixed end loads
    distributedLoad=new DistributedLoad[NDISTL];

    if( NBEAM != 0)
      { beam= new Beam( this, NBEAM);
        beam.INIT();
      }

      if( NTRUS != 0)
      { truss= new Truss( this, NTRUS);
        truss.INIT();
      }
    if( NELSS !=0)
      { elasticSupport= new ElasticSupport( this, NELSS);
        elasticSupport.INIT();
      }

    IPROP= new Prop[NMAT];
    ISECT= new Sect[NSEC];

    io.INIT2( ICOOR, ICOORN, INFIX, IEXLD, IFELD, IPROP, ISECT, beam,
              truss, distributedLoad, elasticSupport);

    if( view!=null) view.INIT2( ICOOR, ICOORN, INFIX, IEXLD, IFELD,
                                IPROP, ISECT, beam, truss,
                                distributedLoad, elasticSupport);
    return;
} // end of INIT()

int IDMAT()
{   int i, j, n;

      n=1;

      for( i=0; i<NNOD; i++)
         for( j=0; j<6; j++)
          if( INFIX[i][j]== 0 ) INFIX[i][j]= n++;
            else INFIX[i][j]= 0;

      return --n;
}  // end of IDMAT

/******************** SKYLINE *******************/
void SKYLINE()
{   int i,j,ii,jj, n, Imin;

    NONZ= new int[NEQ];
    NADD= new int[NEQ];

    for( i=0; i<NEQ; i++) NONZ[i]=i+1;

    if( beam!=null) beam.SKYLINE( IIDND, NONZ, NEQ);

    if( truss!=null) truss.SKYLINE( IIDND, NONZ, NEQ);

    if( elasticSupport!=null) elasticSupport.SKYLINE( IIDND, NONZ, NEQ);

    NADD[0]=0;
    MHB=0;
      for( i=1; i<NEQ; i++)  NADD[i]= NADD[i-1]+ i + 1 - NONZ[i];
      MSS= NADD[NEQ-1]+ NEQ;
      for( i=0; i<NEQ; i++)  if( (n=i-NONZ[i]+2)>MHB) MHB=n;
      return;
}

void CLEAR( double r[], int n)
{   int i;
    for( i=0; i<n; i++) r[i] = 0.;
    return;}

/******************** LOAD *******************/
void LOAD()
{     int i,j,k;
      for( i=0; i<NNOD; i++)
         for( j=0; j<6; j++)
            { if( (k=IIDND[i][j]) == 0) continue;
              IPP[k-1]= IEXLD[i][j]+ IFELD[i][j];
            }
      return;
```

```
}

/******************* DECOM ******************/
void DECOM()
{     int i,j,k,l,I1,JA,JB,MIB,KA,MX;

      INDEX= 0;
      ISIGN= 0;
      if( Math.abs( ISS[0])< EXPL)
        { System.out.println("\n* Diagonal  element s(1)= 0!");
          ISIGN= 1;
          return;}
      else if( ISS[0]< 0.) INDEX++;

      for( i=1; i<MHB; i++)
         if( NONZ[i] == 1)
           { I1= NADD[i];
               ISS[I1] /= ISS[0];}

      for( j=1; j<NEQ; j++)
         { JA= NADD[j]+j;
           for( l=NONZ[j]-1; l<j; l++)
              { JB= NADD[j]+ l;
                ISS[JA] -= ISS[NADD[l]+l]*ISS[JB]*ISS[JB];
              }
           if( Math.abs(ISS[JA])< EXPL)
             { ISIGN= 1;
               return;
             }
           else if( ISS[JA]< 0) INDEX++;

           if( j == NEQ ) return;

           MIB= Math.min( j+MHB, NEQ);
           for( k=j+1; k<MIB; k++)
            { if( NONZ[k] <= j+1)
                { KA= NADD[k]+j;
                  MX= Math.max( NONZ[k], NONZ[j])-1;
                  for( l=MX; l<j; l++)
                   ISS[KA] -=
ISS[NADD[l]+l]*ISS[NADD[j]+l]*ISS[NADD[k]+l];
                  ISS[KA] /= ISS[JA];}}}
      return;}


/******************* SOLVER ******************/
void SOLVER()
{     int i,j,k,ni,nj,NPS,MIB;
      for( i=1; i<NEQ; i++)
         for( j=NONZ[i]-1; j<i; j++)
            for( k=0; k<NLC; k++)
               { ni= i*NLC + k;
                 nj= j*NLC + k;
                 IPP[ni] -= ISS[NADD[i]+j]*IPP[nj];}

      for( i=0; i<NEQ; i++)
         { NPS=NADD[i]+i;
           for( k=0; k<NLC; k++)
              { ni= i*NLC +k;
                IPP[ni] /= ISS[NPS];}}
      for( i=NEQ-1; i>=0; i--)
         { MIB= Math.min( i+MHB, NEQ);
           for( j=i+1; j<MIB; j++)
            if( NONZ[j]-1 <= i)
              { for( k=0; k<NLC;k++)
                   { ni= i*NLC + k;
                     nj= j*NLC + k;
                     IPP[ni] -= ISS[NADD[j]+i]*IPP[nj];}}}
      return;}

/********** Update Coordinate of Nodes **********/
void UpdateCoorN()
{   int i;

    for( i=0; i<NNOD; i++)
      {
         ICOORN[i].x= ICOOR[i].x;
         ICOORN[i].y= ICOOR[i].y;
         ICOORN[i].z= ICOOR[i].z;

         if( INFIX[i][0] !=0) ICOORN[i].x += IPP[ INFIX[i][0]-1];
         if( INFIX[i][1] !=0) ICOORN[i].y += IPP[ INFIX[i][1]-1];
         if( INFIX[i][2] !=0) ICOORN[i].z += IPP[ INFIX[i][2]-1];
      }
}

// static member functions
public static double Length( double x[])
{   return Math.sqrt( x[0]*x[0]+ x[1]*x[1] + x[2]*x[2]);}

public static void Normalize( double x[])
{   double L;
    L= Math.sqrt( x[0]*x[0]+ x[1]*x[1] + x[2]*x[2]);
    x[0] /= L;
    x[1] /= L;
    x[2] /= L;
    return;
}

public static void Crossproduct( double z[], double x[], double y[])
{     z[0]= x[1]*y[2]- x[2]*y[1];
      z[1]= x[2]*y[0]- x[0]*y[2];
      z[2]= x[0]*y[1]- x[1]*y[0];
      return;
}

public static void MV( double V[], double M[][], double U[])   // V=M*U;
{   int i, j;
    double sum;

    for( i=0; i<3; i++)
       { sum=0.;
         for( j=0; j<3; j++) sum+= M[i][j]*U[j];
         V[i]= sum;
       }
}

