Research

A Procedural Approach to the Fast Display and Analysis of Multi-Million Simulation Models in 3-D

Kamran Husain [§]*, Majid Shehry *

Address: Saudi Aramco Oil Company, Dhahran, Saudi Arabia

Email: Kamran Husain - kamran.husain@aramco.com : Majid Shehry - Majid.shehry@aramco.com
*These authors contributed equally to this work
[§]Corresponding author

Published: 30 August 2005 Received: 06 August 2005

E-Journal of Reservoir Engineering 2005, *ISSN: 1715-4677*. Accepted: 20 August 2005

This article is available from: http://www.petroleumjournals.com/

Abstract

The need to visualize huge geological models demands solutions with in-expensive display hardware since it is economically impractical to furnish a large number of high-end workstations to all engineers. The model files in these cases exceed the 2 gigabyte limit on most PC hardware and are harder to handle without the use of high-end workstations or software solutions. The use of intelligent disk caching, the object oriented interpretive programming language Python and Open Source software resulted in an efficient, fast rendering application. The compact code size for the reusable source libraries enabled an in-house solution while ensuring fewer support issues while providing necessary building blocks for leveraging more applications.

Introduction

Geological simulation models are getting larger and there is a need to display these models without the use of expensive hardware. The files in these cases exceed the 2 gigabyte limit on most PC hardware and are difficult to handle; requiring the use of high-end workstations. It is economically impractical to furnish a large number of high-end workstations to all engineers. Therefore resources have to be allocated on an as-needed basis.

In addition to the hardware cost, user specific analytical software requirements for displaying specific attributes within the large data sets take time to develop. Out-sourcing is not always an alternative when handling sensitive data and algorithms.

The procedures described in this paper solved both of these issues. These solutions can offer alternatives to conventional software development in areas of 3-D visualization and data mining in many industries, including the petroleum industry.

This paper discusses the issues in developing software for both pre-processing and post-processing multimillion cell models using the same base code libraries. The paper discusses the use of scripting languages for rapid prototyping of data-handling algorithms for retrieval of attributes from industry standard as well as company specific format files.

The paper further discusses the methods used to offer the solution on a cross-platform basis on both the UNIX® and Windows XP™ platforms. Design issues are addressed specifically to address issues with network bandwidth, byte order of data, the types of data formats, cross platform issues in the choice of displays and the development environment.

Hardware was limited to lower end computers (PCs); however, the solutions were scaled and tested on higher end special purpose workstations. The software worked remarkably well in all situations and was customized to dynamically use the best solution by automatically detecting the underlying hardware and operating system.

The problem

Geological models vary in size from several hundred megabytes for a small model to several gigabytes for a detailed study. Reservoir engineers typically work with simulating underground oil, water and gas activity with the use of simulators which require input from parts of the geological model; since the entire model has simply too much information for the simulator to work efficiently.

This partial input to simulators is called a simulation model, which in this case is derived from a model of the subterranean geology in a large geological model. A bulk of the geological model information must be scaled down to fit as input into a simulator.

The size of geological models for simulation is getting larger and as a direct result the size of simulation models is increasing as well. The use of geological simulation models poses the following problems:

- Displaying data from these geological models requires heavy weight applications and hardware.
- The files in these cases exceed the two gigabyte limit on most PC hardware and are harder to handle; requiring the use of high-end workstations or newer versions of software that are capable of handling these files. The legacy applications would not work without a vendor upgrade.
- Output data from the simulators is in big endian format whereas the reservoir engineers using the software did analysis on little endian machine. The elements in the input arrays had to be byte-swapped for reading into reservoir simulation 3-D (RS3D) software.
- The input format into POWERS™ was based on a proprietary format developed in Saudi Aramco. Although other software vendors had software capable of displaying models in the moderate size (4-10 million cell), these programs require modification to be able to read the POWERS™ Binary Format (PBF). These modifications were not a high priority for the software vendors since it is a solution for only one client, Saudi Aramco.
- At Saudi Aramco, in addition to POWERS™, we use other types of simulators. Each simulator vendor has their own output variation and more importantly, different applications are required to show each type of data. One application does not handle both the input and output data formats from all vendors.

There was an additional issue to consider; it is economically impractical to furnish a large number of high-end workstations to all engineers. Consequently, resources have to be allocated on an as-needed basis. This implied a need to display these geological simulation models without the use of expensive display hardware.

The Solution

The development of the RS3D toolkit addressed the problems efficiently. The resulting code addressed the problems and offered the following benefits:

- The toolkit handles both input and output data visualization in one application for both pre-processing and post-processing data formats, thus requiring the end user to learn only one tool.
- The toolkit runs on desktop PCs running Linux or Windows XP™ as well as remotely off the Linux Clusters.
- The toolkit handles heterogeneous data beyond the 2G limit on desktop machines.
- The code libraries are owned by Saudi Aramco and therefore we own the rights to the algorithms. In addition, outside vendors are not involved in the development of the code.
- The code was developed in-house in a fast, efficient manner and since we own the code, we can modify it to meet our specific requirements.

