EasyFEM—An object-oriented graphics interface finite element/finite volume software

S. Phongthanapanich a, P. Dechaumphai b,∗

a Department of Mechanical Engineering, Siam University, Phasi-charoen, Bangkok 10163, Thailand
b Mechanical Engineering Department, Chulalongkorn University, Patumwan, Bangkok 10330, Thailand

Received 23 May 2005; received in revised form 3 January 2006; accepted 16 May 2006
Available online 28 July 2006

Abstract

An object-oriented finite element/finite volume software, EasyFEM, has been developed. The software, with a fully interactive graphics interface, analyses heat transfer, solid mechanics, and fluids problems by the finite element and the finite volume methods. The Coad and Yourdon methodology is used to describe the general structure of classes and objects implemented by the software. Details of the object-oriented classes for both the graphics pre- and post-processors, as well as for the analysis solutions, are described. Several case studies with graphical representations of numerical solutions are presented to illustrate some functionalities of the software.

Keywords: Object-oriented programming; Graphics user interface; Finite element/finite volume software

1. Introduction

Need for engineering software to analyze interdisciplinary problems has increased recently. Fluid/structure interaction problems, in particular, have arisen in manufacturing industries that demand developments of new analysis tools. The finite element method (FEM) has gained its popularity for solving structural problems in the past decades, while the finite volume method (FVM) is being widely employed in computational fluid dynamics for solving fluid problems. Even though both methods start from the same process of domain discretization, but their analyses and computational procedures are different. Combining both methods in single software thus normally creates programming difficulty, mainly from different backgrounds of programmers who have expertise in different fields. In addition, software maintenance including the implementation of new developments can also pose additional difficulties after the software has been used for sometimes.

To normal users of any software, effective graphical interface for both the pre- and post-processors is needed. A number of tasks, such as constructing geometries, discretizing domains, applying loads and constraints, performing analyses, and displaying solutions, must be simple and easy for users. Furthermore, because most users have good backgrounds or experiences only in one discipline, e.g., either structures or fluids, supportive human–computer interaction (HCI) concept should be incorporated in every state of the software analysis and design to increase the learnability for other disciplines. All of the factors described above which include the programming concepts, the different analysis solvers for disciplinary problems, the full graphical interface environment, and the human–computer interaction concept for improved learnability, must be considered prior to developing a software for analyzing complex engineering problems nowadays.

Object-oriented programming (OOP) is a well-suited approach for serious developers and has received the popularity in the computational engineering community

This is a freeware, free copy of software could be requested directly from the corresponding author.

∗ Corresponding author. Tel./fax: +66 2 218 6621.
E-mail address: fmepdc@eng.chula.ac.th (P. Dechaumphai).
recently. The OOP offers many means of avoiding/reducing difficulties encountered during the software development life cycle [1,2]. It also helps improving the software code reusability, modifiability, maintainability, with good documentation. In the past decade, the object-oriented concept has been applied by researchers [3–7] to develop finite element and finite volume programs by using the OOP languages, such as the SmallTalk and C++. Recently, the new trend of software development is the incorporation of a friendly graphics user interface environment to maximize the user learnability through direct visualization. This work proposes a procedure to combine such graphical interface environment with the finite element/finite volume analysis solvers, all with the object-oriented programming approach.

EasyFEM is a prototype graphics interface software developed for analyzing two-dimensional heat transfer, solid mechanics, and fluid flows problems by the finite element/finite volume methods. The software employs the OOP approach for both the user graphics interface and the analysis solution modules. The pre- and post-processors, with friendly graphics user interface environment, allow users to construct arbitrary geometries, discretize computational domains automatically, apply loads and constraints, perform analyses, and display solutions with ease.

The plain console finite element/finite volume application has been widely developed and used by researchers. Even though such the console application is adequate for solving academic type problems but it is not suitable for problems with complex geometry. The pre- and post-graphics interface software is needed to increase the modeling capability and to display solutions effectively. One of the objectives of this paper is to present an idea for designing a pre- and post-graphics interface software that can be used with the existing solvers. The idea is simple and can be implemented by a programmer with moderate OOP background.

The primary objective of this paper is to present the OOP concept used in the development of the EasyFEM graphics interface software for solving disciplinary problems under a friendly graphics user interface environment. The classes and objects classification idea described in this paper is uncomplicated for finite element/finite volume software development and can be used with several kinds of existing solvers with small modification. To reduce the software development effort, the multi-language software should be implemented. The Visual Basic (VB) is employed herein for the user graphics interface because of its simplicity with convenient debugging feature. The Visual C++ (VC++) is used to provide efficient graphics capability, while the Visual Fortran (VF) is employed in the development of the analysis solvers. The data structure organization of the software and the object handlings are described. Several examples of heat transfer, solid mechanics, and fluid flows problems are presented to highlight distinctive numerical solutions and graphics representations of the software.