} // End of class FEMAPP;
```

fem.core.FEMDMA.java

```
package fem.core;

public class FEMDMA {
// group1
public int NNOD, NMAT, NSEC, NBEAM, NTRUS, NCST, NELSS, NDISTL;

public Coor   ICOOR[], ICOORN[], sICOORN[];
public double IEXLD[][], IFELD[][];
public Prop   IPROP[];
public Sect   ISECT[];

public DistributedLoad distributedLoad[];

public Beam beam;
public Truss truss;
public ElasticSupport elasticSupport;

FEMAPP femapp;

public int INFIX[][];

// group2
public int NEQ;

public double _ISS[], IPP[];

public FEMDMA( FEMAPP femapp)
{   this.femapp=femapp;}

public void INIT1( int nnod, int nmat, int nsec, int nbeam, int ntrus,
int ncst, int nelss,
                   int ndistl)
{
    NNOD=  nnod;         //number of nodes
    NMAT=  nmat;         //number of materials
    NSEC=  nsec;         //number of sections
    NBEAM= nbeam;        //number of beams
    NTRUS= ntrus;        //number of truss
    NCST=  ncst;         //number of constraints
    NELSS= nelss;        //number of elastic support
    NDISTL= ndistl;      //number of distributed load

}

public void INIT2( Coor icoor[], Coor icoorn[], int infix[][], double
iexld[][], double ifeld[][],
                   Prop iprop[], Sect isect[], Beam vbeam, Truss vtruss,
                   DistributedLoad distl[],  ElasticSupport velss)
{
    ICOOR=  icoor;
    ICOORN= icoorn;
    INFIX=  infix;
    IEXLD=  iexld;
    IFELD=  ifeld;
    IPROP=  iprop;
    ISECT=  isect;
    beam=   vbeam;
    truss=  vtruss;
    distributedLoad= distl;
    elasticSupport= velss;

    return;
}

public void INIT3( double _iss[], double ipp[], int neq)
{
    _ISS=_iss;
    IPP= ipp;
    NEQ= neq;
    return;
}

} // end of class FEMDA;
```

fem.core.FEMIO.java

```
package fem.core;

public class FEMIO extends FEMDMA{

boolean sysout=false;

public FEMIO( FEMAPP femapp)
{   super( femapp);}

// input functions
// public member functions
public void setNfix( int i, int N1, int N2, int N3, int N4, int N5, int
N6)
{  INFIX[i][0]=N1;
   INFIX[i][1]=N2;
   INFIX[i][2]=N3;
   INFIX[i][3]=N4;
   INFIX[i][4]=N5;
   INFIX[i][5]=N6;
}

public void setBoundaryConditions( java.util.Vector v)
{   int i, ni;
    Idbm bm;

 // Set Boundary Conditons // 0:free, -1:fixed, -2:may not be concerned
    for( i=0; i<NNOD; i++) setNfix(  i,  0,  0, -1, -1, -1, -2);

    for( i=v.size()-1; i>=0; i--)
       { ni= ((Idbc)v.elementAt(i)).Ni;

         switch( ((Idbc)v.elementAt(i)).type) {
            case Idbc.rollerX:
                 setNfix( ni,  0, -1, -1, -1, -1, -2); break;
            case Idbc.rollerY:
                 setNfix( ni, -1,  0, -1, -1, -1, -2); break;
            case Idbc.hingeZ:
                 setNfix( ni, -1, -1, -1, -1, -1, -2); break;
            case Idbc.fixed:
                 setNfix( ni, -1, -1, -1, -1, -1, -1); break;
         }
       }
    for( i=0; i<NBEAM; i++)
       { bm= beam.IIDBM[i];
         ni= bm.Ni-1;
         if( INFIX[ni][5]== -2) INFIX[ni][5]=0;
         ni= bm.Nj-1;
         if( INFIX[ni][5]== -2) INFIX[ni][5]=0;
       }

    for( i=0; i<NNOD; i++) if( INFIX[i][5]== -2){ INFIX[i][5]=-1;}

}

// concentrated moment not support right now
public void setPointLoad( int i, double a, double VX, double VY, double
VZ)
{   int ni, nj;

    int IB, j;

    double ROT[][], TRN[][];
    double PV[], PM[], V[], M[];
    double FixI[][], FixJ[][];
    double RL, b;

    b=1.0-a;

    ROT= new double[3][3];
    TRN= new double[3][3];
    PV= new double[3];
    V=  new double[3];

    FixI= new double[2][3];
    FixJ= new double[2][3];

    V[0]=VX; V[1]=VY; V[2]=VZ;

    RL= beam.ROTATE( i, ROT, TRN);

    FEMAPP.MV( PV, ROT, V);

    FixI[0][0]= PV[0]*b;
    FixI[0][1]= PV[1]*b*b*(3.*a+b);
    FixI[0][2]= PV[2]*b*b*(3.*a+b);
    FixI[1][0]= 0.;
    FixI[1][1]= -PV[2]*a*b*b*RL;
    FixI[1][2]= PV[1]*a*b*b*RL;

    FixJ[0][0]= PV[0]*a;
    FixJ[0][1]= PV[1]*a*a*(a+3.*b);
    FixJ[0][2]= PV[2]*a*a*(a+3.*b);
    FixJ[1][0]= 0.;
    FixJ[1][1]= PV[2]*a*a*b*RL;
    FixJ[1][2]= -PV[1]*a*a*b*RL;

    ni=beam.IIDBM[i].Ni-1;
    nj=beam.IIDBM[i].Nj-1;

    FEMAPP.MV( V, TRN, FixI[0]);
    IFELD[ni][0]+= V[0];
    IFELD[ni][1]+= V[1];
```

```
        IFELD[ni][2]+= V[2];

        FEMAPP.MV( V, TRN, FixI[1]);
        IFELD[ni][3]+= V[0];
        IFELD[ni][4]+= V[1];
        IFELD[ni][5]+= V[2];

        FEMAPP.MV( V, TRN, FixJ[0]);
        IFELD[nj][0]+= V[0];
        IFELD[nj][1]+= V[1];
        IFELD[nj][2]+= V[2];

        FEMAPP.MV( V, TRN, FixJ[1]);
        IFELD[nj][3]+= V[0];
        IFELD[nj][4]+= V[1];
        IFELD[nj][5]+= V[2];

