Research on the Customized Color Mapping Technology of Reservoir 3D Display

Zhang Yan[1, a], Li Chao[2,b], Nie Yongdan[1c], Hou junlong[1,d], He Wenjie[1,e]

[1] School of Computer and Information Technology, North East Petroleum University, Daqing,China

[2] School of Foreign Language, North East Petroleum University, Daqing,China

[a]zhangyuanyan_309@126.com

Keywords: Open Inventor, DialogViz, MeshViz, color mapping, reservoir.

Abstract. The technology of color mapping could provide the visual representation of data mapping corresponding of the color range, in the three-dimensional visual scene. In order to analyze the characteristics of three-dimensional reservoir model more efficiently, this article researched the key technology of customized color mapping based on the Open Inventor. The MeshViz and DialogViz extended module of the Open Inventor were used to achieve the interactive legend, so the user could adjust the color mapping range dynamically, and change the data mapping mode by GUI interface. The different geological characteristics were displayed with the customized color, which would lead to a more intuitive understanding of the reservoir model.

Introduction

Although the currently popular reservoir numerical simulation software such as Eclipse could perform relatively good 3D reservoir visualization, the disadvantage is that they could not provide a customized display style according to the different user extra needs. So studying the fine display method of reservoir numerical simulation data model in depth, could not load the extra geological objects and data, but also provide a more intuitive understanding of the geological conditions of the target block and reserves information, thereby provide better decision support for oil exploration and development [1].

Reservoir 3D Display

Currently the OpenGL and Open Inventor were the most widely used three-dimensional graphics package. The feature of OpenGL was too powerful and complex to master for learners in a short period of time. Development based on the OpenGL usually needed too many code to make the tedious settings in three dimensional scenes, such as graphic projection, color, texture, light, and so on. While the Open Inventor was the most widely used object oriented graphic and images software development interface, it allowed the user to build a complex three-dimensional scene through the building blocks like approach, so that the users only needed to spend a little period of time to construct a complex three-dimensional scene [2]. According to the business needs of different fields, the Open Inventor included a number of extended modules. The MeshViz XLM extended module provided powerful functions of model grids drawing in the field of applied science, model grids extraction and data matching, and contained advanced data visualization design components, complex surfaces and 3D charts components, and other visual objects. The DialogViz extended module provided the programming interface that could interact with the 3D scene to control and manipulate the objects in the scene, and the usage of DialogViz node was very like the other Open Inventor nodes.

The pillar model was used in the Eclipse and Petrel software, the structure was shown in Figure

Foundation item: Scientific Research Fund of Heilongjiang Provincial Education Department (No.12521050), National Science and Technology Major Project of the Ministry of Science and Technology of China (No.2011ZX05021),CNPC Innovation Foundation (No 2012D-5006-0609)

1.The pillar model was organized by the pattern of dot and line, and the coordinates indexed by i and j are in the same edges of all grids with a straight line, such as the "100-101" edge in the 100 cell, and the "100-101" edge 101 cell were in the same single line [3].According to the z-coordinate of top and bottom points in the particular orientation edge of the target grid, the three-dimensional coordinates of the points could be easily calculated. So all the grid coordinates could be solved in each cell, and the index of attribute value could be obtained in accordance with the order of i, j, k.

With the Open Inventor 3D visualization technology, users could quickly and easily generate 3D stratum model display, and provide an intuitive analysis of the strata and reserves information for geological researchers and decision-makers, combined with a variety of interactive and customized tools, the three-dimensional scene structure was shown in Figure 2.

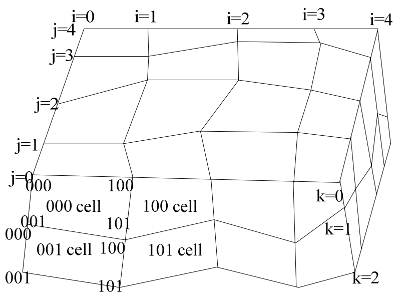
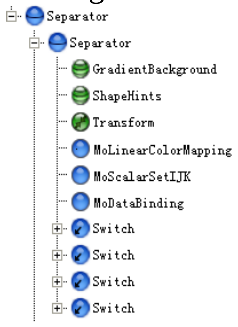


Fig. 1 pillar mode structure


```
Separator
  Separator
    GradientBackground
    ShapeHints
    Transform
    MoLinearColorMapping
    MoScalarSetIJK
    MoDataBinding
    Switch
    Switch
    Switch
    Switch
```
Fig. 2 the 3D scene structure

Legend Creation in MeshViz

Molegend node

The legend node in MeshViz was the Molegend, and the MoLegend node displayed a colored legend of the current colormap in a rectangle. A title could be assigned to the legend, the min and max values could be set, and their value could also be displayed. The legend might be horizontal or vertical. The colormap node it represented must be stored in the scene graph before the legend node.

