INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 2, Supp , Pages 169–176

EXPLORER, A VISUALIZATION SYSTEM FOR RESERVOIR SIMULATIONS

JIFENG YAO

Abstract. In this paper, we introduce Explorer, a visualization system for reservoir simulations. It is designed for large-scale data sets and many technologies have been used during its implementation, such as a 3-layer Client-Commware-Server (CCS) structure, Object-Oriented method, VTK based rendering and etc. Compared with current commercial softwares, Explorer has many features including more data formats support, many user-defined properties and full support for Chinese characters.

Key Words. visualization system, post-processing, reservoir simulation,

1. Introduction

A visualization system is essential to reservoir simulation applications, which makes it possible for both simulation and reservoir engineers to find out what is inside the outputs produced by computing programs. Explorer is such a visualization system for both sequential and parallel reservoir simulators.

With the rapid progress in computing technology, such as CPU speed, disk capacity, network bandwidth and also the software improvements, reservoir simulation systems nowadays can generate large amounts of data (on the order of several hundred gigabytes to terabytes), and it has brought great challenges to current visualization systems, including data accessing, transmitting, processing and displaying. Explorer has made a lot of effort to achieve high performance when handling large-scale data sets.

In the following parts, we first introduce the so-called Client-Commware-Server structure[1]. Compared with the traditional Client-Server structure which is widely used in scientific visualization systems, this 3-layer structure can decrease the interactions between the server for computing and the client for visualization and make the whole system more independent and flexible. Then we discuss how the Object-Oriented method is applied in Explorer. There are all together 4 kinds of objects in Explorer: GUI objects, project administrator objects, document objects and rendering objects. After that, several aspects of Explorer will be mentioned, including a dictionary-like keyword parser, time-varying data manipulation, VTK based rendering and Chinese character handling in OpenGL windows. The last part of this paper is the conclusions and related work in the future.

2. The Client-Commware-Server Architecture

Most visualization systems use the Client-Server architecture (see Figure 1). The computations run on the server and the visualization systems run on the client.

Received by the editors January 1, 2004 and, in revised form, March 22, 2004.
2000 *Mathematics Subject Classification.* 68N19, 68U05, 68U20.

They are connected by networks. (Sometimes the two parts may be located in the same machine.)

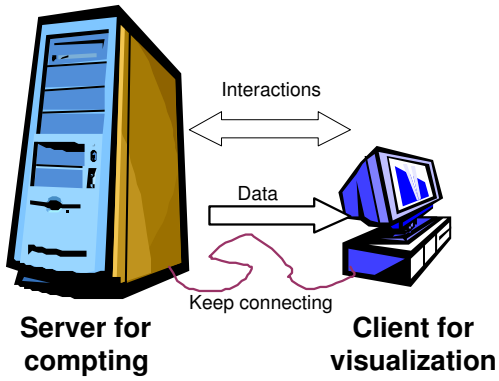

Figure 1 visualization system based on client-server structure

The C-S architecture has many good aspects. First, as we all know, numerical computation and the visualization have different demands of computer abilities. One needs powerful floating calculation ability while the other needs powerful graphics processing ability. Normally these two kinds of abilities are hard and no need to be provided by the same machine. By using the C-S structure, the numerical computation and visualization can be accomplished on different hardwares. Besides that, most applications need some kinds of interactions between the computation and the visualization, such as stopping, restarting or modifying the computation according to the visualization results, and the network between the server and the client provides a tunnel for this communication.

The problem of the traditional C-S structure rests with its high reliance between the two parts. It is hard to modify only a single part and not to affect the other one, because they are tied in an inflexible way. The main difficulty relies on the complexity of current networks and the operating systems and that is why we introduce the 3-layer Client-Commware-Server (C-C-S) architecture (see Figure 2).

