PUC

Beatriz Castier

Visualização 3D de Unidades Geológicas Representadas por Subdivisões Espaciais

Dissertação de Mestrado

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 27 de abril de 1995

Beatriz Castier

Visualização 3D de Unidades Geológicas Representadas por Subdivisões Espaciais

Dissertação apresentada ao Departamento de Informática da PUC-Rio como parte dos requisitos para obtenção do título de Mestre em Informática: Ciência da Computação

Orientador: Luiz Fernando Martha

Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 27 de abril de 1995

BCld
564
92431

UC-64409-5

004
C351v
TESE/DC

A meus pais e irmão.

Agradecimentos

a Luiz Fernando Martha, orientador da dissertação, pelo apoio, entusiasmo e inestimáveis colaborações.

ao professor Marcelo Gattass, pelo interesse em aperfeiçoar o trabalho.

a Paulo Roma Cavalcanti, Luiz Fernando Martha, Nick Lehtola e João Luiz Elias Campos, pela utilização de parte dos seus programas.

aos geólogos Marcos Domingues e Olavo Colela Júnior, pelas colaborações na área de geologia.

aos amigos da Petrobrás e do TeCGraf, que, durante o desenvolvimento do trabalho, auxiliaram em tudo que foi possível, em especial a Raquel Oliveira Prates e Wanda Santos Teixeira.

ao Grupo de Tecnologia em Computação Gráfica (TeCGraf) da PUC-Rio, por ter facilitado a execução deste trabalho.

ao geólogo Benito Leonízio Fuschilo, ao analista Geraldo de Oliveira Santos e à Petrobrás, pela oportunidade de realização do curso de Mestrado.

Resumo

A geologia de petróleo estuda estruturas existentes abaixo da superfície terrestre, com o objetivo de descobrir reservas de óleo e gás. Ferramentas bidimensionais, como mapas de contorno e seções geológicas, são freqüentemente empregadas no trabalho de interpretação geológica. Quando representadas em três dimensões, as complexas relações espaciais em subsuperfície podem ser compreendidas com maior facilidade.

Esta dissertação avalia o emprego da metodologia de construção de subdivisões espaciais para a representação e visualização de blocos diagrama (modelos 3D da subsuperfície). Esta metodologia modela objetos heterogêneos em dimensão e em material, subdividindo o espaço em partes homogêneas e descrevendo a forma de "colagem" dessas partes.

Através do desenvolvimento de um protótipo, são estudadas estratégias de visualização e de manipulação do bloco diagrama. A exibição do bloco procura tirar partido das características da estrutura de dados topológica *radial-edge* e de recursos de *rendering* (cor, remoção de superfícies escondidas e iluminação). É discutida uma taxonomia para a manipulação e visualização de objetos 3D através de dispositivos 2D. A classificação leva em conta o tipo de referencial de movimento (*Screen-based, Walk-through* ou *Object-based*), o tipo de movimento (Translação ou Rotação) e do ajuste de perspectiva e planos de cerceamento. São apresentadas implementações básicas dos modelos *Screen-based* e *Walk-through*. É proposto um modelo para manipulação baseado na orientação do objeto (*Object-based*), que é ideal para objetos que possuem uma caixa envolvente, como é o caso do bloco diagrama. É sugerido um modelo de corte do bloco, que procura criar uma maneira intuitiva

de posicionamento do plano de corte e, também, explorar recursos da metodologia de subdivisões espaciais.

Abstract

In the search and development of oil and gas reservoirs, one of the tasks of subsurface petroleum geology is the mapping of unseen structures that may exist underneath the earth's surface. Earth scientists use 2D tools, contour maps and cross sections, to interpret geological data. However, spatial relationships presented in three dimensions are easier to interpret than if they are shown as a series of 2D representations.

The aim of this work is to evaluate the use of a spatial subdivision methodology in the representation and visualization of 3D geological models (called block diagrams). This methodology models heterogeneous objects (with different parts made of different materials), by decomposing the space into homogeneous parts, and then describing the way they are connected.

To evaluate visualization and manipulation strategies of block diagrams, a prototype interactive-graphics system was developed. Block diagram visualization exploits, as much as possible, the underlying topological data structure characteristics and the rendering functions avaiable in powerful workstations. The work decribes a taxonomy for interactive manipulation of 3D objects using 2D input devices. The proposed classification is based on the type of movement control (Screen-based, Walk-through or Object-based), on the type of geometric transformation (translation or rotation), and on the control of viewing parameters (clipping plane positions with respect to the eye point and distance from eye to reference point). Basic Screen-based and Walk-through models are presented. It is proposed a model for Object-based manipulation, which is ideal for objects which present a natural bounding box, such as a block diagram. The suggested strategy for cutting the 3D model includes the design

of an intuitive cutting-plane positioning and also exploits the spatial subdivision methodology.