Color mapping node

Color mapping meant computing a color from a scalar value. This color could then be applied on top of a surface to visualize a scalar set. The color mapping nodes provided an easy way to define a color look up table. Four types of color mapping nodes were provided. The MoLevelColorMapping node defined a constant color mapping defined by a set of N intervals between scalar values and N-1 colors representing the constant color used for values located in each interval. Thus, for a given value v, Vk <= v <= Vk+1, the associated color was Ck. The MoLinearColorMapping node defined a linear color mapping defined by a set of data values and their corresponding colors. For a given value v, Vk <= v <= Vk+1, the associated color c was determined by a linear interpolation between (Vk,Ck) and (Vk+1,Ck+1). Where Vk,Ck was the pair number k of the colormap. The MoPredefinedColorMapping node set a predefined color map. The range of the scalar field must be given to adjust the color map to the scalar set. Several predefined color maps are provided. The MoCustomColorMapping node set a custom color map. A custom colormap was a class implementing the MiColorMapping interface. This interface must implement a getColor() method which returned a SbColorRGBA from a given scalar value. A custom color map was allowed to create any type of transfer function.

Isovalues node

The Isovalues node specified which values were displayed. The MeshViz provided the PbIsovaluesList and PoIsovaluesList classes. There were two ways to define a list of isovalues. The first one was to instantiate a class derived from PoIsovaluesList and add this object in the scene graph, and the other one was to instantiate a class derived from PbIsovaluesList and call setIsovaluesList() to connect this object to a visualization node.

Legend initiation

In order to provide the convenience for the other modules calling, the legend module was used the modular program design thinking [4], and all functions in the interactive legend were packed in a class called mylegend. After the legend was initialized by the default constructor execution, the object legend was saved and indicated by the SoSeparator node named legendroot.

The legend was initiated by the following steps:

1. Create the legend node by specifying its geometry and attributes (size of the legend box, number of columns, title, value positions, etc.)
2. Associate a data mapping by inserting a PoDataMapping object in the scene graph.
3. Associate a list of isovalues by inserting a PoIsovaluesList object in the scene graph.
4. Specify which values to be displayed by setting the firstValue, lastValue, and periodValue fields of the legend node and choose to display values from the data mapping or the isovalue list node using the same method as for graduations along a linear axis, finally choose to display only the bounding values.
5. Add the legend to the scene graph.

Legend Interaction

Controls definition

The DialogViz module included many nodes such as SoTopLevelDialog, SoMenuBar, SoMenuPopup, and some nodes derived from the SoDialogGroup, like SoColumnDialog SoRowDialog, and SoTabDialog. DialogViz initialization supported in two ways, it not only could directly initiate from the program by using the code, but also could be read from the already definition files.

User Interaction

Unlike Open Inventor, which used callback functions, DialogViz used an "auditor" mechanism to catch interaction with dialog components. An "auditor" was a class contained a set of methods corresponding to the basic components (button, slider, etc). DialogViz provided a set of predefined "auditor" classes [5].

The key codes were shown:

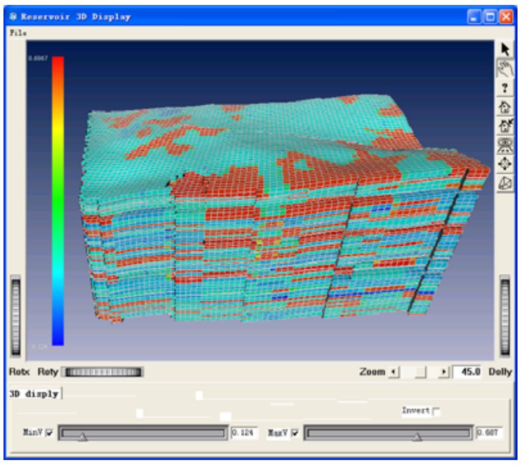
```
class AuditorcolorCheckSwitch : public SoDialogCheckBoxAuditor
Override the virtual function:
    void dialogCheckBox(SoDialogCheckBox* checkBox)
(3)Associate the controls and the Auditor class:
SoDialogCheckBox*colorCheck=(SoDialogCheckBox)
m_topLevelDialog->searchForAuditorId("colorcheck");
  if (colorCheck != NULL) {
       AuditorcolorCheckSwitch*auditorcolorCheckSwitch=new AuditorcolorCheckSwitch(this);
    colorCheck->addAuditor(auditorcolorCheckSwitch);
    auditorcolorCheckSwitch->dialogCheckBox(colorCheck);
}
```