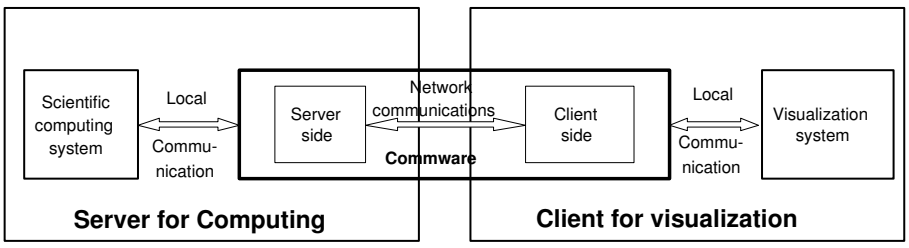

Figure 2 visualization system based on client-commware-server structure

Actually the name commware is borrowed from the well-known "middleware", and to some extend the commware is a kind of middleware which is in charge of the communications between the client and the server. Commware contains 2 parts: one part is located on the server for computing and the other is on the client for visualization. It hides all the details about the communications across different platforms and networks which both the server and the client side don't need to take into account.

Normally the reservior simulation programs are developed by experts of numerical computing and the visualization system comes from professionals familiar with computer graphics. It's always difficult to merge people from different fields together and the commware makes it possible that the two kinds of specialists only need to focus on their own concern and spend little on how to communicate with each other. The only thing they have to do is to follow the specific protocols on communication that we defined. Currently, the communication protocols defined in commware includes 3 parts:

- **Authorization**. Reservior simulations often run on supercomputers most of which have some kinds of user authorization mechanism, e.g. needing a password to access. This part handles the authorization related communications including encrypting user's private information, transmitting encrypted data over networks and set up the connection.
- **Data transmitting**. Besides the simple functions such as sending requests and accepting the computing results, some complicated functions, including break points resuming, real-time data transmitting, network fault tolerance, are also considered here.
- **Interactions**. At present several basic and essential functions are defined here, including starting, stopping or restarting simulations on the server.

3. The Object-Oriented method in Explorer

Object-Oriented (OO) programming has been widely available to developers for over 20 years, and nowadays software based on this concept is pretty ubiquitous[2]. Its major goals are to improve programmer productivity by increasing software extensibility and reusability and to control the complexity and cost of software maintenance. Explorer also adopts OO method during its design and implementation.

Explorer uses Microsoft Visual C++, one of the most popular OO languages, as the development environment. According to MS VC's famous document-view framework, there are four kinds of objects/classes in Explorer. They are

- **GUI objects**: They control the entire graphic user's interfaces (GUI); including answering messages sent by users and starting the corresponding operations. GUI objects are simply derived from MFC (Microsoft Foundation Classes).
- **Project administrator objects**: Explorer uses a project-case structure to manage the simulation results. Generally a project is set up when new data about a reservoir comes and one case stands for one simulation. Each project may contain lots of cases because users often run the simulation program many times in order to get the best result. All the project and case related data are handled by the project administrator objects. They are also in charge of recognizing different data formats and converting them into the Explorer specific data format.
- **Document objects**: These objects are also derived from MFC and they manage data for visualization and also the operations on the data side. Explorer has two kinds of data now, one is the 2-dimension form data, such as the production of wells and the other is the 3-dimension field data, such as the pressure or the saturation of oil over the whole region. Each kind of data is held by a corresponding document class and all the classes or the objects have a relational hierarchy. All the general operations for different data types, e.g. reading and writing data files, are arranged in

the base document class and their respective functions are in the derived classes. Together with the rendering objects, document objects are the key elements of the whole system.

- **Rendering/View objects**: Rendering objects, which combine MFC's view class and the VTK (the Visualization Toolkit) library, do the actual "drawing" of Explorer. Each rendering object has a corresponding document objects which supplies the data for visualization. All the pictures the users see on the screen are produced by rendering objects.

Figure 3 shows how the 4 objects above work together to form an integrated system.

