Uma Arquitetura para Construção de Ferramentas de Manipulação para Visualização Interativa de Dados Volumétricos

LÚCIA TERESA SCHALCHER DA FONSECA [1]
LUCIANO PEREIRA DOS REIS [1]
LUIZ FERNANDO MARTHA [2]

[1] PETROBRAS-Petróleo Brasileiro S.A.
Av. República do Chile, 65 s.1402F
20035-900 Rio de Janeiro, RJ, Brasil
{lucia,luciano}@ep.petrobras.com.br

[2] PUC - Pontifícia Universidade Católica do Rio de Janeiro
Departamento de Engenharia Civil
TeCGraf - Grupo de Tecnologia em Computação Gráfica
Rua Marquês de São Vicente 225, Gávea
22453-900 Rio de Janeiro, RJ, Brasil
lfm@tecgraf.puc-rio.br

Abstract. This paper presents an object-oriented architecture for the construction of 3D widgets, or manipulators, and its use to construct a set of tools for interactive visualization of volumetric data in oil-exploration-and-production applications The main ideas are the use of an event model that encapsulates the processing of events inside the manipulators and the use of objects with simple behaviour to compose their geometry and complex behaviour. The implementation is based on a graphics library that allows immediate-mode rendering (OpenGL), such that objects containing high volumes of data can be handled efficiently.

Keywords: Scientific Visualization, Interactive 3D Graphics, Direct Manipulation, Object-Oriented Design.

1 Introdução

A utilização da visualização tridimensional na área de exploração e produção (E&P) da indústria de petróleo representou uma grande melhoria no trabalho dos geocientistas. Essa tecnologia possibilita criar imagens 3D precisas de formações da subsuperfície e interpretar os dados de exploração e produção com maior acurácia e rapidez.

Ambientes de visualização modulares (MVEs), como o AVS [Upson et al. (1989)], tornaram-se ferramentas populares para visualização e análise de dados científicos mas o paradigma de *data-flow* em que se baseiam limita a interatividade e a manipulação direta dos dados. *Sliders* e entradas textuais são exemplos de ferramentas de manipulação utilizadas.

Widgets 3D são definidos como objetos que encapsulam geometria 3D e comportamento, utilizados para controlar ou exibir informações sobre objetos de aplicações 3D [Conner et al. (1992)]. As partes que compõem um *widget* 3D reagem aos eventos de interação e as restrições e relacionamentos à elas aplicados definem como o *widget* 3D influencia o objeto controlado.

A implementação de técnicas de interação 3D por manipulação direta da cena, especialmente utilizando-se dispositivos 2D como o *mouse*, não é uma tarefa simples. O mapeamento de movimentos 2D da mão para o movimento do objeto pode ser difícil e se complica pela forma indireta como é feito: da mão do usuário para o *mouse*, depois para o *widget* e, finalmente, para o objeto. Dispositivos com mais graus de liberdade vão aliviar esses problemas mas, no estágio atual, ainda apresentam problemas no uso (fadiga visual e muscular) e na resolução espacial [Herndon et al. (1994)]. Soluções que façam uma ponte entre os dispositivos 2D e a interação 3D vêm sendo implementadas, para aumentar a usabilidade das aplicações tridimensionais.

Esse artigo apresenta uma arquitetura para construção de *widgets* 3D, chamados manipuladores, para aplicações de visualização de dados volumétricos e um conjunto específico de manipuladores para visualização interativa de

dados sísmicos, com a utilização do *mouse*.

A arquitetura define um conjunto de classes, e os relacionamentos entre elas, para construção dos manipuladores como componentes de *software* independentes e reaproveitáveis e para implementação de um modelo de tratamento de eventos. A idéia essencial do modelo é encapsular a lógica de processamento dos eventos de manipulação direta na implementação dos manipuladores.

Componentes convencionais da interface recebem eventos gerados sobre a área de desenho. Se a interface se encontra em estado de manipulação direta, eles são passados diretamente para os manipuladores. Caso contrário, os eventos são tratados pelos componentes, por exemplo, para movimentação da câmera.

Este é, essencialmente, o modelo adotado, de forma mais sofisticada, pelo *Open Inventor*, *software* padrão da *Silicon Graphics, Inc.*, e, também, por algumas outras bibliotecas. Procuramos identificar as classes essenciais para implementar o modelo de forma simples e extensível, para sua utilização em aplicações baseadas em bibliotecas gráficas convencionais, como o *OpenGL*. Aplicamos o modelo, em conjunto com outras classes não discutidas neste artigo, para desenvolver aplicações 3D de visualização interativa na área de E&P.