2. Object-oriented programming in VB/VC++ and general structure of EasyFEM

The OOP is a concept that helps programming easier and more effective by coding software to deal directly with objects in the real world. The object-oriented paradigm consists of the three major components that an OOP language should support. The first component, the encapsulation, is the ability to hide detailed implementation of an object. The inheritance, the second component, is the ability to reuse the existing objects for creating new specialized objects. The third component, the polymorphism, is the ability of the code to exhibit multiple behaviors, or to allow the use of identical function names for logical similar methods in different types [1,8].

It is well-known that VB does not fully support the OOP features, for example, VB handles polymorphism a little differently from the way VC++ does. VB does not use inheritance to provide polymorphism, but VB allows programmers to create an interface, which is a set of related properties and method definitions, in an abstract class. This special class is then declared in class modules that implement the interface by using the Implements keyword. Another important feature that VB does not directly support is the pointer implementation. However, the feature could be added to VB by the help of VC++ and some Windows Application Programming Interface (Win32 API) functions. On the other hand, the benefits of using VB for software development can be easily recognized for its simplicity, less time consuming with complete tools for user interface design as compared to VC++.

The pre/post-processing parts of the EasyFEM software has been mostly developed using VB for the design of user interface objects, such as dialog boxes, menu icons, toolbar, graphics canvas, etc. In selected parts, VC++ is used only for graphics implementation and some add-on functions for VB via dynamic-link library (DLL). Because the software was intended for the analyses of disciplinary problems, the user interface design should maintain the consistency, familiarity, observability and responsiveness in order to enhance its usability [2]. To illustrate some implementation of these ideas on the EasyFEM software, Fig. 1(a)–(b) shows an example of the two dialog boxes used to apply nodal loads of the heat transfer and solid mechanics problems, respectively. These dialog boxes look similar in order to improve the learning curve of newcomers for analyzing problems in different disciplines. The same type of the dialog box, shared by all disciplinary problems, provides the proper load types for each specified problem automatically with consistent interface design. So that when a newcomer learn how to use one dialog box for a disciplinary problem, and he/she would grasp the idea for solving different problems in the other disciplines.

EasyFEM software is an integrated graphics/analysis environment that composes of two main parts, the pre/post-processors and the solvers. The general structure of the pre- and post-processors consists of several classes that
could be classified into four categories, which are the graphics, the finite element, the error handle, and the file I/O handle as shown in Figs. 2 and 3 by the Coad and Yourdon diagram [9]. The implemented data structure model follows the idea given in Ref. [10] with modifications for more convenient coding. The modifications include the use of VB to ease data preparation for both the element-based FE and the edge-based FV solvers.

The graphics category consists of several classes that handle all the graphics presentation. For instance, the base class is the abstraction for different geometry shapes such as the lines (Line class), arcs or circles (Arc class) and the splines (Spline class). The Point class stores all information about the points. The Boundary class stores information for the boundary formation by the given curves and also the discretized finite element model that consists of elements, nodes, materials, loads, constraints and solutions. The 2DViz class is the abstraction for several classes performing all the solution presentation by scientific visualization such as the fringe plots (Fringe class), contour lines

Fig. 1. Shared dialog box concept of nodal load applications: (a) for heat transfer problem and (b) for solid mechanics problem.

Fig. 2. General structure of EasyFEM software.
or streamlines (Contour class), deformation shapes (Deform
class), vector plots (Vector class) and particle tracing con-
tours (ParticleTracing class).

The finite element category consists of six main classes
containing all information of the finite element model. The
Element class stores the information of the three-node trian-
gular elements while the Node class stores all information of
the nodes. The mesh generation of EasyFEM consists of the
Delaunay mesh generation technique [11–13], the automatic
node creation technique [14], and the mesh quality improve-
ment by the mesh smoothing technique. The Constraint and
Load classes are abstractions for storing the information of
the constraints and loads, the specified initial conditions
and/or the boundary conditions, respectively. The Material
class stores the material properties assigned to all elements
in the finite element model. The Solution class stores the
numerical solutions that are returned from the solver.

The error control and file I/O manipulations are sepa-
ately designed as ErrorHandle and FileHandle classes.
The ErrorHandle class handles all error events including
inter-processes error which should be introduced by exter-
nal solver. The FileHandle class is an abstraction class for
handling input/output data stream which includes the
ModelFile class for model file management, the FEFile
class for fixed-format text file handling used to communi-
cate with external solver and the TempFile class for con-
trolling cached file which is used to speed up the input/
output data streaming.