    }

    // local coordinate
    public void setPointLoad2( int i, double a, double VX, double VY, double
    VZ)
    {   int ni, nj;

        int IB, j;

        double ROT[][], TRN[][];
        double V[], M[];
        double FixI[][], FixJ[][];
        double RL, b;

        b=1.0-a;

        ROT= new double[3][3];
        TRN= new double[3][3];
        V=  new double[3];

        FixI= new double[2][3];
        FixJ= new double[2][3];

        V[0]=VX; V[1]=VY; V[2]=VZ;

        RL= beam.ROTATE( i, ROT, TRN);

        FixI[0][0]= V[0]*b;
        FixI[0][1]= V[1]*b*b*(3.*a+b);
        FixI[0][2]= V[2]*b*b*(3.*a+b);
        FixI[1][0]= 0.;
        FixI[1][1]= -V[2]*a*b*b*RL;
        FixI[1][2]= V[1]*a*b*b*RL;

        FixJ[0][0]= V[0]*a;
        FixJ[0][1]= V[1]*a*a*(a+3.*b);
        FixJ[0][2]= V[2]*a*a*(a+3.*b);
        FixJ[1][0]= 0.;
        FixJ[1][1]= V[2]*a*a*b*RL;
        FixJ[1][2]= -V[1]*a*a*b*RL;

        ni=beam.IIDBM[i].Ni-1;
        nj=beam.IIDBM[i].Nj-1;
        FEMAPP.MV( V, TRN, FixI[0]);
        IFELD[ni][0]+= V[0];
        IFELD[ni][1]+= V[1];
        IFELD[ni][2]+= V[2];

        FEMAPP.MV( V, TRN, FixI[1]);
        IFELD[ni][3]+= V[0];
        IFELD[ni][4]+= V[1];
        IFELD[ni][5]+= V[2];

        FEMAPP.MV( V, TRN, FixJ[0]);
        IFELD[nj][0]+= V[0];
        IFELD[nj][1]+= V[1];
        IFELD[nj][2]+= V[2];

        FEMAPP.MV( V, TRN, FixJ[1]);
        IFELD[nj][3]+= V[0];
        IFELD[nj][4]+= V[1];
        IFELD[nj][5]+= V[2];

    }

    // distributed moment not support right now
    public void setDistributedLoad( int i, int ib, double VX, double VY,
    double VZ)
    {   int ni, nj;

        int IB, j;

        double ROT[][], TRN[][];
        double PV[], PM[], V[], M[];
        double FixI[][], FixJ[][];
        double RL;

        distributedLoad[i]= new DistributedLoad( ib, VX, VY, VZ);

        ROT= new double[3][3];
        TRN= new double[3][3];
        PV= new double[3];
        V=  new double[3];

        FixI= new double[2][3];
        FixJ= new double[2][3];

        V[0]=VX; V[1]=VY; V[2]=VZ;

        RL= beam.ROTATE( ib, ROT, TRN);

        FEMAPP.MV( PV, ROT, V);

        FixI[0][0]= PV[0]*RL/2.;
        FixI[0][1]= PV[1]*RL/2.;
        FixI[0][2]= PV[2]*RL/2.;
        FixI[1][0]= 0.;
        FixI[1][1]= -PV[2]*RL*RL/12.;
        FixI[1][2]= PV[1]*RL*RL/12.;

        FixJ[0][0]= PV[0]*RL/2.;
        FixJ[0][1]= PV[1]*RL/2.;
        FixJ[0][2]= PV[2]*RL/2.;
        FixJ[1][0]= 0.;
        FixJ[1][1]= PV[2]*RL*RL/12.;
        FixJ[1][2]= -PV[1]*RL*RL/12.;

        ni=beam.IIDBM[ib].Ni-1;
        nj=beam.IIDBM[ib].Nj-1;

        FEMAPP.MV( V, TRN, FixI[0]);
        IFELD[ni][0]+= V[0];
        IFELD[ni][1]+= V[1];
        IFELD[ni][2]+= V[2];

        FEMAPP.MV( V, TRN, FixI[1]);
        IFELD[ni][3]+= V[0];
        IFELD[ni][4]+= V[1];
        IFELD[ni][5]+= V[2];

        FEMAPP.MV( V, TRN, FixJ[0]);
        IFELD[nj][0]+= V[0];
        IFELD[nj][1]+= V[1];
        IFELD[nj][2]+= V[2];

        FEMAPP.MV( V, TRN, FixJ[1]);
        IFELD[nj][3]+= V[0];
        IFELD[nj][4]+= V[1];
        IFELD[nj][5]+= V[2];

    }

    // for Local Coordinate System
    public void setDistributedLoad2( int i, double VX, double VY, double VZ)
    {   int ni, nj;

        int IB, j;

        double ROT[][], TRN[][];
        double V[], M[];
        double FixI[][], FixJ[][];
        double RL;

        ROT= new double[3][3];
        TRN= new double[3][3];
        V=  new double[3];

        FixI= new double[2][3];
        FixJ= new double[2][3];

        V[0]=VX; V[1]=VY; V[2]=VZ;

        RL= beam.ROTATE( i, ROT, TRN);

        FixI[0][0]= V[0]*RL/2.;
        FixI[0][1]= V[1]*RL/2.;
        FixI[0][2]= V[2]*RL/2.;
        FixI[1][0]= 0.;
        FixI[1][1]= -V[2]*RL*RL/12.;
        FixI[1][2]= V[1]*RL*RL/12.;

        FixJ[0][0]= V[0]*RL/2.;
        FixJ[0][1]= V[1]*RL/2.;
        FixJ[0][2]= V[2]*RL/2.;
        FixJ[1][0]= 0.;
```

```
      FixJ[1][1]= V[2]*RL*RL/12.;
      FixJ[1][2]= -V[1]*RL*RL/12.;

      ni=beam.IIDBM[i].Ni-1;
      nj=beam.IIDBM[i].Nj-1;

      FEMAPP.MV( V, TRN, FixI[0]);
      IFELD[ni][0]+= V[0];
      IFELD[ni][1]+= V[1];
      IFELD[ni][2]+= V[2];

      FEMAPP.MV( V, TRN, FixI[1]);
      IFELD[ni][3]+= V[0];
      IFELD[ni][4]+= V[1];
      IFELD[ni][5]+= V[2];

      FEMAPP.MV( V, TRN, FixJ[0]);
      IFELD[nj][0]+= V[0];
      IFELD[nj][1]+= V[1];
      IFELD[nj][2]+= V[2];

      FEMAPP.MV( V, TRN, FixJ[1]);
      IFELD[nj][3]+= V[0];
      IFELD[nj][4]+= V[1];
      IFELD[nj][5]+= V[2];