2 Trabalhos Anteriores

2.1 Brown University Graphics Group

O grupo *Brown University Graphics Group*, da Universidade de Brown, USA, vem trabalhando bastante com projetos de *widgets* 3D, especialmente nas áreas de modelagem e animação. O grupo propõe um *toolkit* interativo para construção de *widgets* 3D como combinações de primitivas [Stevens et al. (1994)]. Ponto, raio e plano são exemplos de primitivas. O comportamento de um *widget* 3D é definido pelos relacionamentos e restrições aplicados às primitivas combinadas.

O grupo desenvolveu *widgets* 3D para aplicação de deformações em objetos 3D [Snibbe et al. (1992)], para operações de modelagem *free-form* [Grimm et al. (1995)] e para visualização interativa de campos escalares em ambientes virtuais [Meyer et al. (1993)], entre outros.

2.2 *Open Inventor*

Open Inventor é um *toolkit* orientado a objetos, utilizado para criar aplicações gráficas 3D interativas [Strauss et al. (1992)] [Wernecke (1994)]. Ele provê uma biblioteca de classes para descrever e manipular cenas 3D e um mecanismo para modificar e estender as classes.

Cenas são descritas em uma estrutura de dados hierárquica (s*cene graph*), composta de nós que representam informações como *shapes*, materiais, transformações geométricas, câmera e luzes. A interação com os objetos da cena 3D é proporcionada por nós especiais, como manipuladores e *draggers*. Os manipuladores são subclasses dos nós editáveis, como os que representam luzes e transformações geométricas, e substituem esses nós no grafo, durante as interações. *Draggers* são os objetos que reagem aos eventos de interação, utilizados pelos manipuladores para alterar seus atributos editáveis. No fim da interação, o grafo é restaurado com o nó original, que tem seus atributos substituídos pelos atributos correntes do manipulador.

Um modelo semelhante é utilizado em VRML 2.0, na implementação dos *sensors*, cuja arquitetura é fortemente baseada no *Open Inventor* [Hartman et al. (1996)].

O *Open Inventor* é uma biblioteca de propósito geral. Estão disponibilizados manipuladores para editar nós representando transformações geométricas e luzes. A sua utilização em sistemas de visualização científica requer a implementação de subclasses para criar nós que representem as diversas técnicas de visualização e de manipuladores para editá-las.

O *Open Inventor* apresenta, no entanto, algumas características que dificultam seu uso para o tipo de aplicação visado. É um produto que tem que ser adquirido para os vários equipamentos e ambientes computacionais que se queira utilizar. Funcionar exclusivamente em *modo retained-mode*, baseando-se na estrutura de dados hierárquica para armazenar as informações necessárias ao *rendering* dos objetos da cena, limita sua utilização em aplicações de visualização científica envolvendo volume de dados elevado. Outra característica é não permitir acesso direto ao *framebuffer*, essencial na implementação de métodos eficientes de *rendering* direto de volume.

3 *Toolkit* para Desenvolvimento de Aplicações 3D Interativas na Área de E&P

A demanda por aplicações 3D na área de E&P, envolvendo grande volume de dados, motivou o desenvolvimento de um *toolkit* orientado a objetos, para facilitar o desenvolvimento dessas aplicações em um nível de abstração mais alto que o proporcionado pela utilização direta de bibliotecas como o *OpenGL* [Reis (1996)].

O *toolkit* se compõe de uma biblioteca extensível de classes e de um conjunto de serviços para gerenciar o relacionamento entre os objetos utilizados. Baseado em

idéias do *Open Inventor*, algumas de suas classes, como câmera, cena, luz e visualizador, têm correspondência direta com classes dessa biblioteca. Classes como contexto gráfico, escala de material, escala de cor e, especialmente, objeto gráfico e visualização, próprias para o domínio de aplicações a que o *toolkit* se destina, foram introduzidas.

Objetos gráficos são os objetos matemáticos com os quais as aplicações lidam. São exemplos de objetos gráficos em E&P: pontos esparsos, linhas poligonais (para representação de poços), grids 2D (superfícies funcionais que representam horizontes geológicos) e grids 3D (volumes com propriedades sísmicas ou de reservatórios).