Data mapping dynamic implementation

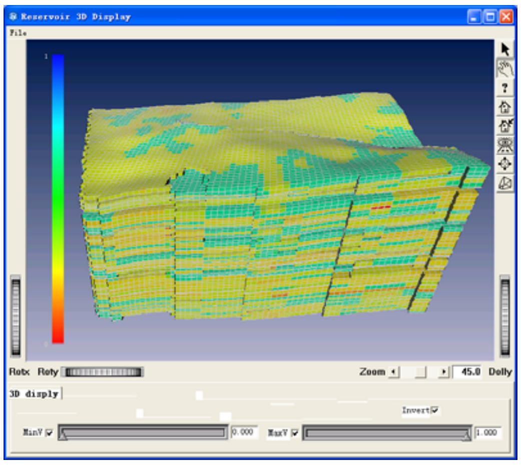
The data mapping method of PredefColorMapping was used to map the value of each attribute to the color in a predefined color range, and it was simple to implement. But the disadvantage of this method was that the distribution of colors and values in some cases were not flexible, and the user could not modify the color distribution conveniently. In order to customize the color mapping mode, and set the color distribution according to the properties of the reservoir model complying with user's habits, the MoLinearColorMapping mapping method was taken to achieve a dynamic legend which the data mapping range and mapping mode could be changed by the user with the interactive interface. The dynamic legend methods were implemented by the class mylegend that provided some public functions, such as the function setmapvalue received the data range parameter min, max, and the data mapping parameters order, so the external modules could change the status of legend by calling the setmapvalue method. The codes for the function were described as follows:

```
Void setmapvalue(float min,float max,int order)
  {
Definition of the length of the 5 numeric vector float values, and the color vector SbVec4f colors [5].
Set values [0:4] value between min to max;
  if(order==1)
     { Setting the color interval value as the normal sequence of the vector colors [0:4];
     }else
     { Setting the color interval value as the reversed sequence of the vector colors [0:4];
     }
The minimum value was set min;
The maximum value was set max;
}
```

The program run screenshots of dynamic legend were shown in Figure 3 and Figure 4, The Figure 3 showed the display effect of data range of 0.124 to 0.687 of water saturation attribute parameter of a certain reservoir model mapped from blue to red, and the Figure 4 showed the legend flip, and the model data range from 0 to 1 mapped from red to blue display effect.



Zoom 45.0
Invert ☐
MinV ☑ 0.124
MaxV ☑ 0.687



Zoom 45.0
Invert ☑
MinV ☑ 0.000
MaxV ☑ 1.000

Fig. 3 normal sequence mapping

Fig. 4 reverse sequence mapping

Summary

The customized color mapping technology of Open Inventor was used to display the three-dimensional reservoir model in this paper. The proposed method was simple, user-friendly and provided an intuitive understanding of the structure of the reservoir model, and the analysis of reservoir model characteristics. As the modular design concept was employed, so it had the strong scalability, and could be easily integrated with other business modules.

References

[1] TENG Yi-jian,CHAI Shan,JING Xu,JIAO Xue-jian.Virtual driving simulation system based on Open Inventor[J].journal of Computer Applications,2009,29(Z1):323-325.
[2] YAN Feng-xin,HOU Zeng-xuan,ZHANG Ding-hua, etc. Open Inventor programming from entry to the master [M]. Beijing: Tsinghua University Press, 2007.
[3]. Open Inventor6 User's Guide. Mercury Computer System Inc.2006.
[4]. Alan Ezust,Paul Ezust.An Introduction to Design Patterns in C++ with Qt4[M]. Prentice Hall,2006.
[5]. Josie Wernecke.The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor,Release 2 [M],Addison-Wesley Publishing Company,1994.