The development environment. The programs were developed using Open Source software. Two significant packages from the Open Source repository were:

- The Python programming language.
- The Visualization Tool Kit (VTK) from Kitware systems, Inc.

Python is an interpreted language and was used as the main tool in developing the user interface and internal data parsing libraries. The Python language is object oriented, has a very fast virtual machine and can dynamically link in modules to adapt to different data input. It has the ability to link in C++ libraries as extensions where speed is critical. In most cases, the

rich set of functions in the language itself sufficed for our requirements.

Python code runs on Windows XP™, Linux and Solaris workstations without modification. The internal interpreter has to be built once for each platform to take advantage of the underlying operating system. Once installed, all subsequent Python programs are run by this internal interpreter.

The VTK leverages existing OpenGL™ libraries to display 3-D graphics. VTK uses hardware support if available otherwise it relies on software libraries to render graphics objects. The VTK libraries provided highly useable graphic primitives and objects that are directly created and accessed from Python objects via an interface layer. See **Fig. 1**.

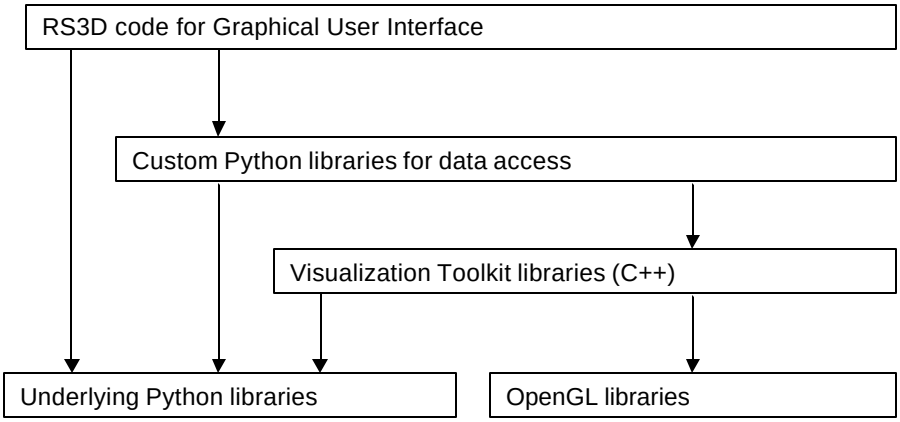

Fig. 1 - Interface layer for Python objects to access VTK objects and the OpenGL library

The major risk with using Open Source software was the apparent danger of not getting support should we run into a problem with the development libraries. Although Saudi Aramco has a contract with vendors that provide a supported commercial version of Python, there was never really a need to use the resource. No severe bugs were experienced with the toolkits. This trouble-free development can be attributed to the following reasons:

1. The quality of programming in the language and the toolkit itself were both excellent.

2. The vast library of useable components on the web.

3. Bugs were few and were fixed almost on a nightly basis when found by the large user community behind the VTK and Python tool kits.

4. An enthusiastic response from the Open Source community to questions posted by users on VTK mailing lists.

Handling Multiple Data Formats. The variety of input and output data formats still presents the biggest challenge in developing RS3D. The viewer has to dynamically check the input format to determine the type of data being rendered. The types of 3-D data encountered are:

- UNRST – ECLIPSE Unified Restart Files
- INIT – ECLIPSE Initialization Files
- CHEARS – Chevron-Texaco Simulator Input format
- GOCAD – Surface geometry
- GLK – Saudi Aramco specific format
- GEOVIEW – Saudi Aramco specific format
- PBF – POWERS ™ Binary Format
- ZMAP™

The problem was solved using Python objects for reading (and writing) each type of data format. Python is very modular and careful programming enables one to include all the source code for handling an object in one file. This file is called a module. In addition to the source code, the objects test routines and interfaces can be included in the module itself. Thus, the modules developed and all

the test routines for each type of input file exist in their own modules.

Application building became easier with the use of modules. Handling a new type of input now simply entails the development of one module to handle the semantics of that new format. The underlying data structures in RS3D allow the easy integration of a new module with only a few lines of code. An example of this is the way CHEARS input format was added to RS3D in less than one day. The only changes in the RS3D main screens were cosmetic changes and the bulk of the changes and additions were made in a module specifically designed for CHEARS input files.

Another primary motivation for developing RS3D™ was the limitations of tools that show data in proprietary input formats for the Saudi Aramco POWERS™ simulator. As the size of the models increased by tens of millions of cells, the existing legacy tools were unable to display the data without subdividing the files.

The first action was to write the input and output modules for reading and writing POWERS™ Binary Format (PBF) files. An additional twist in this was the ability to handle the big and little byte order (endian) of the data. Python allows the use of an internal byte-swapping routine to permit reading data to accommodate different processor architectures.

