Design and Implementation of Object Oriented Large-scale Finite Element Visual System

Xiaohui Wang¹, a, Liang Chang¹, b
¹Aircraft Strength Research Institute of China, Xi’an Shanxi 710065, China

Corresponding author: a wxhd100@163.com; b changliang@163.com

Abstract. As to the large scale visualized demands of Engineering finite element calculation system, a highly open visual software named HAJIF_PrePost is designed and implemented. In the architecture design, the hierarchy structure design of framework is detailed based on the object-oriented design technologies. In the pre-processing, Fem model data was documented and variable memory was managed by Boost which is similarly C++ standard lib. And the method of node-face correlativity and Depth-Buffer mechanism of OpenGL were used to improve visibility efficiency for Fem model. In the post-processing, the contour drawing, deforming and animation generation is design and implemented, then add rear query function. The practical application shows that it has a practical performance, user experience, and extensibility.

1. Introduction
In finite element analysis, pre-process post-process as a result of the inspection and analysis of FEM, has become a widely adopted finite element visualization technology. In recent years, with scientific research and engineering technology in the field of continuous exploration and development, the finite element calculation scale is more and more big, the finite element mesh model often reach millions of elements or even tens of millions of nodes. It is a huge challenge for the finite element calculation and closely related to the visual system. In terms of visualization technology, it discussed the system framework of the finite element post-processing by adopting the UML[1]. Then, according to the characteristic of the millions of 3D finite element model, it has summarized the basic requirements and key algorithm of visual tasks[2]. Then, He has developed the finite element post-processing system by using MVC three-tier architecture based on Direct3D graphics programming interface[3]. It has developed the ElVis system for high order finite element accuracy and dynamic interaction[4]. And then, it has implemented the high order finite element fields rendering of the display based on the GPU technology[5]. In practical applications, most of the CAE software has the strong visual system, but due to limited function, poor extensibility and openness, unavailable development interface, larger calculation kernel, highly price, and the blockaded technology, such as software professional is not strong, large-scale data management problems is still existed.

In view of the growing problems of finite element calculation, the research and development of the visualization system is still needed. So the finite element visualization system (HAJIF-PrePost) was designed and developed. The following main about the framework design, data management and visualization technology are discussed.

2. Framework and data structure
The entire software used Visual Studio 2010 to set up a development environment, with finite element data structure as the core, to carry out data processing, data analysis and data display in three parts, so as to realize various functions and completed a feasible visualization system.

2.1. Framework model
This system should have a cross-platform graphics environment, according to object-oriented programming ideas, using the encapsulation, inheritance and polymorphism to separate and package the basic function of graphics processing [6]. In the overall architecture, it uses a three-tier architecture as shown in Figure 1. The bottom layer is the data layer, manages finite element model data and graphic data, and belongs to the software data foundation; the middle layer is the business layer, serial interface layer and data layer, completes the definition and implementation of specific functional interfaces, model reading and writing, data processing, etc., belongs to the software core; the top layer is the interface layer, complete the data display and user interaction functions, including parameter modeling and solution tasks management.

![Figure 1. System architecture](image)

In view of the openness of the framework, the whole software architecture is composed of a main program and a plurality of basic service components. The whole construction process follows the object-oriented and modular software construction technologies, and forms a series of basic service components through a dynamic link library. Different components through the definition of interface work together to complete the data transfer.

2.2 Data structure
Finite element analysis data has a complex structure and a variety of types. A large amount of data is involved in the analysis process [7], which mainly includes:

- The overall data, mainly describes the data of the finite element analysis, such as the number of units, the number of nodes, the number of types of materials;
- Node data, including node coordinates, node degrees of freedom, node forces, node displacement, etc.;
- Element data, including element type, element attribute data, element material data, etc.;
- Material data, including elastic modulus, Poisson's ratio, etc.;
- Load data, including physical strength, concentration force, surface force, dynamic load;
- Constraint data, including constraint nodes, constraint displacements;
- Result information, including element displacement, element force, stress, strain, and other data.