}

public void setExld( int i, double VX, double VY, double VZ, double MX,
                      double MY, double MZ)
{ IEXLD[i][0]=VX;
  IEXLD[i][1]=VY;
  IEXLD[i][2]=VZ;
  IEXLD[i][3]=MX;
  IEXLD[i][4]=MY;
  IEXLD[i][5]=MZ;
}

public void setExld( int i, double VX, double VY, double MZ)
{ IEXLD[i][0]=VX;
  IEXLD[i][1]=VY;
  IEXLD[i][2]=0.;
  IEXLD[i][3]=0.;
  IEXLD[i][4]=0.;
  IEXLD[i][5]=MZ;
}

public void addExld( int i, double VX, double VY, double MZ)
{ IEXLD[i][0]+=VX;
  IEXLD[i][1]+=VY;
  IEXLD[i][5]+=MZ;
}

public void setExld( int i)
{ IEXLD[i][0]=0.;
  IEXLD[i][1]=0.;
  IEXLD[i][2]=0.;
  IEXLD[i][3]=0.;
  IEXLD[i][4]=0.;
  IEXLD[i][5]=0.;
}
```

```
public void clearLoad()
{   int i, j;
    for( i=0; i<NNOD; i++)
       for( j=0; j<6; j++)
          { IEXLD[i][0]=0.; IFELD[i][j]=0.;}
}

public void setNode( java.util.Vector v)
{
    for( int i=v.size()-1; i>=0; i--) ICOOR[i]= (Coor)v.elementAt(i);
}

// setIdbm( #beam+1, #node_i, #node_j, mat, sect, ...)
public void setIdbm( int i, int NI, int NJ, int MAT, int SEC, int B1,
                     int B2)
{   beam.IIDBM[i]= new Idbm( NI, NJ, MAT, SEC, B1, B2);
}

public void setIdbm( java.util.Vector v)
{   int i, ni, nj;
    Idbm bm;

    for( i=v.size()-1; i>=0; i--)
       { beam.IIDBM[i]= bm= (Idbm)v.elementAt(i);

         ni=bm.Ni-1;
         nj=bm.Nj-1;

         if( Math.abs( ICOOR[ni].x-ICOOR[nj].x) >
             Math.abs(ICOOR[ni].y-ICOOR[nj].y))
           { setVect( i, 0, 1, 0);}
         else
           { setVect( i, 1, 0, 0);}
       }

}

// setIdtr( #truss+1, #node_i, #node_j, mat, sect, ...)
public void setIdtr( int i, int NI, int NJ, int MAT, int SEC, int B1,
                     int B2)
{   truss.IIDTR[i]= new Idtr( NI, NJ, MAT, SEC, B1, B2);
}

public void setIdtr( java.util.Vector v)
{
    for( int i=v.size()-1; i>=0; i--) truss.IIDTR[i]=
(Idtr)v.elementAt(i);
}

public void setIdelss( int i, int node, int dof, double k)
{   elasticSupport.IDELSS[i]= new Idelss( node, dof, k);}

public void setIdelss( java.util.Vector v)
{
    for( int i=v.size()-1; i>=0; i--)
         elasticSupport.IDELSS[i]= (Idelss)v.elementAt(i);
}

public void setVect( int i, double X, double Y, double Z)
{   beam.IVECT[i][0]=X;
```

```
    beam.IVECT[i][1]=Y;
    beam.IVECT[i][2]=Z;
}

public void setProp( int i, double ME, double MP, double B1, double B2)
{   IPROP[i-1]= new Prop( ME, MP, B1, B2);
}

public void setProp( java.util.Vector v)
{
    for( int i=v.size()-1; i>=0; i--) IPROP[i]= (Prop)v.elementAt(i);
}

public void setSect( int i, double a, double IY, double IZ, double j,
                     double B1, double B2)
{   ISECT[i-1]= new Sect( a, IY, IZ, j, B1, B2);
}

public void setSect( java.util.Vector v)
{
    for( int i=v.size()-1; i>=0; i--) ISECT[i]= (Sect)v.elementAt(i);
}

//======================================================================
/********************/
/* output functions */
/********************/
public static void mprintf( String s1)
{ System.out.println( s1);}

public static void mprintf( String s1, int i1)
{ System.out.println( s1+" "+String.valueOf(i1));}

public static void mprintf( String s1, int i1, String s2, int i2)
{ System.out.println( s1+" "+String.valueOf(i1)+" "+
                      s2+" "+String.valueOf(i2));
}

public static void mprintf( String s1, int i1, String s2, double i2)
{ System.out.println( s1+" "+String.valueOf(i1)+" "+
                      s2+" "+String.valueOf(i2));
}

public static void mprintf( double i1, double i2)
{ System.out.println( String.valueOf(i1)+" "+
                      String.valueOf(i2));
}

public static void mprintf( double i1, double i2, double i3)
{ System.out.println( String.valueOf(i1)+" "+
                      String.valueOf(i2)+" "+
                      String.valueOf(i3));
}

public static void mprintf( int i1, int i2, int i3)
{ System.out.println( String.valueOf(i1)+" "+
                      String.valueOf(i2)+" "+
                      String.valueOf(i3));
}
```

```
public static void mprintf( int i1, int i2, int i3, int i4, int i5, int
i6)
{ System.out.println( String.valueOf(i1)+" "+
                      String.valueOf(i2)+" "+
                      String.valueOf(i3)+" "+
                      String.valueOf(i4)+" "+
                      String.valueOf(i5)+" "+
                      String.valueOf(i6));
}

public static void mprintf( double i1, double i2, double i3, double i4,
double i5, double i6)
{ system.out.println( String.valueOf(i1)+" "+
                        String.valueOf(i2)+" "+
                        String.valueOf(i3)+" "+
                        String.valueOf(i4)+" "+
                        String.valueOf(i5)+" "+
                        String.valueOf(i6));
}

public void output()
{   int i,j;
    double tmp;
    mprintf( "\n%Output data");
    mprintf( "NNOD",NNOD);
    mprintf( "NMAT",NMAT,"NSEC",NSEC);

    mprintf( "NBEAM",NBEAM,"NTRUS",NTRUS );

    mprintf( "\n%Nodal coordinates");
    mprintf( "      X          Y          Z");
    for( i=0; i<NNOD; i++)
       mprintf( ICOOR[i].x, ICOOR[i].y, ICOOR[i].z);

    mprintf( "\n%Direction of the the y-axis of the beam");
    mprintf( "      X          Y          Z");

    mprintf( "\n  Fixity of the structural nodes, 0 for restrained\n");
    mprintf( "\t\t     X   Y   Z  éx  éy  éz");
    for( i=0; i<NNOD; i++)
       mprintf( INFIX[i][0], INFIX[i][1],
                INFIX[i][2], INFIX[i][3], INFIX[i][4], INFIX[i][5]);

    mprintf( "\n     DISPLACEMENT\n");
    for( i=0; i<NEQ; i++) mprintf( " ",i, ":", IPP[i]);

    return;
}

public Coor getNodalDisplacement( int i)
{
    return new Coor( ICOORN[i].x-ICOOR[i].x,
                     ICOORN[i].y-ICOOR[i].y,
                     ICOORN[i].z-ICOOR[i].z);