Sumário

1. INTRODUÇÃO **1**

1.1 DEFINIÇÃO DO PROBLEMA 1

1.2 OBJETIVOS 2

1.3 PROPOSTAS 2

1.4 VISÃO GERAL DO TRABALHO 4

2. MODELAGEM DE SÓLIDOS **5**

2.1 MODELAGEM 5

2.2 DEFINIÇÃO DE SÓLIDO 7

2.3 ESQUEMAS DE REPRESENTAÇÃO 9

2.4 EXEMPLOS DE MODELADORES GEOMÉTRICOS NA ÁREA DE GEOLOGIA 12

3. METODOLOGIA DE SUBDIVISÕES ESPACIAIS **14**

3.1 SUBDIVISÕES DO ESPAÇO 14

3.2 REPRESENTAÇÕES PARA SUBIVISÕES DO ESPAÇO 16

3.2.1 Elementos Topológicos *19*

3.2.2 Estrutura de Dados *21*

3.2.3 Operadores Topológicos *24*

3.3 O SISTEMA DE SUBDIVISÕES ESPACIAIS 27

4. EXIBIÇÃO GRÁFICA DE INFORMAÇÕES **29**

4.1 CARACTERÍSTICAS DA ESTRUTURA RADIAL-EDGE 29

4.1.1 Travessia do Modelo *29*

4.1.2 Exibição de Atributos do Modelo *31*

4.2 ALGORITMOS DE RENDERING 31

4.2.1 Remoção de Superfícies Escondidas *32*

4.2.2 Efeitos de Iluminação *32*

5. MANIPULAÇÃO INTERATIVA E VISUALIZAÇÃO DE OBJETOS 3D 36

5.1 PARÂMETROS DE VISUALIZAÇÃO EM 3D 37

5.2 MOVIMENTOS BÁSICOS DA CÂMERA 39

5.3 UMA TAXONOMIA PARA O CONTROLE DE VISUALIZAÇÃO EM 3D 42

5.4 MODELOS BÁSICOS DE MANIPULAÇÃO 45

5.4.1 Modelo Screen-based Básico *47*

5.4.2 Modelo Walk-through Básico *51*

5.5 MANIPULAÇÃO NATURAL DE OBJETOS COM CAIXA ENVOLVENTE 52

5.6 RESULTADOS DA TAXONOMIA 56

5.7 ESTRATÉGIA DE CORTE DO MODELO 57

6. O PROTÓTIPO DE VISUALIZAÇÃO 3D DE UNIDADES GEOLÓGICAS 62

6.1 ORGANIZAÇÃO DO PROTÓTIPO 62

6.2 RESULTADOS EXPERIMENTAIS 64

7. CONCLUSÃO 68

APÊNDICE A. TRANSFORMAÇÕES GEOMÉTRICAS PARA VISUALIZAÇÃO EM 3D 71

A.1. PARÂMETROS DE VISUALIZAÇÃO EM 3D 71

A.2 TRANSFORMAÇÃO DO ESPAÇO DE MODELAGEM PARA O SISTEMA DA CÂMERA 73

A.3 PERSPECTIVA CANÔNICA E ESPAÇO DA TELA 74

A.4 NORMALIZAÇÃO DE COORDENADAS 77

REFERÊNCIAS BIBLIOGRÁFICAS **80**

Lista de Figuras

FIGURA 2-1 TRÊS NÍVEIS DE ABSTRAÇÃO NA MODELAGEM. 6

FIGURA 2-2 ESQUEMA DE REPRESENTAÇÃO. 7

FIGURA 2-3 SÓLIDO COM PORÇÕES DE DIMENSÕES MENORES PENDENTES. 8

FIGURA 2-4 CONDIÇÕES *NON-MANIFOLD* NA ARESTA (A) E NO VÉRTICE (B). 8

FIGURA 3-1 COMPLEXOS GEOMÉTRICOS AMBÍGUOS. 15

FIGURA 3-2 ORDENAÇÃO RADIAL DAS FACES AO REDOR DA ARESTA. 17

FIGURA 3-3 HIERARQUIA DOS ELEMENTOS NA ESTRUTURA *RADIAL-EDGE*. 19

FIGURA 3-4 ORGANIZAÇÃO DA ESTRUTURA *RADIAL-EDGE*. 22

FIGURA 3-5 ARQUITETURA DO SSE. 27

FIGURA 4-1 EXIBIÇÃO DE UNIDADES GEOLÓGICAS (A) E DE HORIZONTES (B). 31

FIGURA 4-2 EXIBIÇÃO DAS REGIÕES DO MODELO. 34

FIGURA 4-3 EXIBIÇÃO DAS FACES DO MODELO QUE NÃO PERTENCEM À REGIÃO EXTERNA 34

FIGURA 4-4 EFEITOS DE ILUMINAÇÃO: EM (A), A LUZ 0 ESTÁ LIGADA; EM (B), ESTÁ DESLIGADA 35

FIGURA 5-1 MODELO DE CÂMERA E FRUSTUM DE VISÃO. 37

FIGURA 5-2 UM OBJETO SIMPLES VISTO PELA POSIÇÃO INICIAL DA CÂMERA. 40

FIGURA 5-3 ROTAÇÃO DA CÂMERA EM TORNO DE EIXO VERTICAL. 41

FIGURA 5-4 ROTAÇÃO DA CÂMERA EM TORNO DE EIXO HORIZONTAL 41

FIGURA 5-5 TRANSLAÇÃO VERTICAL DA CÂMERA. 41

FIGURA 5-6 TRANSLAÇÃO HORIZONTAL DA CÂMERA. 41

FIGURA 5-7 ROTAÇÃO AXIAL DA CÂMERA. 42

FIGURA 5-8 TRANSLAÇÃO RADIAL DA CÂMERA. 42

FIGURA 5-9 UMA TAXONOMIA PARA O CONTROLE DE PARÂMETROS DE VISUALIZAÇÃO. 43

FIGURA 5-10 CÁLCULO DO ÂNGULO DE ROTAÇÃO EM TORNO DO EIXO AXIAL DA CÂMERA. 49

FIGURA 5-11 ARESTAS DA CAIXA ENVOLVENTE E AS SUAS DIREÇÕES TANGENTES. 53

FIGURA 5-12 NORMAIS DAS FACES DA CAIXA ENVOLVENTE. 55

FIGURA 5-13 CERCEAMENTO DO OBJETO. 58

FIGURA 5-14 NA INSERÇÃO DO RETALHO S À SE (A), S É DECOMPOSTO EM RETALHOS MAIS SIMPLES (B) E ADICIONADOS À SE COM OPERADORES TOPOLÓGICOS. 59