Based on the above features, according to the object-oriented programming method, the principle of hierarchical design is applied, the related classes are abstracted and defined. Among them, the basic data classes of node, element, constraint and load are at the core of the data structure, which
respectively realize the encapsulation of node data, element data, load data and constraint data, so as to generate derived classes and apply it to different types of finite element analysis, at the same time can be related to each other.

For example, the moment class and temperature class can be derived from the load class, and the stem class, the beam class, and the shell class can be derived from the property class. Here, the Element class is taken as an example. The class hierarchy structure is shown in Figure 2. The Property and Material classes implement the encapsulation of the unit attribute data and material data, respectively, and are at the bottom of the data structure. They directly serve the Element class and its derived classes, and are not associated with other classes. It can be seen that through the inheritance design of the class, a data structure framework adopting the object-oriented design concept is built. Finally, the resulting data is stored primarily with the aid of nodes and elements, which in combination with various finite element analysis results form an organic whole.

![Figure 2. Element class data hierarchy structure](image)

### 3. Pre-processing module

#### 3.1 Fem model

In practice, finite element model files, unit forms and calculation results are varied, requiring the module to have good scalability. Therefore, the HAJIF_PrePost software adopts the factory method design pattern [8]. It uses all the important features of object-oriented inheritance to filter and classify all categories of finite element pre- and post-processing, and establishes a complete class hierarchy of pre- and post-processing. This model defers the instantiation of a class into its subclasses by defining a product object interface and a factory object interface, allowing the system to introduce new products without modifying the factory interface [8]. An example of a factory method pattern is shown in Figure 3.

![Figure 3. Factory method design patterns](image)

Corresponding to the finite element model module, the main definition of the file import class, the element class and the result class 3 product object interface and file import description class, element
description class and result description class 3 factory interfaces. Element class used to save the unit, the node data and its structure related edge and patch information, specifically contains the model data in each base class. Due to the existence of multiple extension interfaces and factory interfaces throughout the module, a model management class is required to manage various basic classes. In order to facilitate the call of other components, the management class is stored in the system global object list for use by other components when the component is loaded.

3.2 Model data management
Model management is the basis for CAE software before and after processing. On the one hand, a unified database file is required to archive the model data in memory. On the other hand, it requires quick and convenient exchange of internal and external data. The quasi-C++ standard library Boost Serialization provides this important function in the form of a library for C++. It can serialize various types in C++, and can completely separate the archive format from the serialization of types. Any data type can be archived in any format and has great flexibility [9]. Use the Serialization library to convert the model object to a byte stream for storage or transmission. When necessary, restore the equivalent to the original state. Use the Boost library's smart pointer for memory management. Avoids memory leaks when Boost Serialization is used for serialization of model objects.

3.3 Model node visualization
The three-dimensional model visualization mainly includes the rapid formation and efficient blanking of the node graph, in essence, only the information of the outer surface of the calculation area is displayed.

Firstly, determine the inside and outside of the model. In the 3D model node, the inner element plane belongs to and belongs to only two elements, and the outer element plane belongs to only one element [10,11]. Therefore, taking the node correlation surface method can determine the inside and outside of the model, the steps are as follows:

1) Create a node-related list \( F(\text{node}) = F_1, F_2, \ldots \), where node is the node number and \( F_i \) represents a unit plane. For example, the tetrahedron with unit number 127 has 127.1 on one side and 127.2 on the side;

2) Comparing all related surfaces of non-empty \( F(\text{node}) \) list, excluding the nodes with the same composition, the rest is the outer surface related to the node.

Then, complete the model node drawing. In the process of drawing, we need to hide the internal elements of the 3D model and the blocked external elements through OpenGL's Deep Cache technology to complete the 3D model visualization.

4. Post processing module
Finite element analysis of the post-processing technology is to study the finite element simulation of the visualization of the problem, in the analysis of forming, visualization technology has displayed, check, adjust, improve the finite element analysis of the results of the function, therefore, in a large number of engineering applications An important role.