As visualizações são formas de *rendering* para os objetos gráficos. São elas que aparecem na tela e com elas o usuário quer interagir. Um exemplo de visualização para superfícies funcionais é a visualização de contorno. Exemplos de visualizações para volumes são: fatias, isosuperfícies e o *rendering* volumétrico direto.

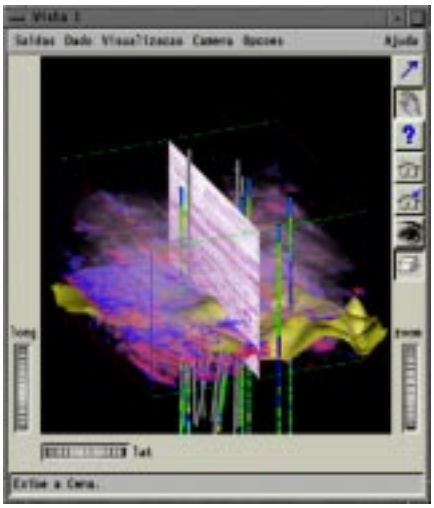

Figura 1. Visualizador com uma cena 3D.

Uma cena é composta de visualizações e, para ser exibida, é associada a um visualizador (*viewer*). Ela pode conter, também, manipuladores para editar as visualizações.

Um visualizador é um *widget* convencional de um sistema de janelas, que contém uma área de desenho para o *rendering* da cena e uma interface com o usuário para controle da câmera, definição de parâmetros de *viewing* e para execução de operações da aplicação. A Figura 1 apresenta um visualizador que exibe uma cena composta de uma visualização de fatia e de uma visualização volumétrica direta, para um dado volumétrico, e de visualizações de poços e de uma superfície geológica.

4 Arquitetura para Manipulação 3D

Um *widget* 3D é um objeto composto cujo comportamento é definido pelo comportamento e relacionamentos dos objetos mais simples que o compõem (primitivas, *handles* ou *draggers*). Seja qual for a implementação utilizada na construção de um *widget* 3D, interagir com ele requer reconhecer qual de seus objetos simples foi selecionado para interação, interpretar seu movimento de acordo com seu comportamento, manter os relacionamentos entre os objetos simples e alterar o objeto por ele controlado.

A arquitetura para manipulação 3D proposta acompanha o paradigma de orientação a objetos utilizado no desenvolvimento do *toolkit* para desenvolvimento de aplicações 3D interativas. Ela utiliza o modelo de eventos e as classes para interação 3D do *Open Inventor* como referência, mas com mecanismos de tratamento de eventos e conexão com os objetos da cena (visualizações) mais simples e fáceis de implementar.

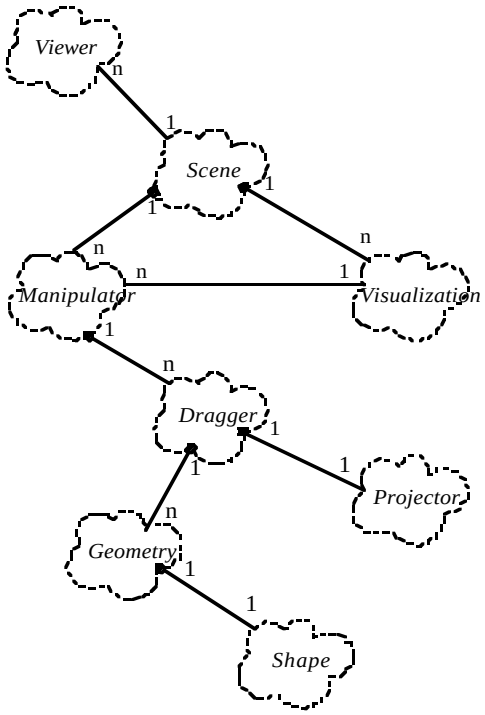

Figura 2. Diagrama de Classes.

O diagrama de classes apresentado na Figura 2 ilustra as principais classes envolvidas nas tarefas de interação por manipulação direta.

4.1 Manipuladores

Um manipulador é um *widget* 3D, presente na cena

associado a uma visualização, e que pode ser editado, resultando em alterações na visualização.

O usuário utiliza o *mouse* para interagir com o manipulador, que reage a eventos de botão pressionado / arrastado / solto, indicando, respectivamente, o início de uma interação, a interação propriamente dita e o fim da interação.

Funções *callbacks* são definidas no manipulador para que ele se comunique com a visualização durante uma interação: uma *callback* de início invocada quando ocorrem eventos de botão pressionado; uma *callback* de movimento, invocada quando ocorrem eventos de botão arrastado; e uma *callback* de fim, invocada quando ocorrem eventos de botão solto.