FIGURA 5-15 DETERMINAÇÃO DO RETALHO DA SEÇÃO 60

FIGURA 6-1 ESQUEMA DA TELA INICIAL DO PROTÓTIPO. 62

FIGURA 6-2 ATRIBUIÇÃO DE LITOLOGIA À UNIDADE GEOLÓGICA SELECIONADA. 65

FIGURA 6-3 AMPLIAÇÃO DE PARTE DO BLOCO ATRAVÉS DA OPÇÃO DE *ZOOM*. 65

FIGURA 6-4 EXIBIÇÃO DOS HORIZONTES DO MODELO. 66

FIGURA 6-5 REPRESENTAÇÃO DO BLOCO DIAGRAMA ATRAVÉS DE ARAMES. 66

FIGURA 6-6 UMA UNIDADE GEOLÓGICA É ESCONDIDA, PARA QUE DETALHES DE OUTRAS UNIDADES POSSAM SER VISTOS 67

FIGURA 6-7 O CORTE REVELA A EXISTÊNCIA DE UMA UNIDADE GEOLÓGICA NO INTERIOR DO BLOCO DIAGRAMA. 67

FIGURA A-1 MODELO DE CÂMERA E FRUSTUM DE VISÃO. 72

FIGURA A-2 PERSPECTIVA CANÔNICA. 75

Lista de Tabelas

TABELA 3-1 RELACIONAMENTOS DE ADJACÊNCIA 18

TABELA 3-2 OPERADORES QUE AGEM SOMENTE SOBRE AS FACES DE UMA SE. 25

TABELA 3-3 OPERADORES PARA CRIAR ARAMES E ADICIONAR FACES LIGADAS A ARESTAS OU ARAMES ESPECÍFICOS. 26

TABELA 4-1 RELAÇÃO ENTRE OS OBJETOS GEOMÉTRICOS, TOPOLÓGICOS E GEOLÓGICOS. 30

1. Introdução

A aplicação de técnicas de construção e visualização de modelos geológicos 3D na exploração e produção de petróleo tem sido assunto de interesse das comunidades de Geologia e Computação Gráfica e será o tema estudado neste trabalho.

1.1 Definição do Problema

A geologia de petróleo visa a descobrir e desenvolver reservas de óleo e gás. Estruturas existentes a muitos metros abaixo da superfície são estudadas com o objetivo de encontrar possíveis formações que sirvam como armadilhas de hidrocarbonetos. Nesse estudo, é fundamental que sejam integradas e avaliadas todas as informações disponíveis sobre a região geológica. Intérpretes (geólogos, geofísicos e engenheiros de petróleo) identificam e classificam regiões da subsuperfície, utilizando dados de poços, de sísmica e de produção de petróleo.

Termos como unidades geológicas, horizontes e blocos diagrama serão freqüentemente utilizados ao longo da dissertação. **Unidades geológicas** são definidas, para os objetivos deste trabalho, como regiões da subsuperfície identificadas de acordo com alguma característica de interesse geológico, como, por exemplo, tipo da rocha, idade da rocha, conteúdo fossilífero ou tipos e características dos fluidos. Serão chamadas de **horizontes** as superfícies que separam unidades geológicas.

Mapas de contorno e seções geológicas são as ferramentas freqüentemente utilizadas no trabalho de interpretação, para obtenção de uma idéia da geometria das unidades geológicas. **Mapas de contorno** representam uma superfície em um plano horizontal através de linhas de contorno, linhas que conectam pontos de igual valor. **Seções geológicas** são

planos verticais que cortam a crosta terrestre. Usadas em conjunto com mapas, ajudam intérpretes a imaginar a subsuperfície em três dimensões.

A interpretação de informações 3D com figuras 2D é uma tarefa difícil. As relações espaciais em subsuperfície podem ser observadas com mais facilidade, quando representadas em três dimensões. Esta é a motivação principal para esta dissertação que descreve o desenvolvimento de um protótipo para visualização de blocos diagrama.

Um **bloco diagrama** pode ser entendido como um volume retirado da subsuperfície, formado por vários materiais, ou seja, por várias unidades geológicas. As unidades resultantes da interpretação podem ser bastante irregulares, com cavidades e limitadas por superfícies descontínuas.

1.2 Objetivos

A dissertação tem como objetivo estudar o emprego de uma metodologia (conjunto formado por estrutura de dados e procedimentos) capaz de criar e manter objetos que possuam as características de um bloco diagrama.

Sendo o bloco um objeto tridimensional, como tal ele deve ser visto e manipulado pelo usuário do protótipo de Visualização 3D de Unidades Geológicas. Assim, outro objetivo deste trabalho é sugerir estratégias de visualização e de manipulação que, explorando as potencialidades da metodologia adotada, proporcionem ao usuário a sensação tridimensional de volume do bloco diagrama, com todas as suas características e nuances.

1.3 Propostas

Levando-se em consideração estes objetivos, o desenvolvimento do protótipo envolve três aspectos principais: o emprego de uma estrutura de dados ideal para representar blocos

diagrama; a exibição gráfica do bloco diagrama e a criação de uma interface para manipulação de objetos tridimensionais.

Modeladores de sólidos convencionais (CSG, B-Rep) [Mäntylä (1988)] tratam de objetos que seguem rigidamente a definição de sólidos. Estes são compostos por um único material, não possuem estrutura interior, nem partes de dimensões menores pendentes. Como o protótipo lida com blocos formados por diferentes unidades (por vários materiais), é necessária a utilização de uma estrutura de dados capaz de tratar objetos que estendam a noção original de sólido. Uma forma utilizada para modelar objetos heterogêneos em dimensão (com partes pendentes) e em material (com camadas) é descrever a forma de colagem dessas partes [Cavalcanti et al. (1993)]. Em sua tese, Cavalcanti (1992) sugere uma metodologia para construção de sólidos e cria as bibliotecas do SSE (Sistema de Subdivisões Espaciais) que gerenciam a geração de sólidos na estrutura *radial-edge*. Na dissertação, propõe-se que o bloco diagrama seja representado como uma subdivisão espacial e que sejam empregadas as bibliotecas do SSE no desenvolvimento do protótipo.