4.1 Result data
The main function of the post-processing system is to graphically display a large amount of result data generated by the finite element calculation. Therefore, establishing an effective data structure is the key to the success or failure of the post-processing system. According to system requirements and functional analysis, finite element post-processing system generally involves the node data and Element data. The result data describes the information, including result type, analysis type, et al.

4.2 Result data visualization
The visualization techniques in finite element post-processing analysis mainly include scalar field visualization and vector field visualization. Scalar field visualization technology mainly includes color
contour, contour map and slice, of which the most commonly used color cloud. The vector field visualization technology is more complex, in most cases it must be converted into the corresponding scalar field visualization.

4.2.1 Color contour. Finite element data, including displacement, element force, stress, strain and other information, can be based on color contour mapping, through the color changes reflect the physical data changes, quickly, directly and accurately represent the distribution of physical quantities in the model, location and extreme value Size and so on. Among them, one of the keys to drawing a color contour is to establish the correspondence between the magnitude and the color of the physics[12], as shown in Figure 4.

![Figure 4. Physical field values and color correspondence](image)

A common way to determine color in a Windows environment is to use RGB macros. Generally speaking, the results of a finite element result in a large number of dangerous parts in red, indicating smaller and relatively safe parts in blue. The brief operation flow is:

- Firstly, determine whether the drawn variable is a node variable, and if the unit variable is a conversion type;
- Secondly, determine a color value set according to a certain rule, that is, a chromatogram, and select a color according to the relative value of the variable;
- Finally, the model is colored, fused and anti-aliased through OpenGL drawing commands to complete the cloud drawing.

In addition, on the basis of ordinary cloud graph, Lagrange linear interpolation is applied to the environmental color and dispersion color of each unit node, and the continuous cloud image is realized by smooth transition. In addition, the internal unit of the model is eliminated to reduce the rendering amount of the model unit and improve the display efficiency. The OpenGL Display List Mode[13] is designed as a command cache to avoid using the drawing mode of the drawing pipeline in the traditional mode so as to reduce the resource consumption, improve the operating efficiency of the program.

4.2.2 Deform. The engineering structure under the load will produce the deformation in the direction of each degree of freedom. After the finite element analysis, the displacement of the discrete nodes will be reflected, and the displacement of the nodes will be shown in the form of deformation map, which is more intuitive, accurate and more effectively evaluate the structure of the force deformation or vibration. However, the displacement of a discrete structure is much smaller than the size of a structure, and an appropriate proportion of the displacement of the node needs to be amplified, and then the coordinates of the node are re-superimposed on the discrete structure unit to thereby more vividly represent the deformation of the entire structure. In the HAJIF_PrePost software, the maximum node displacement is enlarged to $\gamma$ (0.1 ~ 0.2) times of the structure size. The scale conversion factor for the 2D ( $Def_{Ratio2}$ ) and 3D ( $Def_{Ratio3}$ ) model deformation diagrams is determined by the formula (1).
Among them, $S_X$, $S_Y$, $S_Z$ respectively the biggest span structure model in every direction $X, Y, Z$ direction, $X_i$, $Y_i$, $Z_i$ respectively the displacement value of node in $X, Y, Z$ direction.

$$\begin{align*}
\text{Def Ratio}_2 &= \gamma \frac{\max(S_X, S_Y)}{\max(\sqrt{X_i^2 + Y_i^2})(i = 1, 2, \ldots, n)} \\
\text{Def Ratio}_3 &= \gamma \frac{\max(S_X, S_Y, S_Z)}{\max(\sqrt{X_i^2 + Y_i^2 + Z_i^2})(i = 1, 2, \ldots, n)}
\end{align*}$$

(1)

5. Application Examples

In order to verify the actual performance of HAJIF_PrePost software, different models were selected to verify the three aspects of software display scale, display effect and display type. Test machine selected Windows 7 system, memory 8G.