The RS3D toolkit tackles the problem of displaying the data from these huge models with intelligent disk caching and slicing algorithms. The output from reservoir simulation simulators comes in a verbose, run length encoded format. This format does not permit direct access to key portions of data. The RS3D toolkit uses Python dictionaries and file offsets to index into the data areas for faster access. The built-in capability of the Python programming language to handle slices was a key factor in extracting data in various levels of detail (LOD).

Speed considerations. The need for speed was the main driving force behind encoding test routines within each module. The main algorithms cached locations on the disk for each sector and layer. Thus, loading layers in the model simply amounted to a series of seek-and-read operations per layer.

The way to handle layers depended on the input model being read. Some models hold the origin on the lower left corner of the model, whereas some models define the origin on the top right corner.

The way depth and cell dimensions are defined varies on the type of model. While most geological formats stored data in FORTRAN order, the GLK format defines layers in a transposed manner with respect to the other types of models and the layers in a bottom up fashion. Also, some models are stored in text form, some models store data in pure binary form while others store indices in text and the attribute, raw data is stored in binary form.

For all such cases, the Python language permitted switching handles to parsers with use of dictionaries. Each entry of the dictionary was populated at the time of reading a file with the type of file being read. The rendering program sees only the handlers and is not involved in the details of how the data is brought in by the handler. Essentially, this framework involves only the addition of a new handler for each new type of data we encounter in the future.

Handling large output files. Output files from the simulators are written in individual variations of ECLIPSE format by three simulators in use at Saudi Aramco. The ECLIPSE format uses variable length records with a large overhead in the way large blocks are segregated. This segregation of blocks renders quick offset jumps (within a record) useless and forces a slow, sequential search operation. Prior to RS3D, the important information from a simulator run was mined by running a special extraction program and a whole new set of extracted data. This procedure had the following problems:

- The data was duplicated in another set, creating a data management headache
- Versions of ECLIPSE simulator output forced the development of different versions of the extraction software.
- Not all the information was extracted from the original output
- Some of the extracted information was further processed through algorithms in the extraction utility. This process may introduce unknown elements in the data.

RS3D solves the above problems with the use of dynamic modules in Python which load the correct module in real time based on the type of data being read. Thus all known variations of the ECLIPSE data are handled in one version of the program.

The internal libraries handle the byte order for the data and indexing unbeknownst to the user. Changes are made to one source tree for all platforms.

Using intelligent indexing, RS3D obviates the need for a separate extraction run. The data viewed by the user is taken from the actual simulator output. An extra data set is not required thereby reducing

the headache of managing another data set and reduces the probability of errors.

Well paths are displayed if present in the geological model. The user can include text files with cell paths for displaying well trajectories for speculative work.

An equation editor and parser are also included in the toolkit to allow the user to make their own attributes based on their own functions. This function is currently being enhanced and optimized to allow for more complex mathematical operations.

Additional Benefits. Displaying GOCAD data in RS3D does not require the use of a GOCAD license. A small number of concurrent licenses for GOCAD are currently shared among several users who may only want to view files rather than edit them. Viewing a file in GOCAD uses a license as if the user were editing a file. RS3D permits a user to view the files without using a valuable GOCAD license.

Another benefit from using Python modules was the reusable library of input and output routines that were available to the rest of the members in our department. Other team members successfully used the libraries for building streamline applications using the base libraries from RS3D. The library source code is self documenting and generates HTML documents with the use of a simple utility that is a part of the Python distribution.

Other applications based on the RS3D code libraries included those for extracting simulated logs from simulation models for comparison with actual field log data to perform quality checks as well as the viewing of summary data from simulator runs. The XY plotting libraries allowed a user to view data from MS Excel™, log data and even adding and viewing of user-specified speculative well-paths with ease.

Conclusions

The solutions presented here confirm that it is possible to economically display huge data models on common machines using rapid prototyping techniques for specialized software. The resulting application set is fast, easy to maintain and extend. The internal libraries are useable in other project development teams and have proven to be a solid base for future development.

Acknowledgements

Many thanks are due to the following users who provided criticism and comments in making this application a success:

- Mohamed Diamond, RSD.
- Chung Lin, RSD
- Raja Tariq Abbas, RSD
- Bevan Yuen, RSD

References

1. Lutz, Mark. **Programming Python.** 2nd ed. Beijing: O'Reilly, 2001.
2. Grayson, John E. **Python and Tkinter programming.** Greenwich, CT: Manning, 2000.
3. Schroeder, Will, Ken Martin, Bill Lorensen, with special contributors Lisa Sobierajski Avila, Rick Avila, C. Charles Law. **The Visualization Toolkit.** 2nd ed. Upper Saddle River, NJ: Prentice Hall PTR, 1998.