É importante que as relações espaciais em subsuperfície sejam facilmente compreendidas pelo usuário da aplicação. Para que as informações contidas no bloco diagrama sejam comunicadas de forma clara e para que o usuário tenha a percepção de volume, são usados recursos de *rendering,* como cor, remoção de superfícies escondidas e efeitos de iluminação.

Para o intérprete de geologia, além de visualizar o bloco, é importante manipulá-lo: a rotação do bloco permite a obtenção de novos detalhes; o corte, a observação do interior do volume. A dissertação estuda alguns modelos de manipulação de objetos tridimensionais através de dispositivos de entrada 2D. Os modelos sugeridos para translação, rotação e corte

do bloco procuram sempre transmitir ao usuário a sensação de manipulação de um objeto 3D. O modelo proposto para manipulação de objetos com caixa envolvente procura simular, na translação e na rotação, os movimentos de uma pessoa ao transladar ou rodar um sólido em suas mãos. No corte, o plano que secciona o bloco é paralelo ao plano da tela e pode ser movido para frente ou para trás. O resultado da seção, ao ser exibido junto com as demais partes do sólido que não são escondidas pelo plano de corte, dá ao usuário a impressão sólido-volumétrica do bloco diagrama.

Recursos de visualização e de manipulação do bloco diagrama podem aumentar a produtividade dos geólogos, ao permitir uma melhor compreensão dos relacionamentos espaciais em subsuperfície e ao possibilitar a deteção de incongruências geométricas nos resultados de interpretações.

1.4 Visão Geral do Trabalho

O capítulo 2 introduz os conceitos utilizados em Modelagem de Sólidos e apresenta características de alguns tipos de modeladores e exemplos destes na área de Geologia. O capítulo 3 explica o modelo matemático de subdivisões do espaço e descreve o SSE (Sistema de Subdivisões Espaciais). O capítulo 4 analisa características da estrutura *radial-edge* e algoritmos de *rendering* que influenciam a exibição gráfica de informações do bloco diagrama. No capítulo 5, é proposta uma taxonomia para o controle de visualização em 3D e são apresentadas estratégias para rotação, translação e corte do bloco diagrama. O capítulo 6 mostra resultados do desenvolvimento do protótipo. O capítulo 7 apresenta conclusões e sugestões para futuros trabalhos. O apêndice A complementa o capítulo 5, resumindo as transformações geométricas utilizadas para a visualização do objeto na tela.

2. Modelagem de Sólidos

A modelagem de sólidos é a área da modelagem geométrica responsável pela criação e processamento, em computador, de representações não ambíguas de objetos sólidos [Requicha-Rossignac (1992)]. Essas representações devem responder, através de algoritmos, questões arbitrárias sobre as propriedades geométricas do objeto modelado [Requicha-Voelcker (1982)].

A pesquisa em modelagem geométrica começou em meados da década de 60. Uma das mais antigas técnicas de modelagem geométrica é a de arame (*wireframe modeling*), que representa objetos por arestas e vértices. Nos anos 70, pesquisadores reconheceram que os sistemas CAD/CAM requeriam representações geométricas mais poderosas que a dos modelos de arame utilizados até então. Na metade dos anos 70, a primeira geração de sistemas de modelagem de sólidos revelou limites e potencialidades da área. No final da mesma década, a teoria sobre modelagem de sólidos havia avançado e muitos algoritmos e aplicações haviam sido desenvolvidos, permitindo o surgimento, nos anos 80, de vários sistemas voltados para o mercado industrial.

Os sistemas para modelagem da geometria de objetos sólidos têm grande utilização em áreas como engenharia, arquitetura, computação gráfica, visão por computador, geologia, enfim, em áreas que tratam de fenômenos espaciais.

2.1 Modelagem

A finalidade da contrução de modelos é facilitar a observação de objetos do mundo real. Modelos são úteis no estudo de objetos que ainda não existem, que estão em fase de projeto ou de objetos que, por razões físicas, não podem ser diretamente observados.

Uma divisão clássica da modelagem [Requicha (1980)] em três níveis é mostrada abaixo (Figura 2-1).

Figura 2-1 Três níveis de abstração na modelagem.

Os **objetos físicos** são objetos do mundo real. A complexidade desses objetos é bastante grande e, por isso mesmo, é impraticável representar todas as suas características em meio computacional. Os **objetos matemáticos** são idealizações de objetos físicos. A caracterização desses objetos é obtida através da utilização de conceitos provenientes da teoria matemática. Uma vez definidos os objetos matemáticos, a eles são atribuídas **representações.** Estas consistem em um conjunto de estruturas de dados e em um conjunto de procedimentos para manipulação das estruturas.

A relação entre os objetos matemáticos e as representações é conhecida por **esquema de representação**. Um **esquema de representação** (Figura 2-2) é definido, formalmente, como uma relação $s:M \rightarrow R$.

M é o **espaço de modelagem matemático**. R , o **espaço de representação**, é uma coleção de todas as **representações sintaticamente corretas. Representações sintaticamente corretas** são estruturas simbólicas finitas construídas a partir de símbolos de um alfabeto, de acordo com regras sintáticas. O espaço de representação R pode ser visto como uma linguagem gerada por alguma gramática. O **domínio** de s é D, isto é, o conjunto

de elementos de ***M*** que podem ser representados via ***s***. A **imagem** de ***s*** é ***V***, isto é, o conjunto de representações que são imagens de elementos de ***D***.