}

// ========================================
public void solveMatrix()
{   femapp.solveMatrix();}
```

```
public void setupMatrix()
{   femapp.setupMatrix();}

public void setupMatrix2()
{   femapp.setupMatrix2();}


} //end of class FEMIO;
```

fem.core.FEMVIEW.java

```
package fem.core;

import java.awt.*;

public class FEMVIEW extends FEMDMA{

double Beamwid=.2;

// If drawScale != 1, geometry of the frame, that shown on the screen
may not be as the same as the real one.
double drawScale= 1.;
double forceColorScale= 0.01;
double fForce= 1.;

Engineer jcc;

public FEMVIEW( FEMAPP femapp, Engineer jcc)
{
    super( femapp);
    this.jcc=jcc;
}

public void updatesICOORN()
{   int i;

    if( sICOORN== null)
      { sICOORN= new Coor[NNOD];
        for( i=0; i<NNOD; i++)
           { sICOORN[i]= new Coor( (ICOORN[i].x - ICOOR[i].x)*drawScale
+ ICOOR[i].x,
                                   (ICOORN[i].y - ICOOR[i].y)*drawScale
+ ICOOR[i].y,
                                   (ICOORN[i].z - ICOOR[i].z)*drawScale
+ ICOOR[i].z);
           }
      }
    else
      { for( i=0; i<NNOD; i++)
           {  sICOORN[i].x= (ICOORN[i].x - ICOOR[i].x)*drawScale +
ICOOR[i].x;
              sICOORN[i].y= (ICOORN[i].y - ICOOR[i].y)*drawScale +
ICOOR[i].y;
              sICOORN[i].z= (ICOORN[i].z - ICOOR[i].z)*drawScale +
ICOOR[i].z;
           }
      }
}
```

```
public void drawBoundaryCondition( Graphics g)
{   int i;

    // now just support 2D, hinge, roller, fixed, free
    for( i=0; i<NNOD; i++)
       { if( INFIX[i][5]>0)
           { if( INFIX[i][0]==0)
               { if( INFIX[i][1]==0 )   // hinge
                   { g.setColor(Color.blue);
                     jcc.f_rect( g, sICOORN[i].x-1.0, sICOORN[i].y,
sICOORN[i].x+1.0, sICOORN[i].y-0.5);
                   }
                 else  // y-roller
                   { g.setColor(Color.cyan);
                     jcc.f_rect( g, sICOORN[i].x, sICOORN[i].y-1.,
sICOORN[i].x+.5, sICOORN[i].y+1.);
                   }
               }
             else
               { if( INFIX[i][1]==0 )   // x-roller
                   { g.setColor(Color.cyan);
                     jcc.f_rect( g, sICOORN[i].x-1.0, sICOORN[i].y,
sICOORN[i].x+1.0, sICOORN[i].y-0.5);
                   }
                 else continue;// free
               }
           }
         else
           { if( INFIX[i][0]==0)
               { if( INFIX[i][1]==0 )   // fixed
                   { g.setColor(Color.red);
                     jcc.f_rect( g, sICOORN[i].x-1.0, sICOORN[i].y,
sICOORN[i].x+1.0, sICOORN[i].y-0.5);
                   }
                 else  // y-roller
                   { g.setColor(Color.cyan);
                     jcc.f_rect( g, sICOORN[i].x, sICOORN[i].y-1.,
sICOORN[i].x+.5, sICOORN[i].y+1.);
                   }
               }
             else
               { if( INFIX[i][1]==0 )   // x-roller
                   { g.setColor(Color.cyan);
                     jcc.f_rect( g, sICOORN[i].x-1.0, sICOORN[i].y,
sICOORN[i].x+1.0, sICOORN[i].y-0.5);
                   }
                 else continue;// free
               }
           }
       }

}

public void drawMembers( Graphics g)
{     int i;
// Draw Truss and Beam
      Vector2D bS1, bS2, dS1, dS2, fS1;
      int nS1, nS2;
      double x, y, dx, dy, L, N;

      // Displacement IPP[]:ft
      for( i=0; i< NBEAM; i++)
        {
             nS1= beam.IIDBM[i].Ni-1;
             nS2= beam.IIDBM[i].Nj-1;

             bS1= new Vector2D( ICOOR[nS1].x, ICOOR[nS1].y);
             bS2= new Vector2D( ICOOR[nS2].x, ICOOR[nS2].y);

             dS1= new Vector2D( sICOORN[nS1].x, sICOORN[nS1].y);
             dS2= new Vector2D( sICOORN[nS2].x, sICOORN[nS2].y);

             L=(bS1.Minus(bS2)).Abs();

             N=(dS1.Minus(dS2)).Abs();

             drawBeam( g, dS1, dS2, Beamwid, DispColor(N/L));

        }

      for( i=0; i< NTRUS; i++)
        {
             nS1= truss.IIDTR[i].Ni-1;
             nS2= truss.IIDTR[i].Nj-1;

             dS1= new Vector2D( sICOORN[nS1].x, sICOORN[nS1].y);
             dS2= new Vector2D( sICOORN[nS2].x, sICOORN[nS2].y);

             drawBeam( g, dS1, dS2, Beamwid, DispColor(
truss.ElementForce(i,IPP)*forceColorScale+1.0));

        }

      for( i=0; i< NNOD; i++)
         {
           bS1= new Vector2D( sICOORN[i].x, sICOORN[i].y);

           fS1= new Vector2D( IEXLD[i][0]*fForce, IEXLD[i][1]*fForce);

           if( fS1.Abs()> FEMAPP.EXPL) jcc.arrow(g,bS1,bS1.Plus(fS1),2);
         }

      for( i=0; i< NDISTL; i++)
         {
           nS1= beam.IIDBM[distributedLoad[i].ib].Ni-1;
           nS2= beam.IIDBM[distributedLoad[i].ib].Nj-1;

           dS1= new Vector2D( sICOORN[nS1].x, sICOORN[nS1].y);
           dS2= new Vector2D( sICOORN[nS2].x, sICOORN[nS2].y);

           drawDistributedLoad(g,dS1,dS2, Color.black);
         }

}

public void drawMembers2( Graphics g)
{     int i;