- Full subway model visualization
  HAJIF_PrePost is very useful and effective in the face of very large scale finite element models (1.0 million nodes, 1.0 million Elements). Due to the Boost library-based model data management approach, the finite element model reading and writing operations become faster and more efficient, and model import speed and PARTRAN quite, as shown in Figure 5. Very large-scale model of the data processing capabilities and graphics rendering efficiency has high requirements, here, HAJIF_PrePost software supports large model visualization, fully normal display, and rotation, scaling smooth operation.

![Figure 5](a) Full subway model visualization, (b) Full subway model color contour

- Full aircraft model visualization
  The example model of Full aircraft model consists of 687,562 nodes and 909,297 elements (as shown in Figure 6), which includes the calculation results of displacement, stress and strain. The color contour can be drawn by importing the memory efficiently and quickly.

![Figure 6](a) Full aircraft model visualization, (b) Full aircraft model color contour
6. Conclusion

On the basis of requirement analysis, by encapsulating the basic data class and deriving the advanced class method, an object-oriented system architecture is established and a unified database is designed. In the concrete implementation, the factory method design mode, component technology, Qt graphical user Interface library, OpenGL graphics library and C++ standard library Boost, to achieve a large-scale data processing capabilities, openness and extensibility finite element pre-and after the processing module HAJIF PrePost; through the application of proven software that has good practical It supports a variety of analysis types and result types. It gives users intuitive feeling and convenient operation, solves some engineering problems, and enhances the market competitiveness of domestic CAE software HAJIF. It has broad prospects for development.

Acknowledgments

Thanks to leaders and colleagues for their guidance to me as well as their contributions during the writing of this article. Finally, I sincerely thank my wife for understanding and supporting my work!

Reference

[1] Zheng Xianzhong, Wang Cheng. Efficient design and implementation of the functionality framework of a finite element post-processing system[J]. Computer Engineering & Science, 2005, 27(6):53-56.
[2] Zou Jing, Ji Hongguang. Fast visualization algorithm for huge 3D-finite element models[J]. Journal of Graphics, 2012, 33(2):13-19.
[3] Gan Hai, Dong Xianghui. The post-processing technology of finite element simulation based on Direct3D [J]. Computer Engineering & Science, 2011, 33(8):122-127.
[4] Nelson B, Kirby R M, Haimes R. ElVis: A system for the accurate and interactive visualization of high-order finite element solutions[J]. IEEE Transaction on visualization and Computer Graphics, 2012, 18(12):2325-2334.
[5] Nelson B, Kirby R M, Haimes R. GPU-based volume visualization from high-order finite element fields[J]. IEEE Transaction on visualization and Computer Graphics, 2014, 20(1):70-83.
[6] Wei Yongtao, Yu Jianhua, Chen Junkai. Object-oriented approach to the Finite Element Programming: Basic Data Classes[J]. Journal of Sichuan University (Engineering Science Edition), 2001, 33(2):17-21.
[7] Liu Yang, A Novel Post Processing Platform for CAE Result Analysis[J]. Electromechanical Integration Technology, 2012, (7):1-5.
[8] Gamma E, Helm R. Johnson R, et al. Design Patterns: Elements of Reusable object-Oriented software[M]. Boston Addison Wesley, 1994.
[9] Luo Jianfeng. Deciphering Boost C++ standard libraries[M], Beijing: Tsinghua University Press, 2012:373-424.
[10] Yu Weiping. Quick Visualization of Finite Element Analysis Results[J]. Journal of Computer Aided Design & Computer Graphics, 2003, 15(12):1561-1565.
[11] Lin Genghao, Ma Tianbao, Ning Jianguo. Large-scale node generation and display technology in 3D finite difference computation[J]. Computer Aided Engineering, 2012, 21(4):1-5.
[12] Jian Xuedong, Lu Ling. An improvement to z-buffer hidden surface remove algorithm[J]. Computer Application and Software, 2007, 24(9):149-150.
[13] Zhou Wei, Tian Hongqi. Visualization in Scientific Computation of FEM[J]. Journal of Engineering Graphics, 2010, (5):112-117.