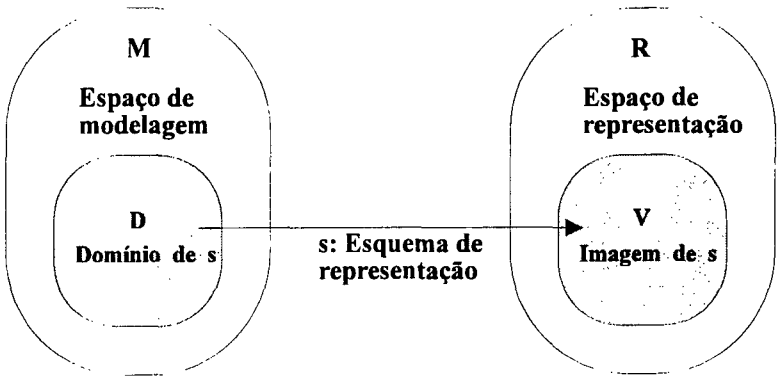


Figura 2-2 Esquema de representação.

Modelos matemáticos de sólido e alguns esquemas de representação utilizados em modelagem de sólidos serão apresentados nas próximas seções.

2.2 Definição de Sólido

Como modelos matemáticos de sólidos são propostos conjuntos **semi-algébricos, limitados, homogeneamente tridimensionais (regulares)**. Estes conjuntos são chamados de **conjuntos-r** [Requicha-Rossignac (1992)].

Um conjunto **semi-algébrico** é o resultado de um número finito de operações boolenas aplicado a semi-espaços definidos por inequações algébricas, isto é, semi-espaços da forma $H=\{(x,y,z) \mid p(x,y,z) <= 0\}$, onde ***p*** é uma função polinomial.

Conjunto **limitado** é aquele que está contido em um esfera de raio ***r***, isto é, ocupa uma porção finita do espaço.

Um conjunto é **regular** se não tem porções de dimensões menores pendentes (Figura 2-3).

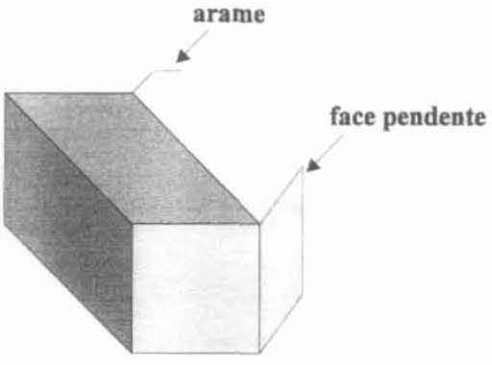


Figura 2-3 Sólido com porções de dimensões menores pendentes.

Os conjuntos-r são comumente aceitos como modelos adequados para sólidos. Alguns autores, porém, adotam uma posição mais rígida: definem sólidos como conjuntos-r cujas fronteiras são variedades de uma dimensão menor que o espaço que define o conjunto. Estes sólidos são denominados ***manifold***. Em um sólido *manifold* (Figura 2-4), cada aresta é compartilhada por exatamente duas faces e cada vértice tem precisamente um cone formado por faces adjacentes.

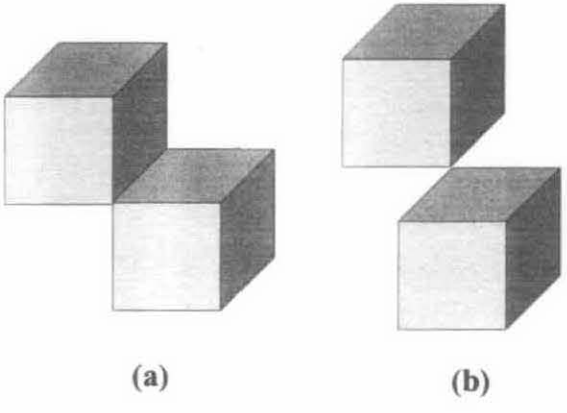


Figura 2-4 Condições *non-manifold* na aresta (a) e no vértice (b).

Muitas aplicações científicas requerem um domínio geométrico que estenda a noção de sólido. Os métodos usuais de modelagem sólida (**B-Rep**, **CSG**) têm problemas ao modelar objetos heterogêneos em dimensão e em material. Como exemplo de objetos heterogêneos em dimensão têm-se sólidos com arestas e faces pendentes. Objetos heterogêneos em material são formados por várias regiões com materiais de diferentes propriedades.

Uma forma adequada de tratar objetos heterogêneos é modelá-los como conjuntos de subconjuntos. Atualmente, muitos esforços de pesquisa têm se concentrado em representações onde o espaço é subdividido em partes homogêneas e é descrita a forma de colagem dessas partes.

2.3 Esquemas de Representação

Os esquemas de representação podem ser divididos em três grandes categorias: de decomposição, construtivos e de fronteira [Requicha-Rossignac (1992)].

Os **modelos de decomposição** descrevem os sólidos através de primitivos (sólidos mais simples) que são mantidos unidos por meio de operação de colagem. Os tipos de primitivos usados e os modos de colagem levam a variações nos modelos de decomposição: na enumeração exaustiva, voxels, pequenos cubos, dividem o espaço de forma regular; as decomposições do espaço podem ser feitas de forma hierárquica, como no caso das *octrees*; nas decomposições celulares, são usados vários tipos de primitivos e o espaço é decomposto de forma irregular. Os modelos de decomposição são de grande utilidade no processamento de imagem, visão por computador e robótica. Essas representações ganharam muita popularidade devido à simplicidade e à existência de algoritmos eficientes. No entanto, uma grande quantidade de memória é necessária para obtenção de resultados acurados.

Os **modelos construtivos** descrevem os sólidos através de operações booleanas em sólidos mais simples ou em semi-espaços. O modelo mais conhecido é o **CSG** (*Constructive Solid Geometry*). Os primitivos **CSG** são sólidos e as operações são movimentos rígidos (translação, rotação e escala) ou operações boolenas regularizadas de união, diferença e interseção. Representações **CSG** são árvores binárias, onde os nós não terminais representam operações e os nós terminais correspondem a primitivos. As representações **CSG** sempre correspondem a objetos válidos dentro do conceito de conjuntos-r. Elas requerem pouco espaço de memória, pois apenas a árvore de combinação de primitivos é armazenada. O processo construtivo de criar um sólido mais complexo a partir de objetos tridimensionais pré-existentes possibilita uma boa forma de interação com o usuário. Para criação de saídas gráficas, obtenção de informações topológicas e de cálculos geométricos, a árvore **CSG** precisa ser avaliada, o que é dispendioso em termos de execução.