      Vector2D bS1, bS2, dS1, dS2, fS1;
      int nS1, nS2;
      double x, y, dx, dy, L, N;
      // Draw Truss and Beam
      // Displacement IPP[]:ft
      for( i=0; i< NBEAM; i++)
        {
             nS1= beam.IIDBM[i].Ni-1;
             nS2= beam.IIDBM[i].Nj-1;

             bS1= new Vector2D( ICOOR[nS1].x, ICOOR[nS1].y);
             bS2= new Vector2D( ICOOR[nS2].x, ICOOR[nS2].y);

             drawBeam( g, bS1, bS2, 0.1, Color.black);
        }

      for( i=0; i< NTRUS; i++)
        {
             nS1= truss.IIDTR[i].Ni-1;
             nS2= truss.IIDTR[i].Nj-1;

             bS1= new Vector2D( ICOORN[nS1].x, ICOORN[nS1].y);
             bS2= new Vector2D( ICOORN[nS2].x, ICOORN[nS2].y);

             drawBeam( g, bS1, bS2, 0.1, Color.black);
      }
}

public void drawSymbols( Graphics g)
{   int i;

    for( i=0; i< NNOD; i++)
       jcc.symbol( g, sICOORN[i].x-0.6, sICOORN[i].y-0.6,
                   String.valueOf(i));

}

public void drawBeam( Graphics g, Vector2D x0, Vector2D x1,
                      double width, Color CL)
{
          Vector2D Unit01=new Vector2D((x1.Minus(x0)).Unit());
          double Hwid=width*.5;
          Vector2D Tang01=new Vector2D(Unit01.Mult(Hwid));
          Vector2D Norm01=new Vector2D((Unit01.Rot(90.)).Mult(Hwid));

          Vector2D v[]=new Vector2D[8];
          v[0]=(x0.Plus(Norm01)).Plus(Tang01.Mult(-.5));
          v[1]=(x0.Plus(Norm01.Mult(+.5))).Minus(Tang01);
          v[2]=(x0.Plus(Norm01.Mult(-.5))).Minus(Tang01);
          v[3]=(x0.Minus(Norm01)).Plus(Tang01.Mult(-.5));
          v[4]=(x1.Minus(Norm01)).Plus(Tang01.Mult(+.5));
          v[5]=(x1.Plus(Norm01.Mult(-.5))).Plus(Tang01);
          v[6]=(x1.Plus(Norm01.Mult(+.5))).Plus(Tang01);
          v[7]=(x1.Plus(Norm01)).Plus(Tang01.Mult(+.5));
          Poly2DPaint(g,CL, v,8);

          g.setColor(Color.black);
          jcc.u_plot(g,v[0],3);
          for(int i=1;i<8;i++) jcc.u_plot(g,v[i],2);
          jcc.u_plot(g,v[0],2);
          jcc.marker(g,x0.Get_Elem(0),x0.Get_Elem(1),2);
          jcc.marker(g,x1.Get_Elem(0),x1.Get_Elem(1),2);

  }
```

```
public void drawElasticSupport( Graphics g)
{   int i;
    Vector2D dS1, dS2;
    int nS;
    double x, y;

    // Draw Elastic Support
    // Displacement IPP[]:ft
    for( i=0; i< NELSS; i++)
      {
        nS= elasticSupport.IDELSS[i].Ni;

        dS1= new Vector2D( sICOORN[nS].x, sICOORN[nS].y);

        if( elasticSupport.IDELSS[i].type== Idelss.springX)
          dS2= new Vector2D( x=ICOOR[nS].x-1.5, y=ICOOR[nS].y);
        else
          dS2= new Vector2D( x=ICOOR[nS].x, y=ICOOR[nS].y-1.5);

        g.setColor(Color.red);
        jcc.f_rect( g, x-0.2, y-0.2, x+0.2, y+0.2);

        drawSpring( g, dS1, dS2, DispColor(
            elasticSupport.ElementForce(i,IPP)*forceColorScale+1.0));

      }
}

public void drawSpring( Graphics g, Vector2D x0, Vector2D x1, Color CL)
{
    double polyline[][];
    double L, W;

    polyline= new double[7][2];

    Vector2D Unit01=new Vector2D((x1.Minus(x0)).Unit());

    L=-(x0.Minus(x1)).Abs()/7.;
    W=0.15;

    polyline[0][0]= W*0.; polyline[0][1]= L*1.5;
    polyline[1][0]= W*3.; polyline[1][1]= L*2.;
    polyline[2][0]=-W*3.; polyline[2][1]= L*3.;
    polyline[3][0]= W*3.; polyline[3][1]= L*4.;
    polyline[4][0]=-W*3.; polyline[4][1]= L*5.;
    polyline[5][0]= W*0.; polyline[5][1]= L*5.5;
    polyline[6][0]= W*0.; polyline[6][1]= L*7.;

    //rotateScale( polyline, angle, sLength);

    Vector2D v[]=new Vector2D[8];
    v[0]=x0;
    for(int i=1; i<8; i++) v[i]=x0.Plus(new Vector2D(polyline[i-1][0],
                                                 polyline[i-1][1]));

    g.setColor(CL);

    jcc.u_plot(g,v[0],3);
    for(int i=1;i<8;i++) jcc.u_plot(g,v[i],2);
```

```
}

public void drawDistributedLoad( Graphics g, Vector2D x0, Vector2D x1,
                                 Color CL)
{

    Vector2D Unit=new Vector2D((x1.Minus(x0)).Unit());

    Vector2D Norm=new Vector2D((Unit.Rot(90.)).Mult(0.5));

    Vector2D dV=new Vector2D((x1.Minus(x0)).Mult(0.1));

    Vector2D Unit01=new Vector2D((x1.Minus(x0)).Unit());

    Vector2D li[]=new Vector2D[12];
    Vector2D lj[]=new Vector2D[12];

    li[0]=x0.Plus(Norm); lj[0]=x1.Plus(Norm);

    for( int i=1; i<12; i++)
       { lj[i]=x0.Plus( dV.Mult(((double)i-1.)));
         li[i]= lj[i].Plus(Norm);
       }

    g.setColor(CL);

    for( int i=0; i<12; i++)
       { jcc.u_plot(g,li[i],3);
         jcc.u_plot(g,lj[i],2);
       }

