DEVELOPMENT OF AN OBJECT-ORIENTED GRAPHICAL USER INTERFACE FOR THE STRUCTURAL ANALYSIS PROGRAM

Ming-Chieh Chuang¹, Bo-Zhou Lin², Fang-Wei Hsu³ and Keh-Chyuan Tsai⁴

ABSTRACT

Computer software for academic research (e.g. DRAIN2D) is often deficient in Graphical User Interface (GUI) and commercial software doesn’t open GUI’s framework to the user. The advantage of GUI is so evident and the requirements of GUI always exist. Thus, there exist sufficient reasons to research on the GUI’s framework. This article introduces that how to build an extensible and flexible GUI system for the structural analysis program. Changes of requirement are considered in this exploratory study and the authors proposed Pattern-Oriented Design by which GUI’s framework becomes true object oriented. Authors adopt the nonlinear structural analysis program PISA3D as the analysis engine and implement a GUI named GISA3D (Graphical Interface of Inelastic Structural Analysis for 3D Systems) for PISA3D using the proposed methodology. In addition to collaborations and consequences of design pattern, authors show the essential difference between VISA3D and GISA3D in this paper.

Keywords: PISA3D, GISA3D, VISA3D, GUI, Pattern-Oriented, Object-Oriented

INTRODUCTION

Greatly improved technology in the late twentieth century eliminated a host of barriers to the design of user interfaces, and the vast improvement unleashed a variety of new display and interaction techniques named the graphical user interface (GUI). When the GUI time is prevailing, more and more application software supply an easy-to-use graphical interface. The trend of GUI is hard to resist and it advantage is even more difficult to deny. Thus, structural analysis program must take its full advantage. In the past, structural analysis program for research purposes usually use text-based I/O. When users generate their model, they sometimes have to obey fixed format. After a structural analysis, the user also must analyze tremendous amount of data and make sense of them. However, some commercial software, such as ETABS and SAP series provide complete GUI in addition to powerful analysis ability, thus have gained wide acceptance in structural engineering practice. Since PISA3D (Lin and Tsai, 2003) is completely developed in the National Center for Research on Earthquake Engineering, and has several features which not available in the commercial software, it has gradually gained wide applications by a number of practitioners and researchers. Similarly, PISA3D’s User Interface is as deficient as other academic software, so objective of this study is not only to create all new GUI named GISA3D (Chuang and Tsai, 2005) for PISA3D but also to construct a well-designed framework in order to make the GUI easy-to-maintain.

EVOLUTION OF STRUCTURAL ANALYSIS PROGRAM USER INTERFACE

¹ Assistant Research Fellow, National Center for Research on Earthquake Engineering, Taipei, Taiwan, mcchuang@ncree.org.tw
² Assistant Research Fellow, National Center for Research on Earthquake Engineering, Taipei, Taiwan, bzlin@ncree.org.tw
³ Assistant Research Fellow, National Center for Research on Earthquake Engineering, Taipei, Taiwan, fwhsu@ncree.org.tw
⁴ Director, National Center for Research on Earthquake Engineering, Taipei, Taiwan, kctsai@ncree.org.tw
Free format input file of PISA3D
PISA3D is the Platform of Inelastic Structural Analysis of 3D system for short and it provides a user-friendly input format for commands with full free format (Lin and Tsai, 2003). Any command can be input in any order and the input of the total number of command sets is not required. Besides, in any single one command it is not necessary to enter data in fixed column with fixed data type. Undoubtedly, full free format input is the important innovation of PISA3D’s user interface.

Visual post-processor VISA3D
VISA3D (Hsu and Tsai, 2003) is the Visual Post-process of Inelastic Structural Analysis of 3D system for short, and it has also been designed and developed using object orientation. VISA3D is implemented using Visual C++, MFC (Microsoft Foundation Class) and OpenGL. Although VISA3D is a kind of GUI, however, it is a post-processor and only assists PISA3D’s user to verify model and examine analysis results.

Lack of PISA3D’s user interface
In general, it requires more time to check the accuracy of the input data when a PISA3D’s model is generated from the original text-based input mode. Although VISA3D provides functions for model and result verifications, it still can’t satisfy full users’ requirements, especially model generating. User interface has essentially two components which are input and output. Post-processor is only in charge of output, but a complete GUI should also has a pre-processor that is in charge of input.

MOTIVATION OF GUI DEVELOPMENT

Advantage of Graphics Environment
The success of a graphical system is attributed to a host of factors. In the view of structural analysis applications, the following has been considered by the authors as advantages of GUI.

Model Generation
The direct manipulation lets cursor action and motion occur in a physically obvious and intuitively natural way. Model’s geometry generation can be completed in the physical object based graphical interface by mouse clicking or dragging. Quick data input through property sheet can efficiently prevent the user from operating error. Ideally, model generation provides whole fool-proof function and guarantees that the user operates structural analysis program correctly.