Um exemplo de manipulador, utilizado para movimentar uma visualização do tipo fatia de volume, é apresentado de forma esquematizada na Figura 3 e será referenciado pelas seções seguintes para ilustrar a composição dos manipuladores.

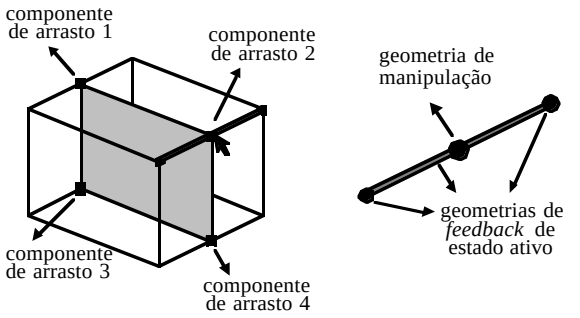

Figura 3. Manipulador para movimentação de fatia (a) e detalhe da geometria do componente de arrasto ativo (b).

4.2 Componentes de arrasto

Os manipuladores empregam componentes de arrasto (*draggers*) para compor sua geometria e suas edições complexas. Eles possuem geometria e reagem diretamente aos eventos de interação com o usuário, realizando edições simples como escala, rotação e translação.

Os componentes de arrasto utilizam funções *callbacks* para reagir aos eventos de interação. A função *callback* de início trata os eventos de botão pressionado, preparando o componente para uma interação. A função *callback* de movimento trata os eventos de botão arrastado, interpretando o movimento do componente. A função *callback* de fim trata os eventos de botão solto, encerrando a manipulação com o componente. Esses três tipos de função *callback* tratam a interação localmente. Existe ainda um quarto tipo de função *callback*, chamada *callback* de valor alterado, que é associada ao componente para que ele se comunique com seu manipulador, passando as informações sobre seu movimento.

No projeto de manipuladores, foram identificados os seguintes tipos de movimento para os componentes de arrasto: translação sobre uma linha; translação sobre um plano; translação em um volume; rotação sobre a superfície de uma esfera; e rotação sobre a superfície de um cilindro.

A Figura 3(a) ilustra a utilização de quatro componentes de arrasto na composição do manipulador para movimentação de fatia. Como o movimento da fatia é executado na direção perpendicular ao seu plano, o movimento a ser realizado por eles é o de translação sobre uma linha.

4.3 Projetores

Os componentes de arrasto utilizam projetores para mapear movimentos do *mouse* em movimentos 3D. O mapeamento é feito no sistema de coordenadas normalizadas e consiste em projetar a posição do *mouse* sobre a geometria de um objeto 3D, utilizado como restrição para o comportamento do componente (linha, plano, esfera, cilindro, ...). A posição calculada é transformada de coordenadas normalizadas para coordenadas do mundo. Sucessivos mapeamentos da posição do *mouse*, enquanto ele se desloca, definem os movimentos do componente em 3D.

Os componentes de arrasto do manipulador para movimentação de fatia se movimentam ao longo de uma linha 3D, correspondente a uma aresta do volume, utilizando projetores de linha para mapear seus movimentos.

4.4 Geometria

Os componentes de arrasto podem se apresentar com representações visuais diferentes em cada manipulador. O modelo geométrico (*shape*), o posicionamento na cena (através de transformações de translação e de rotação), o estilo de *rendering* (sólido ou *wireframe*) e o material especificam a geometria de cada componente.

São três os tipos de geometria que podem ser definidos: de manipulação, de *feedback* e de *feedback* de estado ativo. Uma única geometria de manipulação é definida e é ela que é utilizada na operação de *picking*. Os outros dois tipos de geometrias são utilizados para ajudar os componentes de arrasto a expressar sua função. Uma geometria de *feedback* é exibida junto com a geometria de manipulação e uma geometria de *feedback* de estado ativo é exibida enquanto o componente de arrasto está ativo, ou

seja, está sendo manipulado.

A Figura 3(a) mostra a geometria do manipulador para movimentação de fatia. Os três componentes de arrasto no estado inativo têm exibida somente a geometria de manipulação. O componente de arrasto selecionado para manipulação (componente 2) tem exibidas, além da geometria de manipulação, três geometrias de *feedback* de estado ativo, uma para indicar a direção do movimento e duas para indicar os limites do movimento (Figura 3(b)).