}

public void Poly2DPaint(Graphics g,Color Ucolor,Vector2D[] v,int n)
{
        int xPoints[]=new int[10],yPoints[]=new int[10];
        if(n>8)return;

        for(int i=0;i<n;i++)
        {
             jcc.u_plot(g,v[i],3);
             xPoints[i]=(short)jcc.ixold;
             yPoints[i]=(short)jcc.iyold;
        }
        g.setColor(Ucolor);
        g.fillPolygon(xPoints,yPoints,n);
}

public Color DispColor(double a)
{
    float da=7.0f;
    if(Math.abs(a-1.) <= 0.01) return Color.darkGray;

    if(a<1.)
      {
         if(a<1.-1./da) return new Color( 0.2f, 0.2f, 1.0f);
         else return new Color(0.2f,0.2f,(float)(1.-a)*da);
      }
    else
      {
         if(a>1.+1./da) return new Color( 1.0f, 0.2f, 0.2f);
```

```
            else return new Color((float)(a-1.)*da,0.2f,0.2f);
         }
}

}// end of class FEMVIEW;
```

fem.core.Truss.java

```
package fem.core;

public class Truss extends Element{
// data members
int NTRUS;

Idtr IIDTR[];
int  LM2[][];

// external link to the node coordinate
Coor ICOOR[];
Prop IPROP[];
Sect ISECT[];

int  NADD[];
double _ISS[];

fem.core.FEMAPP femapp;

Truss( fem.core.FEMAPP fa, int n)
{
    femapp= fa;
    NTRUS=n;
}

public void INIT()
{
    IIDTR= new Idtr[NTRUS];
      LM2= new int[NTRUS][6];
}

public void SKYLINE( int iidnd[][], int nonz[], int NEQ)
{   int i, j, ii, jj, n, Imin;

      for( i=0; i<NTRUS; i++)
         { ii=IIDTR[i].Ni-1;
           jj=IIDTR[i].Nj-1;
           Imin= NEQ;
           for( j=0; j<3; j++)
              { LM2[i][j]= iidnd[ii][j];
                LM2[i][j+3]= iidnd[jj][j];
              }
           for( j=0; j<6; j++) if( (n= LM2[i][j])!=0 && n< Imin) Imin=n;
             for( j=0; j<6; j++)
                { if( (n=LM2[i][j])!=0 && nonz[n-1]> Imin)
                         nonz[n-1]=Imin;
                }
         }
}

/******************* STIFF ******************/
public void STIFF( Coor icoor[], Prop iprop[], Sect isect[], int nadd[],
                   double _iss[])
{   int IB,i,j;
    double ESL[][], RSR[][];
    double ROT[][], TRN[][];
    double RL;

    ICOOR= icoor;
    IPROP= iprop;
    ISECT= isect;
    NADD=  nadd;
    _ISS=  _iss;

    ESL= new double[6][6];
    RSR= new double[6][6];
    ROT= new double[3][3];
    TRN= new double[3][3];

    for( i=0; i<6; i++) for( j=0; j<6; j++) ESL[i][j]=0.;
    for( IB=0; IB<NTRUS;  IB++)
       { RL= ROTATE( IB, ROT, TRN);
         ELEMENT( IB, RL, ESL);
         TRANS( ROT, TRN, ESL, RSR );

         ASSEM( LM2[IB], RSR);
       }
    return;
}

double ROTATE( int ib, double rot[][],  double trn[][])
{     int Ni, Nj;
      double RL;
      double VECT[];
      int i,j;

      VECT= new double[3];

      Ni= IIDTR[ib].Ni-1;
      Nj= IIDTR[ib].Nj-1;

      rot[0][0]= ICOOR[Nj].x- ICOOR[Ni].x;
      rot[0][1]= ICOOR[Nj].y- ICOOR[Ni].y;
      rot[0][2]= ICOOR[Nj].z- ICOOR[Ni].z;

      RL= fem.core.FEMAPP.Length( rot[0]);
      fem.core.FEMAPP.Normalize( rot[0]);

      if( Math.abs( Math.abs(rot[0][1])-1.0)< fem.core.FEMAPP.EXPL)
        { VECT[0]=1.0; VECT[1]=0.; VECT[2]=0.; }
      else { VECT[0]=0.; VECT[1]=1.0; VECT[2]=0.; }

      fem.core.FEMAPP.Crossproduct( rot[2], rot[0], VECT);
      fem.core.FEMAPP.Normalize( rot[2]);

      fem.core.FEMAPP.Crossproduct( rot[1], rot[2], rot[0]);
      fem.core.FEMAPP.Normalize( rot[1]);

      for( i=0; i<3; i++) for( j=0; j<3; j++) trn[i][j]= rot[j][i];

      return(RL);
}
```

```
void TRANS( double rot[][], double trn[][], double esl[][],
            double rsl[][])
{   int i, j;
    for( i=0; i<6; i+=3)
      for( j=0; j<=i; j+=3) Trans( rot, trn, esl, rsl, i, j);
      return;
}

void Trans( double rot[][], double trn[][], double esl[][],
            double rsl[][], int m, int n)
{   int  i,j,k;
    double sum;
    double tmp[][];
    tmp= new double[3][3];

      for( i=0; i<3; i++)
         for( j=0;  j<3; j++)
            { sum=0;
              for( k=0; k<3; k++) sum += trn[i][k]*esl[k+m][j+n];
              tmp[i][j]= sum;
            }

      for( i=0; i<3; i++)
         for( j=0;  j<3; j++)
            { sum=0;
              for( k=0; k<3; k++) sum += tmp[i][k]*rot[k][j];
              rsl[i+m][j+n]= sum;
            }
      return;
}


void ASSEM( int lm[], double rsr[][])
{     int i,j,Ni,Nj;
      for( i=0; i<6; i++)
         for( j=0; j<=i; j++)
            { if( lm[i]==0 || lm[j]==0 ) continue;
              Ni=lm[i]-1; Nj=lm[j]-1;
              // add stiffness matrix[Ni][Nj];
              if( Ni>= Nj ) _ISS[ NADD[Ni]+Nj] += rsr[i][j];
              else  _ISS[ NADD[Nj]+Ni] += rsr[i][j];
            }
      return;
}

void ELEMENT( int ib, double rl, double esl[][])
{
      int i,j;
      double EE, AREA; //ERIZ, ERIY;
      int mt, st;
      double L, L2, L3;

      L=  rl;
      L2= L*L;
      L3= L2*L;

      mt= IIDTR[ib].Mat-1;
      st= IIDTR[ib].Sec-1;