Model Verification
The graphical presentation is easy to check the model geometry and assignment of mass, nodal load, etc. through the use of pointing and selecting mechanisms. The verification capability efficiently reduces the risk of human error and ensures that input data are all correct.

Results Verification
Without a GUI, users often dealt with numerical results using MS excel and could not work in a single platform. Graphs can be recognized faster than text and that’s exactly why VISA3D provides graphical representation for mode shape, deformed shape, plastic hinge locations and sizes, etc. Graphical representation is much more intuitive and a better way to grasp analysis results.

Graphical Environment shorten the time for structural simulation
A structural simulation can be separated into three stages. First is the model generation, second is the analysis computation, and the third is the result understanding. In the structural engineering research community, many scholars study the algorithm of structural analysis and try to reduce the computation effort. They intend to shorten the duration of structural simulation. Likewise, pre- or post-processor also can save much time in first and third stages, thus, a powerful GUI is very much desired.

THE OBJECT-ORIENTED PARADIGM OF SYSTEM DEVELOPMENT
Developing environment and the tool kit
The Borland C++ (Hollingworth et al, 2002) is adopted as the objected-oriented programming language in this work, and the developing environment is Borland C++ Builder 6.0. Borland’s Visual Component Library is used to construct windows program and authors apply OpenGL (Wright and Sweet, 1999) and MATFOR (AnCAD Company, 2005) to three dimensional computer graphics.

Independent development
Although PISA3D has a complete objected-oriented framework, the GUI’s framework should be independent of that associated with the PISA3D in order to avoid the coupling. A complete independent framework makes the GUI’s analysis engine much more changeable.

The design pattern
Designing an object-oriented software is often hard, and designing a reusable object-oriented software is even harder. Software designers should not only solve the problem at hand but the techniques adopted have to be also general enough to meet future requirements. Once requirements are changed, the programmers usually have to modify original code several times. Experienced designers usually can avoid redesign, or at least minimize it, but inexperienced ones might not. Actually, expert designers never solve every problem from first principles. Rather, they reuse excellent and elegant solutions that have worked for them in the past. The value of design experience is undoubted and the purpose of design patterns is to record expert’s experience in designing an object-oriented software. Gamma et al. (1998) systematically named and explained each design pattern (Gamma et al, 1998) recurring design in object-oriented systems. Designers can apply the design pattern to solve specific problems and make object-oriented designs more flexible and ultimately reusable without having to rediscover them. The true essence of pattern-oriented design means that the well-designed framework applies many design patterns appropriately.

Apply design patterns to construct a well-designed framework in the implementation
Considering that material models or analysis requirements of nonlinear structural analysis program are always changing rapidly, for easy maintenance, a well-designed object-oriented framework of the GUI is necessary. Because the GUI system code is huge and complex, the authors have applied many design patterns to solve problems and design reusable framework. Some of the design patterns usually are not applied only one time, so the authors enumerate the best example and describe its collaborations and consequences using UML (Alhir, 2003). UML is Unified Modeling Language for short. It defines a standard for graphical representation to show the relationships and interactions between the classes and objects.

Singleton  Some of the classes manage the database relative to the system should have only one instance. For example, class ElementDB is in charge of the container of elements, so it only has one instance and provides a global point of access to it. Singleton pattern is the solution, and it promises that class only has one instance. The implementation of class ElementDB is depicted in Fig. 1. It is implemented through static data member and static member function. The static and private variable of ElementDB* belongs to the scope of the class and has to be accessed externally via the scope operation ElementDB::getInstance(). Similarly, authors applied the same method to implement these database classes which manage nodes or materials.

```
  ElementDB
- instance : ElementDB*
+ getInstance() : ElementDB*
- ElementDB()
```

Figure 1 Class diagram of singleton application

Observer  Designers use the pattern to define a one-to-many dependency between objects so that when one object changes its state, all its dependents are notified and updated automatically. The relationship between the node and element is a one-to-many dependency. When removing certain
elements, system must guarantee that the corresponding nodes that other elements still share cannot be deleted simultaneously, and the Observer pattern is the best solution. In the design pattern terminology, the key objects in Observer pattern are subject and observer, and a subject may have any number of dependent observers. Node is a subject object and Element is an observer object in the node-element dependency mechanism. The example of node-element observer pattern is depicted in Fig. 2.

**Command** The pattern encapsulates a request as an object and supports undoable operations. The authors apply it to the common use cases such as redo and undo. The key to this pattern is an abstract command class, which declares an interface for executing operations, and concrete command subclasses specify the executing operations to invoke the request. Because the command object can be stored and passed around like other objects, designers can provide a singleton class CommandHistoryCenter for command history that is the command container, and then support the Undo and Redo mechanism. The example of application is depicted in Fig. 3.