Os **modelos de fronteira** (**B-Rep** - *Boundary Representation*) descrevem o sólido através da representação das superfícies que o delimitam. Embora uma simples enumeração das faces do sólido seja suficiente para descrevê-lo sem ambigüidade, informações adicionais de vizinhança e de orientação são armazenadas para ajudar na travessia da estrutura e responder, eficientemente, questões topológicas (de adjacência entre elementos). Os modelos de fronteira têm grande poder de representação, pois uma grande variedade de operações pode ser usada. A obtenção de informações topológicas é bastante rápida e fácil. No entanto, a complexa estrutura de dados de uma representação de fronteira torna elevado o gasto de memória. Sem a utilização de outras representações, a criação de um sólido, em um modelo de fronteira, através de interação com o usuário, é bastante difícil. Um grande problema da representação de fronteira é que a garantia de integridade geométrica do sólido é bastante complexa.

Nenhum dos três enfoques é, indiscutivelmente, superior aos demais. Isto motiva o uso simultâneo de múltiplas representações em alguns sistemas de modelagem geométrica, na tentativa de aproveitar o melhor de cada uma delas. Muitos modeladores têm empregado, conjuntamente, **B-Rep** e **CSG**.

Todos os tipos de modelador mencionados acima representam objetos que seguem, rigorosamente, a definição de sólidos (conjunto-r ou *manifold*). Esses objetos são compostos de um único material, não têm estrutura interior (cavidades consideradas fora da fronteira) nem partes pendentes e, no caso de sólidos *manifold*, as suas superfícies limitantes são localmente planas. Essa série de limitações impede a modelagem de diversos objetos importantes em algumas aplicações científicas.

Modeladores para objetos heterogêneos em dimensão e em material têm sido tema de pesquisa. Embora algumas estruturas de dados já fossem capazes de tratar agregados de objetos, somente mais recentemente, estudo teórico rigoroso tem sido desenvolvido sobre o assunto. Dois resultados desses estudos são citados abaixo.

Rossignac e O'Connor (1990) propõem a representação de objetos heterogêneos através de **SGC**s (*Selective Geometric Complexes*). Primeiramente, um subconjunto do espaço é subdividido em células que obedecem a determinadas condições. Um conjunto válido de células forma um complexo geométrico. Posteriormente, células do complexo geométrico, correspondentes ao objeto a ser representado, são selecionadas.

Requicha e Rossignac (1991) criaram o esquema de representação **CNRG** (*Constructive Non-Regularized Geometry*), que amplia o esquema **CSG** em vários aspectos [Requicha-Rossignac (1991)]. Objetos CNRG são conjuntos de componentes mutuamente disjuntas. Componentes são conjuntos de pontos do $\Re^n$, não necessariamente conexos nem

regulares. Grafos **CNRG** são árvores binárias. Os nós terminais são primitivos CNRG, que podem ser formados por mais de uma componente. Os nós internos correspondem a objetos intermediários obtidos através de operações booleanas, topológicas, de simplificação ou de filtragem.

Representações baseadas em *features* (características locais) têm sido estudadas para aplicação em projetos e construção de peças de engenharia. Primitivos usuais em modelagem (linhas, cilindros, paralelepípedos) nem sempre são adequados para engenharia. As *features* são formas mais significativas para projetos de peças mecânicas. Embora não haja uma definição precisa, muitos pesquisadores consideram *features* como sólidos que podem ser subtraídos (formando reentrâncias) ou adicionados (formando saliências) a outro sólido.

2.4 Exemplos de Modeladores Geométricos na Área de Geologia

Gresmod (*Geometric Reservoir Modeler*) [Sabella-Carlbom (1989)] é um sistema orientado a objetos para a modelagem de dados empíricos e tem como objetivos construir, manipular e exibir modelos geométricos de reservatórios. Para atender aos requisitos da modelagem de dados de reservatório, foi utilizada uma extensão do tradicional esquema CSG, denominada ECSG (*Extended Constructive Solid Geometry*). O modelo geométrico é especificado e modificado através de um editor gráfico ECSG.

O sistema **SGM** (*Stratigraphic Geocellular Modeling*) [Denver-Phillips (1990)] tem como objetivo modelar propriedades físicas do reservatório (porosidade, temperatura etc). Superfícies importadas de sistemas de mapeamento servem como arcabouço para a criação de uma malha usada para a interpolação de valores do dado mapeado. A malha é constituída por várias camadas (que se acomodam entre as superfícies) e estas são subdivididas em muitos milhares de células. SGM exibe perspectivas, seções e mapas do modelo.

3D-Aims da GX Technology é um sistema de modelagem 3D criado para simulação sísmica e visualização da subsuperfície. Uma característica de 3D-Aims é a utilização de representação do modelo baseada em retalhos (*patch-based model representation*). Os retalhos são colados, formando as superfícies que compõem o modelo geológico.

O sistema **Recon** [Ferraz (1993)] aplica modelagem geométrica à geologia estrutural em duas dimensões. Técnicas de balanceamento de seções geológicas estudam a restauração de uma estrutura geológica atual à sua conformação original do passado. Para a implementação, em computador, dessas técnicas, é utilizada a estrutura de dados topológica *half-edge*. A estrutura armazena a subdivisão planar que representa a seção geológica. Um conjunto de operadores criam e mantêm a subdivisão planar, modelando a formação geológica.