A Figura 4 ilustra as geometrias definidas na implementação do componente de arrasto ativo do manipulador para movimentação de fatia.

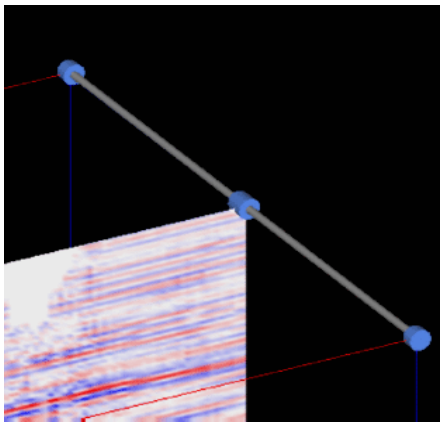

Figura 4. Geometrias definidas para o componente de arrasto ativo do manipulador para movimentação de fatia.

4.5 Mecanismo de Interação

Um visualizador pode trabalhar em dois modos de operação. No modo de *viewing*, o visualizador utiliza os eventos sobre a área de desenho para controlar a câmera associada. No modo de *picking*, os eventos que ocorrem sobre a área de desenho são interpretados como eventos de manipulação sobre algum objeto da cena. Neste modo, o visualizador passa os eventos diretamente para a cena.

A cena, ao receber um evento de botão do *mouse* pressionado, utiliza o mecanismo de *picking* da biblioteca gráfica *OpenGL* para verificar se ele ocorreu sobre algum manipulador. Utilizar esse mecanismo consiste em redesenhar as geometrias de manipulação dos componentes de arrasto dos manipuladores da cena no modo *Selection* do *OpenGL*, restringindo o desenho à uma região próxima ao cursor. Nesse modo, as informações de desenho não são enviadas ao *framebuffer* mas retornadas para a aplicação, na forma de registros que contêm uma hierarquia de identificadores das primitivas que interceptaram a região de desenho e os valores de z das interseções. Analisando essas informações, a cena reconhece o manipulador e o componente de arrasto sobre os quais o evento ocorreu.

A cena passa para o manipulador selecionado o evento e o identificador do componente de arrasto. O manipulador realiza as operações necessárias ao início de uma interação e encaminha o evento para o componente selecionado. O componente se prepara para o início da interação, executando operações como alterar sua geometria para indicar que está ativo e definir a posição inicial do movimento. Se necessário, o manipulador estabelece uma comunicação com a visualização ao qual está associado.

Os eventos de botão arrastado e de botão solto são passados diretamente para o manipulador identificado no início da interação que, por sua vez, passa os eventos diretamente para o componente de arrasto ativo.

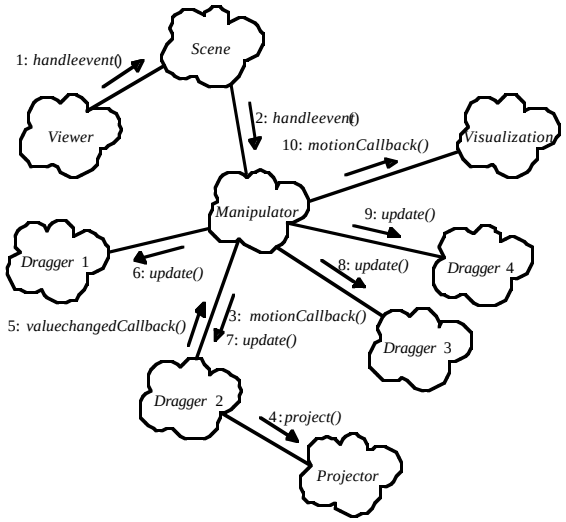

Figura 5. Diagrama de Objetos.

Nos eventos de botão arrastado, o componente de arrasto utiliza seu projetor para interpretar o movimento do *mouse* e informa o manipulador sobre seu movimento. O manipulador valida o movimento, atualiza seus componentes, para que os relacionamentos entre eles sejam mantidos, e, se necessário, interage com a visualização ao qual está associado. O diagrama de objetos apresentado na Figura 5 exemplifica a troca de mensagens entre os objetos envolvidos no tratamento de eventos de botão arrastado pelo manipulador para movimentação de fatia.

Um evento de botão do *mouse* solto encerra a interação. O componente de arrasto executa as operações necessárias ao fim da interação, como restaurar seu estado inativo, e o manipulador encerra a interação com a visualização.