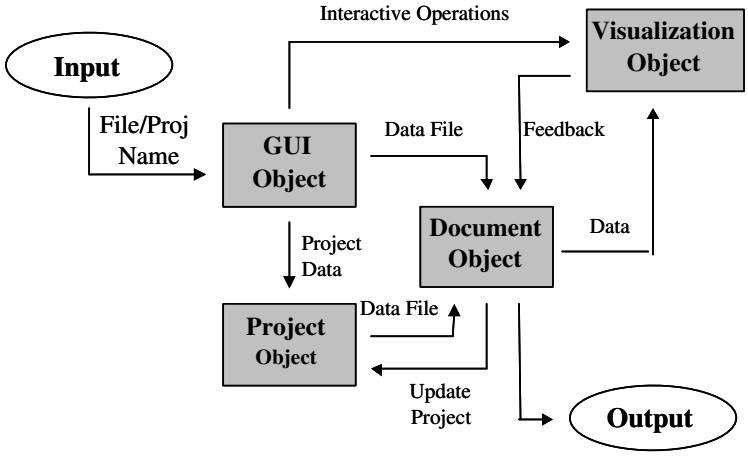

Figure 3 Objects in Explorer

4. Implementation Issues

Explorer is developed by Microsoft Visual C++ and runs on the Windows platforms. Currently its source code is more than 30,000 lines. Many techniques, such as multi-threading and movie generation, are adopted in Explorer either to improve its performance or to enhance its functionality. Several important implementation issues of Explorer are addressed in the following sections.

4.1. Dictionary-like Keyword Parser.

Most reservoir simulation programs use keywords to organize their outputs. For example, in Simbest II the keyword "RATEWP" in an output file indicates that the following array gives the production rate information of certain wells and the keyword "SW" is always put before a data array which stores the water saturation of the whole field. Different simulation programs have different keyword sets and Explorer have to recognize all the keywords it supports, the total amount of which is up to several hundreds. Furthermore, when the simulation programs add new keywords, Explorer must be capable to support them immediately. A dictionary-like keyword parser used in Explorer makes these possible.

A keyword dictionary is built in Explorer and it's easy to add, remove or modify words contained in it. Each word needs a registration process, which defines an action for this word. Normally the action stands for a function which will be called when system comes across the given word. When a new keyword comes, the only thing for the programmers is to write a function to handle this new word and add

both the word and the function to the dictionary by the well-defined registration process.

When a data file is ready, the keyword parser searches in the dictionary every keyword it comes across in the file and calls the pre-defined functions to handle this keyword. A Hash table is used here to improve the searching performance.

4.2. Time-varying Data Manipulation.

Reservoir simulation is a time-dependent process and it outputs time-varying data. A simulation always contains many time steps and during each step the simulation program writes the corresponding results to the output file. Normally there are several parameters in the simulation programs for users to decide what kinds of variables should be contained in the output file, e.g. pressure, oil saturation or gas saturation. All the selected variables or arrays will be sorted in the output file according to their time steps.

Nowadays the number of grid points used for reservoir simulation is up to several million and the size of a double array which stores one variable may be tens of megabytes. The data file size can be enormous if it contains dozens of variables and lots of time steps and it's always difficult to get the specific information from such a large file. We noticed that during the post-processing, mostly the users wanted to know how a single variable, for example, the production rate of an appointed well, changed when time went by. However in the output file different variables with the same time step are put together and this kind of data structure is not convenient for visualization. So in Explorer when a data file from the simulation system is ready, the first thing is to convert the data structure from the time steps based order to variables based order. The original data file is separated to dozens of well-organized small files. Basically these small files can be classified as 2-D plot data files (*.ppd) and 3-D field data files (*.pmd). Each well or region has its own plot data file, for example file well2.ppd stores all the variables of well no.2. Each filed data is also stored in a single file, for example, p.ppd stores the pressure values during different time steps. When users want to check how a variable changes during a simulation, Explorer simply opens the data file relevant to this variable and display information by time steps one after another.

4.3. VTK Based Rendering.

VTK (the Visualization Toolkit) is an open-source, object-oriented software system for computer graphics, visualization, and image processing[3][4]. VTK is based on OpenGL, the de facto industry standard for 3D graphics application development, but it hides the complicated OpenGL APIs and is easy to use when having learned about its basic object-oriented design and implementation methodology.

In Explorer, all the graphics related parts, including both the 2D graphics and 3D graphics, are accomplished by VTK. VTK is well combined with the MFC view classes and a set of hierachical VTK-view classes are used to handle different rendering requests.