**Memento** Without violating encapsulation, memento pattern is the solution to capture and externalize an object's internal state so that the object can be restored to this state later. For example, consider the use case as adding a beamcolumn. Previously, command and command history have introduced as the basic of Undo and Redo operations. Actually, designers must provide the memento class to store the variation of command execution and the variation should not be public accessible, thus memento pattern is the solution. There are the originator, the memento and the caretaker three participants in the...
memento pattern. In the pattern-oriented terminology, class ElementDB is ‘Originator’, class ElementDBMemento is ‘Memento’, and class AddBeamColumnCommand is ‘Caretaker’. Caretaker only sees a narrow interface to the Memento and it can only pass the memento to other objects. Originator can see a wide interface that lets it access all the data necessary to restore itself to its previous state. The example for adding a beamcolumn is depicted in Fig. 4.

![Figure 4 Class diagram of memento application](image)

**Strategy** The authors have proposed the pattern to define a family of making input file algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm of making input file vary independently from the GUI context and realizes the GUI to be compatible with various different analysis engines. The example of application is depicted in Fig. 5.

![Figure 5 Class diagram of strategy application](image)

**OVERVIEW OF GISA3D**

Through the way described in these preceding sections, authors complete a new GUI named GISA3D for PISA3D. GISA3D is the Graphical Interface of Inelastic Structural Analysis of 3D system for short. The basic interface of GISA3D is shown in Fig. 6 and users can build model via these 2D or 3D child forms shown in Fig. 7. Authors have enumerated certain key functions of GISA3D as given in the following. Some example applications of the implementation are shown.

![Figure 6 Main Form](image)  ![Figure 7 Child Forms](image)
Model Generation
1. Object based intuitive 2D/3D graphical model generation.
2. Interactive menu-driven model generation with simultaneous 3D display.
3. Easy to assign nodal restraint and element’s section or material.
4. Easy “snap-to” grid for laying out element.
5. Unlimited Undo and Redo.

Model Verification
1. Toggle display of nodes, elements, loads, boundary conditions, etc.
2. Click on a member to obtain all the information on its section, material, etc.
3. Translation or rotation for full 2D/3D viewing.
4. Pointing and selecting to check the assignment of nodes or elements.
5. Display groups (Fig. 8) or loads (Fig. 9).

![Figure 8 Display groups](image1)
![Figure 9 Display loads](image2)

Results Verification
1. Mode shape (Fig. 10).
2. Deformed shape (Fig. 11).
3. Plastic hinge distribution (Fig. 12).

![Figure 10 Mode Shape](image3)
![Figure 11 Deformed shape](image4)
![Figure 12 Plastic hinge distribution](image5)

Extending application of GISA3D
At the same time as development of GISA3D, NCREE executes some experiments. These experiments are more complicated than other common ones, so we convey and modify original GISA3D in order to provide the function of simultaneous 3D display in the NCREE’s PNSE (Platform for Networked Structural Experiments). For example, Fig. 13 shows the model of the 2 story BRB frame (Wang et al, 2005) and is subjected to bi-directional earthquake loads. Similarly, Fig. 14 shows the experiment for the model of the 2 story SPSW frame (Lin and Tsai, 2005), and Fig. 15 shows the model of bridge system (Yang et al, 2006) that subjected to bi-directional earthquake. In addition to customized modification about GISA3D, we also add some new functions to the monitoring system. For example, fig. 16 shows “Gauge” enhances the readability about run-time information of experiment. Obviously,
simultaneous display supplies more intuitive and easy way to represent the current experimental state. Undoubtedly, extreme applications are the best evidence to prove GISA3D is “reusable”.

Next generation of GISA3D
As we all know, GISA3D originally adopts OpenGL to expedite the graphical efficiency and ability. Because VISA3D also adopts OpenGL to implement, GISA3D almost refers to VISA3D’s kernel. Actually, the code about post-processor in VISA3D is so huge, but now MATFOR provides us a better solution. MATFOR, the product of ANCAD is specially enhanced to facilitate the visualization for the application of structural engineering. Through the collaborative project with ANCAD, we extract VISA3D’s achievement in the post-processor and make it becomes API (Application Program Interface) in the MATFOR. In addition to speeding up the programming (i.e. show structure, deformed shape and plastic hinge distribution), it is an easier way to implement these functions (i.e. mouse dragging, pointing and selecting in 3D space) in GISA3D via MATFOR, so we will replace OpenGL with MATFOR in the next GISA3D version and provide more friendly functions for PISA3D’s users in the future.
CONCLUSIONS

GISA3D provides a complete solution for pre/post-process requirements and enables PISA3D’s users to finish their work in an integrated platform. Especially, most of the changeable requirements have already been considered and solved by the design patterns. Pattern-oriented design allows GISA3D to possess a well-designed framework that is flexible and easy to maintain. Actual applications of simultaneous 3D display for experiments prove the flexibility of GISA3D’s framework. By cooperating with a software company ANCAD, GISA3D extracts the achievement of VISA3D and adopts advanced 3D graphic techniques capable of increasing graphics performance and efficiency. In the future, this research will continue developing more powerful and efficient functions of graphic user interfaces to satisfy PISA3D’s users. If it is necessary, GISA3D can be extended to support other analysis engine.

REFERENCES