5 Aplicação

Geocientistas trabalham na busca e desenvolvimento de reservas de óleo e gás e analisam dados de diversas fontes, dentre eles, os de análise sísmica.

Dados sísmicos são obtidos através da gravação de amplitudes sísmicas, que indicam o tempo necessário para que um sinal (onda sísmica), emitido por determinada fonte, reflita em uma ou mais descontinuidades do solo (superfícies que separam camadas de rochas não similares) e retorne a um ponto receptor. A aquisição de dados sísmicos é feita ao longo de determinadas direções, denominadas linhas sísmicas e que, agrupadas, formam o dado sísmico 3D.

Os valores escalares de amplitude dos sinais sísmicos são denominados amostras. Na visualização do dado sísmico 3D (grid 3D de amostras), as amostras são transformadas em primitivas coloridas (*pixels*, na visualização de fatias, ou *voxels*, na visualização volumétrica direta), para que as imagens 3D produzidas revelem as estruturas das camadas geológicas e permitam a identificação de potenciais reservatórios de óleo ou gás.

Explorar dados científicos envolve interagir com as suas representações: examiná-las, aplicar ações que as altere, questionar, comparar resultados e classificar. O trabalho de interpretação de dados sísmicos requer recursos para realçar áreas de interesse, visualizar subconjuntos do dado original e extrair geometrias como superfícies representando camadas geológicas, entre outros.

Essas necessidades definem um conjunto de manipuladores a ser implementado. Manipuladores para movimentar visualizações do tipo fatia nas direções do volume, definir subvolumes e investigar dados já estão implementados. Manipuladores para realçar áreas de interesse, através da edição seletiva da cor e opacidade das amplitudes sísmicas, e para definir direções arbitrárias para as fatias estão em fase de implementação.

5.1 Movimentação de Visualização de Fatia

O manipulador para movimentação de fatia do volume é composto de quatro componentes de arrasto, posicionados nas arestas do volume perpendiculares ao plano da fatia, que executam movimento sobre linha (Figura 6). O componente selecionado para manipulação (componente de arrasto ativo) exibe geometrias adicionais (*feedback*) para indicar a direção e os limites para seu movimento.

O projetor garante que o movimento do componente de arrasto ativo fique restrito à linha definida pela aresta sobre a qual ele está posicionado e o manipulador garante que o movimento fique restrito ao intervalo definido pelos vértices da aresta. A cada movimento do componente de arrasto ativo, o manipulador atualiza a posição dos demais componentes e passa para a visualização (fatia) sua nova posição.

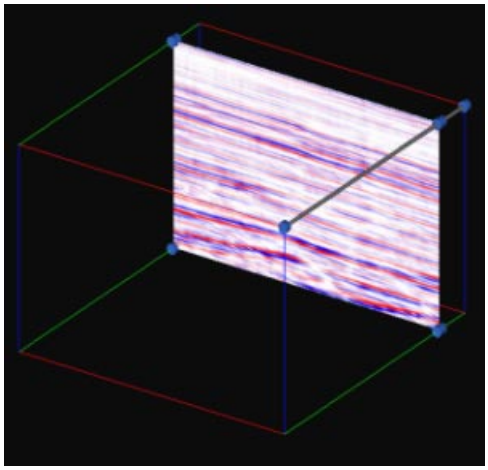

Figura 6. Manipulador para movimentação de fatia.

5.2 Definição de Subvolume

O manipulador para definição de subvolume, utilizado para visualização e edição de subconjuntos do dado original, tem a forma de uma caixa, cujas dimensões iniciais correspondem às dimensões do volume original, e emprega três tipos de componentes de arrasto.

Um esquema do posicionamento dos componentes de arrasto em cada face do manipulador é ilustrado na Figura 7. Um componente de arrasto posicionado em um vértice da face executa movimento sobre o plano da face e altera duas dimensões do volume ao mesmo tempo. Um componente de arrasto posicionado no meio de uma aresta executa movimento sobre a linha definida pela sua posição e a posição do componente na aresta oposta e altera uma dimensão do volume. Um terceiro tipo de componente de arrasto é a própria face e é utilizado para transladar todo o manipulador sobre o plano da face.

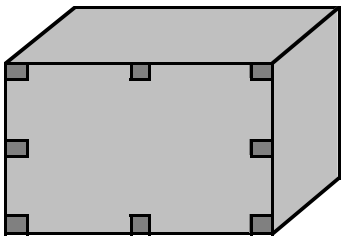

Figura 7. Posicionamento dos componentes de arrasto em uma face do manipulador para definição de subvolume.