Pode parecer estranho citar o **Recon** como exemplo de modelagem de sólidos, já que é um sistema que trata de problemas 2D. Torna-se importante esclarecer que a estrutura *half-edge* foi criada, originalmente, para representar a fronteira de sólidos *manifold*. No entanto, a razão principal para a citação é a existência de uma analogia entre as estruturas *half-edge* e *radial-edge*. A *half-edge* é capaz de representar uma subdivisão planar (SP), ou seja, um complexo completo de dimensão 2; *a radial-edge* é capaz de representar uma subvisão espacial (SE), ou seja, um complexo completo de dimensão 3.

3. Metodologia de Subdivisões Espaciais

Somente mais recentemente, foi desenvolvido trabalho conceitual mais cuidadoso de definição de um modelo matemático para objetos heterogêneos e *non-manifold.*

Rossignac e O'Connor (1990), com o **SGC** (*Selective Geometric Complexes),* criaram um modelo para subdivisões do espaço. Cavalcanti (1992) ampliou esses estudos e sugeriu um processo de modelagem para construção de uma **SE** (Subdivisão Espacial).

Este capítulo apresentará, inicialmente, o conceito de subdivisões do espaço. Em seguida, será explicada a estrutura de dados *radial-edge*, criada por Weiler (1986). Finalmente, será dada uma visão da arquitetura do sistema para criação e manutenção de subdivisões espaciais desenvolvido por Cavalcanti (1992).

O capítulo não pretende esgotar os assuntos abordados. Informações mais detalhadas podem ser encontradas nos trabalhos de Cavalcanti (1992) e Weiler (1986). O objetivo, neste trabalho, é apenas apresentar o **Sistema de Subdivisões Espaciais** e documentar a estrutura de dados usada no protótipo de Visualização 3D de Unidades Geológicas.

3.1 Subdivisões do Espaço

Os **SGCs** (*Selective Geometric Complexes*) representam um objeto complexo através de duas etapas: subdivisão e seleção. Primeiro, um subconjunto do $\Re^n$ é subdividido em células que obedecem a determinadas condições.

As células de um complexo geométrico devem seguir os critérios enunciados abaixo.

- Os conjuntos de pontos de duas células quaisquer devem ser disjuntos.

- Um complexo de dimensão n pode ser representado por uma coleção de células cuja dimensão varie no intervalo de 0 a n. Assim, um complexo 3D pode ser representado por células de dimensão 0, 1, 2 e 3. Uma **célula de dimensão n** é um subconjunto limitado do espaço Euclidiano, homeomorfo a um disco aberto de dimensão n.

- Toda fronteira de uma célula do complexo geométrico é composta por células de dimensão menor. Devido a essa condição, um complexo é sempre fechado.

Após a etapa de subdivisão, as células do complexo correspondentes ao objeto são selecionadas (são tornadas **ativas**).

As informações de adjacência mantidas nos complexos geométricos determinam os relacionamentos de adjacência entre as células, mas são insuficientes para caracterizar, completamente, a topologia do complexo em relação ao espaço. A Figura 3-1 mostra complexos ambíguos cujos relacionamentos de adjacência são idênticos. Surge, então, a necessidade de introduzir o conceito de **complexo geométrico completo** [Cavalcanti et al. (1993)]. Em um complexo geométrico completo, a união de todas as células é todo o espaço.

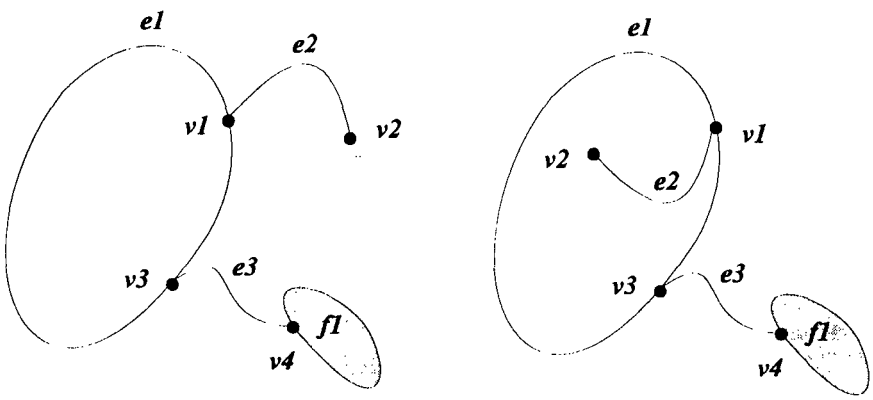


Figura 3-1 Complexos geométricos ambíguos.

Uma vez citados os conceitos de **complexo geométrico** e de **complexo geométrico completo**, torna-se possível utilizar a definição de Cavalcanti [Cavalcanti et al. (1993)] para **subdivisões planares e espaciais**.

"Uma $\mathbf{SE}_n$ é um complexo geométrico de dimensão n, completo e que possui apenas uma célula ilimitada (esta célula tem dimensão n). Uma $\mathbf{SE}_2$ é chamada de **subdivisão planar** ou **SP**. Uma SE_3 é chamada de **subdivisão espacial** ou **SE**."

Uma **SE** é formada por células de dimensão 3 (regiões), cujas fronteiras são subdivididas em células de dimensão 2 (faces), cujas fronteiras são subdivididas em células de dimensão 1 (arestas), cujas fronteiras são formadas por células de dimensão 0 (vértices).

3.2 Representações para Subdivisões do Espaço

Representações de sólidos são estruturas de dados criadas com a finalidade de descrever um sólido. Estruturas de dados topológicas não guardam somente dados geométricos (por exemplo, coordenadas de vértices), elas descrevem, também, a topologia do objeto (adjacências e incidências). O termo **topologia**, no contexto de Subdivisões Espaciais, normalmente, se refere às adjacências (vizinhanças) entre células.