VTK uses a graphics pipeline to transform graphical data into pictures and many objects are involved during the rendering process, among which the most important one is the vtkProp object. Props represent the things that we "see" in the scene on the screen and Explorer develops many specific props to show the reservoir related things. For example, the CPrisXYPlotActor class which is derived from the vtkProp class is used to generate x-y plots from one or more input data sets as shown in Figure 4. The most complicated VTK-view class in Explorer is the CPmdView class. It's used to represent the reservoir in 3D and it is combined

with many vtkProp derived actors for 3D reservoir structure, faults, wells, scalar bar, titles, and etc.

Many useful functions have been implemented in Explorer, such as showing information in specific layers or regions, selecting regions according their values ranges, saving outputs to pictures or movies. Figure 5-6 show some snapshots of Explorer's 3D outputs.

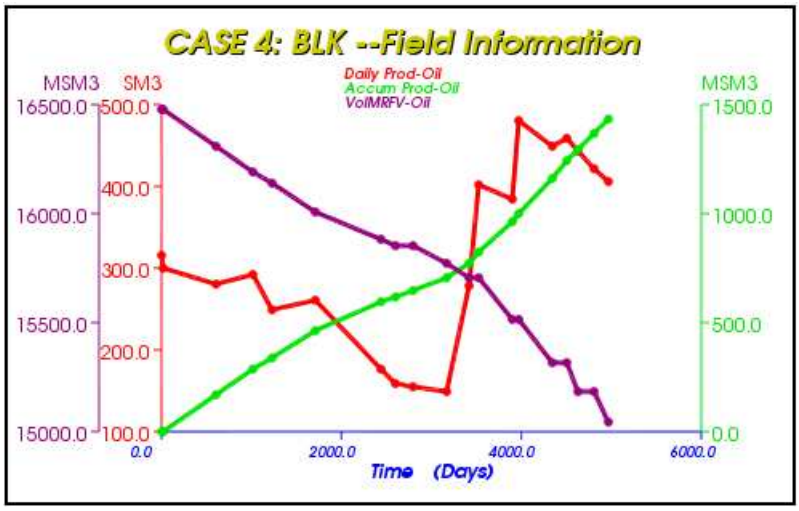

Figure 4 X-Y plot of Explorer

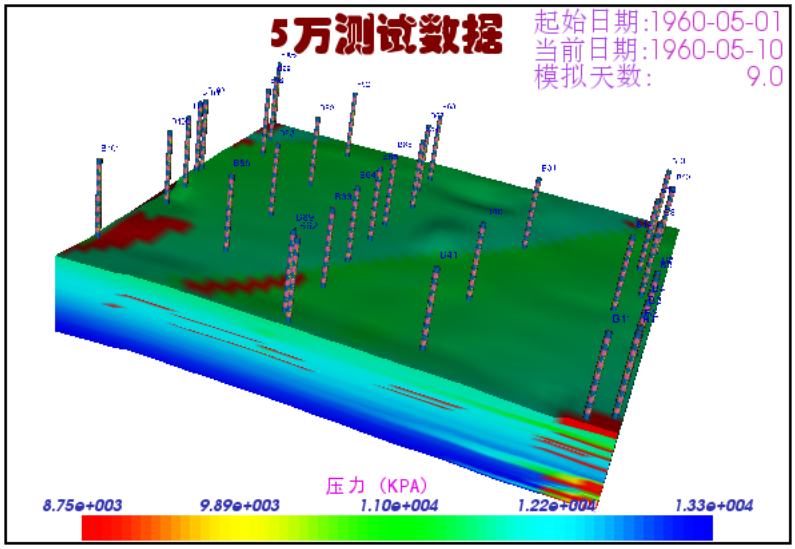

Figure 5 3D output sample (1)

4.4. Chinese Character Handling.

One of Explorer's features is its full support for Chinese characters. (Actually it now can support all the UNICODE characters such as Japanese and Korean.) This problem is addressed here because most visualization system uses OpenGL and OpenGL supports English characters only.