A sequência de figuras, Figura 8 à Figura 10, ilustra a definição de um subvolume.

A Figura 8 apresenta o manipulador no estado inicial.

A Figura 9 ilustra a manipulação de um componente de arrasto de um vértice (indicado pela seta vermelha), que altera duas dimensões do volume. O projetor deste componente trata as restrições do seu movimento. O manipulador garante que o movimento não extrapole os limites do volume e que o volume não seja invertido. A cada movimento do componente de arrasto ativo, as posições dos outros componentes são atualizadas, para que sejam mantidos os relacionamentos entre eles. Durante a interação, a geometria de *feedback* utilizada é a de uma caixa em *wireframe*, indicando a posição e as dimensões correntes do subvolume.

Finalizada a interação, o manipulador passa para a visualização (volume) as coordenadas a serem utilizadas na geração do subvolume, ilustrado na Figura 10.

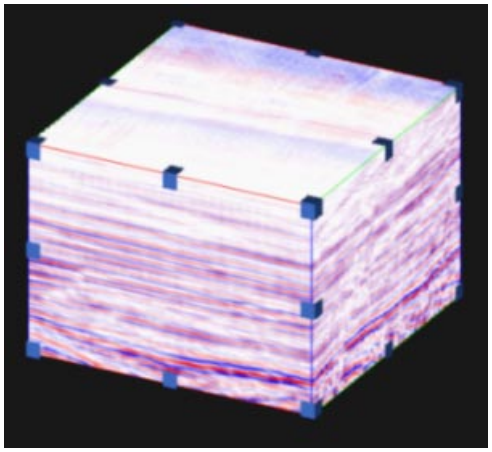

Figura 8. Manipulador para definição de subvolume.

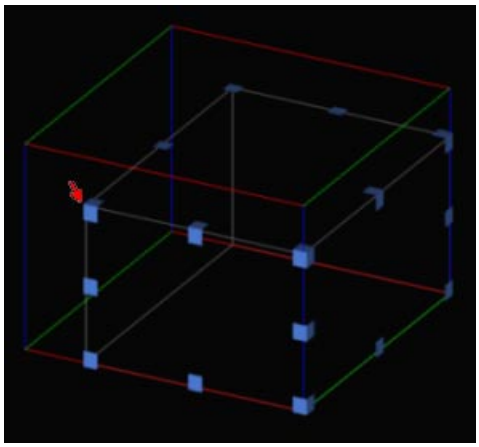

Figura 9. Manipulação de um componente de arrasto.

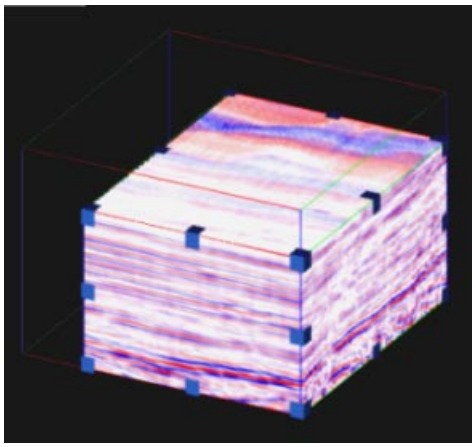

Figura 10. Subvolume definido.

5.3 Investigação de Dados

Um manipulador que se movimenta pelo volume foi implementado (Figura 11). Possíveis usos para ele são retornar informações sobre a amostra correspondente à posição corrente ou utilizar o valor dessa amostra como um valor de *threshold* para extração de isosuperfícies.

O manipulador é composto de seis componentes de arrasto: três que se movimentam sobre linha e três que se movimentam sobre plano. Os três componentes de arrasto que se movimentam sobre linha são sempre exibidos e são utilizados para transladar o manipulador nas três direções do volume. A geometria de manipulação de cada um desses componentes é um cilindro orientado longitudinalmente a um dos eixos do volume, com altura igual à dimensão do volume nesse eixo, e dois cubos indicando os limites do volume nessa direção. Os componentes de arrasto que se movimentam sobre plano têm a geometria de uma esfera que representa a localização do manipulador no volume. Apenas um deles fica disponível para manipulação. O componente de arrasto inicialmente exibido é o que se movimenta sobre o plano que passa pelo centro da esfera e é perpendicular ao eixo Z do volume. A tecla <Alt> é utilizada para alternar a manipulação entre os três componentes.