The solvers for analyzing disciplinary problems are
developed independently by using VF due to its efficiency
for handling mathematical and matrix/vector operations with several object-oriented features supported by Fortran 90/95 [8]. The general structure of the solvers consists of several classes implemented in modules, while derived-data types are encapsulated to provide interface with external objects. The six solvers perform analyses for problems in the different disciplines by the FEM or the FVM such as the heat transfer, solid mechanics, thermal stress, potential flow, low-speed viscous flow, and high-speed compressible flow analyses, respectively.

The design concept of the EasyFEM software and its analysis procedure for solving disciplinary problems are shown in Fig. 4. The main features of the software are summarized as follows.

2.1. Pre-processor

- Create, modify and delete the model geometry consisting of point and curve objects with the help of dialog boxes. The software supports many types of curves such as lines, arcs, circles and splines and uses several methods to create both the point and curve objects.
- Change background color of the graphics window.
- Define multi-boundary domain by selecting curves to form a simple closed loop, with or without holes, to define the boundary object.
- Create, modify or delete material types (e.g., steel, water, wood, etc.), model properties (e.g., thickness of elements), and analysis types (e.g., plane stress or plane strain case).
- Discretize the selected boundary objects into three-node triangular elements.
- Apply loads and boundary conditions on nodes or edges of elements depending on types of problems.

2.2. Solver

- Perform linear stress analysis [15].
- Perform heat transfer analysis [16].
- Perform thermal stress analysis [15].
- Perform potential flow analysis [15].
- Perform low-speed viscous flow analysis [17].
- Perform high-speed compressible flow analysis [18–20].
- Create ASCII output files for post-processor.

2.3. Post-processor

- Read ASCII output files.
- Display model deformation shapes.
- Display fringe plots or contour lines.
- Display flow vectors or displacement vectors.
- Display streamlines.
- Generate AVI files or show animation of particle paths by a particle tracing method.
- Support multi-color or single-color visualization with shading up to 1024 levels.
- Report minimum or maximum values of solutions.

3. Examples

Four examples are used to demonstrate some functionalities and the capability of the EasyFEM software for analyzing different disciplinary problems. The first example is a rectangular plate with a circular hole at the center, subjected to uniform loading along the left and the right edges. Due to symmetry of the problem, only a quarter of the plate as shown in Fig. 5 is used for modeling. For a domain with irregular shape, the EasyFEM software allows users to split the model into multi-boundary domains or objects as shown in the figure. Each object consists of a closed loop of curves that may include holes if needed. Discretization into a finite element model can then be performed using either the structured or unstructured mesh. Fig. 6(a)–(b) shows the graphical objects consisting of the boundary objects and the finite element objects, respectively.
Fig. 7(a) shows a structured finite element mesh with the applied forces for all the nodes along the right edge, and the proper constraints along the problem symmetrical lines. After the problem is analyzed by the Solid solver, the von Mises stress solution can be displayed in the form of a fringe plot on the deformed geometry as shown in Fig. 7(b).

The second example is a heat transfer problem in a circular plate with holes. The processes for constructing the geometry and the discretization to create a finite element model are the same as described in the previous example. However, the boundary conditions for this example are specified as the temperature of 80 °C for all nodes along the small holes and 30 °C along the outer edge of the plate. Fig. 8 shows the fringe plot of the temperature solution obtained from the heat solver.

The third example is a low-speed viscous fluid flow in a square cavity at the Reynolds number of 400 [21]. Again, the preparation processes for creating the geometry and the finite element model are the same as the preceding examples. For the fluid analysis, the boundary conditions...
of the velocity components and the pressure are required at the nodes on the flow inlet, outlet, and along the wall boundary. Proper initial conditions are also needed for all the nodes in the entire flow field for the iterative analysis method. Fig. 9(a) shows a fringe plot of the streamline distribution. Details of the solutions can be visualized by using the zoom feature, such as that shown by the velocity vector plot highlighted in Fig. 9(b).

The last example is the high-speed Mach 3 flow past the NACA 0012 airfoil configuration. Fig. 10(a) shows an unstructured finite element mesh with refined elements near the airfoil surface. Fig. 10(b) shows a typical solution of the density distribution around the airfoil. These four examples highlight the benefits of using the object-oriented programming to develop single software for analyzing disciplinary engineering problems.

4. Conclusion

The object-oriented approach was applied for the development of the graphics interface EasyFEM software. A simple idea of classes and objects classification for front-end graphics interface software design and development for existing console-based solver is described. The software uses the finite element/finite volume methods for analyzing the solid mechanics, the heat transfer, and the fluid flows problems. The interactive graphics user interface environment was implemented for the ease of constructing geometries, performing analyses, and displaying solutions. The data structures for both the pre- and post-processors, including the solvers were described. Four examples were used to demonstrate some functionalities and to highlight the performance of the software for analyzing different disciplinary engineering problems.

Acknowledgements

The authors are pleased to acknowledge the Thailand Research Fund (TRF) for supporting this research work.

References