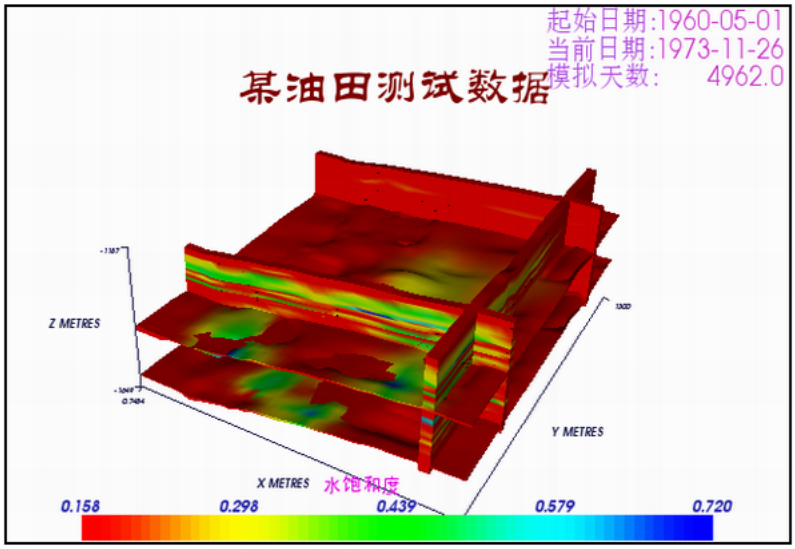

Figure 6 3D output sample (2)

Unlike the menus, the buttons or the texts displayed in a normal window, the OpenGL window is not controlled by the operating system. Simply putting non-English characters in an OpenGL window can only get some weird lines or points. Since Explorer uses VTK as its graphics library, we solve this problem by take an inside look of VTK.

An open-source library, FTGL, is used in VTK to display characters in OpenGL windows and FTGL uses another open-source library FreeType, which deals with the vector character files (*.ttf, *.ttc). They are combined together and make VTK have the ability to show vector characters. Operating systems of different language versions all have a set of vector character files and the problem is how these vector characters can be used in VTK based OpenGL windows. The answer relies on the translation between different character code sets. Two codes sets are involved here. One is the ASCII (American Standard Code for Information Interchange) supported by OpenGL and the other is UNICODE which is used for many languages including Chinese. If we can convert the Chinese characters for being displayed in VTK windows into UNICODE and give a vector character file, FTGL can successful draw the characters with FreeType library's help. Fortunately many ways are possible to accomplish such a translation and the function MultiByteToWideChar is a good choice for the Windows platform. In Figure 5-6, we can see many Chinese characters well displayed there.

5. Conclusions and Future Work

Explorer is a post-processing system for current reservoir simulation programs and it is intent on dealing with large-scale data sets. Many technologies have been used during its design and implementation, such as the Client-Commware-Server structure, the Object-Oriented method, VTK based rendering, and etc. Compared to existing commercial software, Explorer has its own distinctive features, for example full support for non-English characters. Explorer does not have as many capabilities as the commercial software, but it tries to provide the users the most practical functionalities in a very convenient way.

In the near future, we will do our endeavor to improve Explorer in two ways. On the one hand, we will try to add some new functionalities and features to Explorer, such as full remote control and more supported data formats. On the other hand, some new technologies for the next generation system, including GRID based visualization, data compression, parallel visualization and large-scale display, have been put into our schedule. Actually we've got some financial support and the related research has already started.

References

[1] J. Yao, Scientific Visualization System and HFFT over Non-Tensor Product Domains, Ph.D. thesis, Graduate School of the Chinese Academy of Sciences,2004.
[2] S. Khanal, Object Oriented Programming: An Introduction, Published in CORE, Jan/Feb, 1994.
[3] http://www.vtk.org
[4] W. Schroeder, K. Martin, B. Lorensen, The visualization ToolKit: An Object-Oriented Approach To 3D Graphics, 3rd Edition, Kitware, Inc. publishers, 2002.

Parallel Computing Lab, Institute of Software, Chinese Academy of Sciences, Beijing 100080, PRC
E-mail: jifeng.yao@gmail.com