      EE=    IPROP[mt].mE;
      AREA=  ISECT[st].A;
      //ERIZ=  EE* ISECT[st].Iz;
      //ERIY=  EE* ISECT[st].Iy;

      esl[ 0][ 0]=  EE * AREA / L;
      esl[ 0][ 3]= -esl[0][0];
      esl[ 1][ 1]=  0.;  //12.* ERIZ / L3;
      esl[ 1][ 4]= -esl[1][1];
      esl[ 2][ 2]=  0.;  //12.* ERIY / L3;
      esl[ 2][ 5]= -esl[2][2];

      esl[ 3][ 0]= -esl[0][ 0];
      esl[ 3][ 3]= -esl[0][ 3];
      esl[ 4][ 1]= -esl[1][ 1];
      esl[ 4][ 4]= -esl[1][ 4];
      esl[ 5][ 2]= -esl[2][ 2];
      esl[ 5][ 5]= -esl[2][ 5];

      return;
}

/*********** FORCE ************/
void FORCE( double IPP[])
{   int IB,i,j;
      double ESL[][];
      double ROT[][], TRN[][];
      double EDG[], EDL[], FF[];
      double RL, sum;

      ESL= new double[6][6];
      ROT= new double[3][3];
      TRN= new double[3][3];
      EDG= new double[6];
      EDL= new double[6];
      FF=  new double[6];


      for( i=0; i<6; i++) for( j=0; j<6; j++) ESL[i][j]=0;
      for( IB=0; IB<NTRUS;  IB++)
         { RL= ROTATE( IB, ROT, TRN);
           for( i=0; i<6; i++)
                { if( (j=LM2[IB][i]) == 0) { EDG[i]= 0; continue;}
                  EDG[i]= IPP[j-1];
            }

           for( i=0; i<3; i++)
                { EDL[i]= ROT[i][0]*EDG[0]+ ROT[i][1]*EDG[1]+
ROT[i][2]*EDG[2];
                  EDL[i+3]= ROT[i][0]*EDG[3]+ ROT[i][1]*EDG[4]+
ROT[i][2]*EDG[5];
            }

           ELEMENT( IB, RL, ESL);
           for( i=0; i<6; i++)
                { sum=0;
                  for( j=0; j<6; j++) sum += ESL[i][j] * EDL[j];
                  FF[i]= sum;
            }

        }
    return;
```

```
}

/*********** Element FORCE ************/
public double ElementForce( int IB, double IPP[])
{   int i,j;
    double ESL[][];
    double ROT[][], TRN[][];
    double EDG[], EDL[], FF[];
    double RL, sum;

    ESL= new double[6][6];
    ROT= new double[3][3];
    TRN= new double[3][3];
    EDG= new double[6];
    EDL= new double[6];
    FF=  new double[6];


      for( i=0; i<6; i++) for( j=0; j<6; j++) ESL[i][j]=0.;

      RL= ROTATE( IB, ROT, TRN);
      for( i=0; i<6; i++)
         { if( (j=LM2[IB][i]) == 0) { EDG[i]= 0; continue;}
             EDG[i]= IPP[j-1];
         }

      for( i=0; i<3; i++)
         { EDL[i]= ROT[i][0]*EDG[0]+ ROT[i][1]*EDG[1]+ ROT[i][2]*EDG[2];
           EDL[i+3]=ROT[i][0]*EDG[3]+ ROT[i][1]*EDG[4]+ROT[i][2]*EDG[5];
         }

      ELEMENT( IB, RL, ESL);
      for( i=0; i<6; i++)
         { sum=0;
           for( j=0; j<6; j++) sum += ESL[i][j] * EDL[j];
             FF[i]= sum;
         }

    return FF[0];
}

public double valueOf( int n, int d)
{
    return ElementForce( n, femapp.IPP);
}

} //end of class Truss;
```

FemApplet.html

```
<HTML>
<HEAD>
   <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-
1">
   <META NAME="GENERATOR" CONTENT="Mozilla/4.04 [en] (WinNT; U)
[Netscape]">
   <TITLE>ModelBuilder</TITLE>
</HEAD>
<BODY>
```

```
<B><FONT COLOR="#993300"><FONT SIZE=+1>Structural
Analysis</FONT></FONT></B>
<BR>
<HR>
<TABLE BORDER >
<TR>
<TD><APPLET code="FemApplet.class" id="FemApplet" width="500"
HEIGHT="400"></APPLET></TD>

<TD><B><FONT COLOR="#3333FF">Truss:</FONT>&nbsp;</B>&nbsp;
<BR>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Add truss elements&nbsp;
<BR><B><FONT COLOR="#3333FF">Beam:</FONT></B>&nbsp;
<BR>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Add beam elements&nbsp;
<BR><B><FONT COLOR="#3333FF">B.C.:</FONT></B>&nbsp;
<BR>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Set boundary conditions&nbsp;
<BR><B><FONT COLOR="#3333FF">Load:</FONT></B>&nbsp;
<BR>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Add nodal loads&nbsp;
<BR><B><FONT COLOR="#3333FF">Run:</FONT>&nbsp;</B>&nbsp;
<BR>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Execute FEM analysis&nbsp;
<BR><B><FONT COLOR="#3333FF">Clear:</FONT></B>&nbsp;
<BR>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Clear all the objects&nbsp;

<P><I><FONT COLOR="#FF0000"><FONT SIZE=-1>1.Push the left mouse button,
drag it,</FONT></FONT></I>&nbsp;
<BR><I><FONT COLOR="#FF0000"><FONT SIZE=-1>&nbsp;&nbsp; then release it
to add truss and beam</FONT></FONT></I>&nbsp;
<BR><I><FONT COLOR="#FF0000"><FONT SIZE=-1>&nbsp;
element.</FONT></FONT></I>&nbsp;
<BR><I><FONT COLOR="#FF0000"><FONT SIZE=-1>2. Y is positive in the
downward
direction.</FONT></FONT></I>&nbsp;
<BR><I><FONT COLOR="#FF0000"><FONT SIZE=-1>3.The scrollbar controls the
first load.</FONT></FONT></I></TD>
</TR>

<TR>
<TD COLSPAN="2"><TT>Unit: Force: Kips,
kN&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
Moment: Kips-ft, kN-m&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Area:
in<SUP>2</SUP>,
cm<SUP>2</SUP></TT>&nbsp;
<BR><TT>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Length: ft,
m&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
Displacement: in, cm&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Young's
Modulus: Ksi, MPa</TT>
<BR><TT>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Stiffness: Kips/ft,
kN/m&nbsp;&nbsp;&nbsp;
Moment of Inertia: in<SUP>4</SUP>, cm<SUP>4</SUP></TT></TD>
</TR>
</TABLE>
&nbsp;
</BODY>
</HTML>
```