Em uma interação sobre um eixo, a geometria de *feedback* utilizada é um cilindro vermelho, indicando a posição do movimento. Na interação com a esfera, o plano sobre o qual ela se movimenta é desenhado em *wireframe* como *feedback* para o movimento.

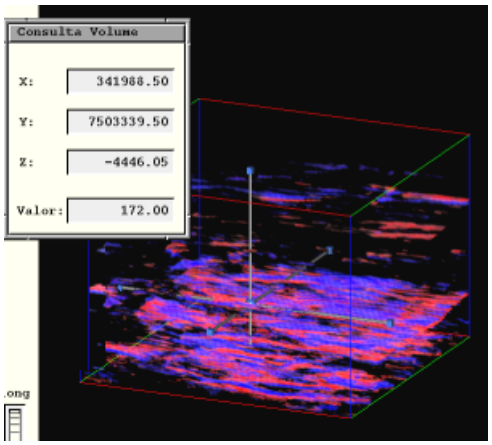

Figura 11. Manipulador para investigação de dados.

6 Conclusões e Trabalhos Futuros

Esse artigo mostra as possibilidades de utilização de *widgets* 3D em sistemas de visualização científica e, em particular, em sistemas de visualização e interpretação de dados de exploração e produção de óleo. Uma primeira aplicação, que faz um estudo sobre velocidades a usar em um processo chamado migração sísmica, está utilizando o manipulador para movimentação de fatia de volume na etapa de seleção das fatias do volume sísmico a serem utilizadas no estudo.

O conjunto inicial de manipuladores definido para implementação contempla necessidades atuais de manipulação de dados volumétricos. O enfoque de orientação a objetos utilizado permite que novos manipuladores, componentes de arrasto, projetores e modelos geométricos sejam implementados.

As técnicas de visualização comumente utilizadas na maioria dos sistemas de visualização científica podem se tornar pesadas quando aplicadas em conjuntos de dados muito extensos, restringindo a interatividade. Para garantir um uso mais efetivo das ferramentas de interação, deve-se buscar a utilização de meios para aumentar a velocidade dessas técnicas, mantendo-se algum tipo de visualização como *feedback*, tais como aproximações, multiresolução e refinamento incremental.

Agradecimentos

À equipe da Petrobras/E&P/GEREX/GETINF/GEDES, responsável pelo desenvolvimento de aplicações correlatas, atualmente em utilização na companhia.

A Beatriz Castier, em especial, que trabalha na implementação do *toolkit* para desenvolvimento de aplicações 3D na área de E&P.

Referências

D. B. Conner, S. S. Snibbe, K. P. Herndon, D. C. Robbins, R. C. Zeleznik, A. van Dam, *Three-Dimensional Widgets*, Proceedings of the 1992 Workshop on Interactive 3D Graphics, pp. 183-188, 1992.

C. Grimm, D. Pugmire, M. Bloomenthal, J. Hughes, E. Cohen, *Visual Interfaces for Solid Modeling*, Proceedings of UIST '95, ACM Press, November, 1995.

J. Hartman, J. Wernecke, *The VRML 2.0 Handbook*, Addison Wesley, 1996.

K. P. Herndon, A. van Dam, M. Gleicher, *Workshop on the Challenges of 3D Interaction, SIGCHI Bulletin*, vol. 26, no. 4, pp. 1-9, 1994.

L. P. Reis, *A Toolkit for Interactive Scientific Visualization*, Manuscrito, 1996.

S. S. Snibbe, K. P. Herndon, D. C. Robbins, D. B. Conner, A. van Dam, *Using Deformations to Explore 3D Widgets Design*, Computer Graphics (Proceedings SIGGRAPH'92) , vol. 26, no. 2, pp. 351-352, 1992.

M. P. Stevens, R. C. Zeleznik, J. F. Hughes, *An Architecture for an Extensible 3D Interface Toolkit*, Brown University, 1994.

P. S. Strauss, R. Carey, *An Object-Oriented 3D Graphics Toolkit*, Computer Graphics (Proceedings SIGGRAPH'92), vol. 26, no. 2, pp. 341-349, 1992.

C. Upson, T. Faulhaber, D. Kammins, D. Laidlaw, D. Schlegert, J. Vroom, R. Gurwitz, A. van Dam, *The Application Visualization System: a computational environment for scientific visualization*, IEE Computer Graphics and Applications, vol. 9, no. 7, pp. 30-42, 1992.

J. Wernecke, *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor*, Release 2, Addison Wesley, 1994.