A estrutura de dados ***radial-edge***, criada por Kevin Weiler (1986), é capaz de armazenar informações de adjacência de uma **SE**. O nome da estrutura se deve ao fato de ela possibilitar a ordenação radial de faces ao redor de uma aresta (Figura 3-2).

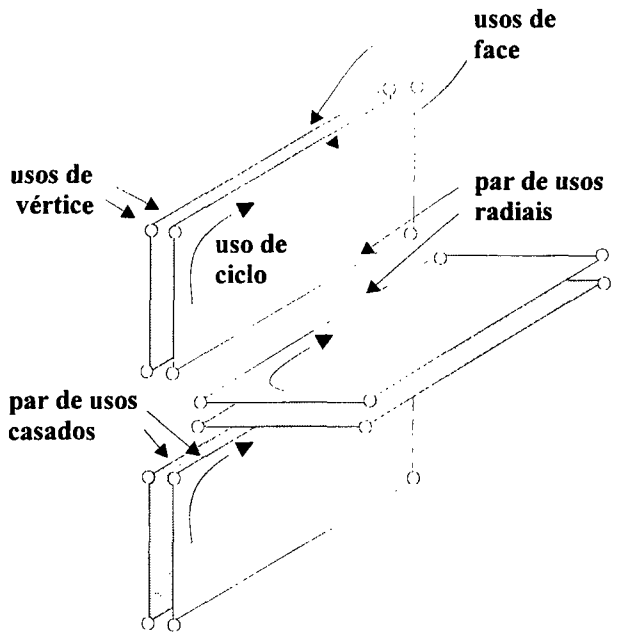


Figura 3-2 Ordenação radial das faces ao redor da aresta.

Em uma SE, é possível a obtenção de dezesseis tipos de relacionamento de adjacência. A tabela seguinte (Tabela 3-1) relaciona essas adjacências entre vértices, arestas, faces e regiões. Os relacionamentos são representados por duas letras: a primeira indica o elemento de referência; a segunda, o grupo de elementos adjacentes à referência. Na tabela, é usada a seguinte notação:

- < > indica grupo de adjacência que pode ser ordenado ciclicamente;
- { } indica grupo de adjacência sem ordenação;
- **letra minúscula** representa uma certa célula;
- **letra maiúscula** representa um conjunto de células;
- **expoente** indica a cardinalidade de um grupo.

Tabela 3-1 Relacionamentos de adjacência.

v {V}	conjunto de vértices adjacentes a *v*
v {A}	conjunto de arestas que incidem em *v*
v {F}	conjunto de faces que incidem em *v*
v {R}	conjunto de regiões que incidem em *v*
a {V}2	dois vértices que delimitam *a*
a {A}	conjunto de arestas adjacentes a *a*
a <F>	conjunto de faces que incidem em *a*
a <R>	conjunto de regiões que incidem em *a*
f <V>	conjunto de vértices que delimitam *f*
f <A>	conjunto de arestas que delimitam *f*
f <F>	conjunto de faces adjacentes a *f*
f {R}2	duas regiões que incidem em *f*
r {V}	conjunto de vértices que delimitam *r*
r {A}	conjunto de arestas que delimitam *r*
r {F}	conjunto de faces que delimitam *r*
r {R}	conjunto de regiões adjacentes a *r*

3.2.1 Elementos Topológicos

Em uma **SE**, um objeto é representado por vértices, arestas, faces e regiões, ou seja, por células de dimensão 0, 1, 2 e 3. Esses elementos são conhecidos, em modelagem, por **elementos topológicos.**

Como os elementos topológicos com cavidades possuem múltiplas fronteiras desconexas, para representá-los na estrutura, foram criados os elementos ciclo e casca.

Para armazenar a topologia de uma SE, a estrutura *radial-edge* utiliza ainda o conceito de uso de elemento topológico. São definidos quatro tipos de usos associados aos elementos vértice, aresta, ciclo e face.

A estrututura *radial-edge* organiza os elementos topológicos e os usos de elementos topológicos de maneira hierárquica, partindo dos elementos mais altos em dimensão (regiões) para os mais baixos (vértices), como mostra a Figura 3-3. Os elementos de nível mais baixo definem o contorno dos elementos de nível mais alto.

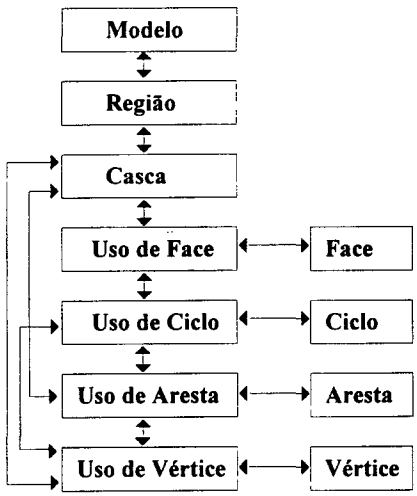


Figura 3-3 Hierarquia dos elementos na estrutura *radial-edge.*

Os elementos topológicos e os usos de elementos topológicos empregados na estrutura são descritos a seguir.

Um **vértice** (*vertex*), célula de dimensão 0, corresponde a um único ponto no espaço. Não existem dois vértices que compartilhem a mesma posição geométrica.

Uma **aresta** (*edge*), célula de dimensão 1, corresponde a um segmento de curva limitado por um vértice em cada extremo (os dois vértices não são, necessariamente, distintos). Geralmente, uma aresta faz parte da fronteira de uma ou mais faces. Um arame é uma aresta que não faz parte da fronteira de nenhuma face.

Um **ciclo** (*loop*) é um conjunto conexo e ordenado, composto, alternadamente, de arestas e vértices na fronteira de uma face. Um ciclo pode ser formado por um vértice isolado e, nesse caso, ele é chamado de ciclo pontual.

Uma **face** (*face*), célula de dimensão 2, corresponde a uma porção conexa e limitada de uma superfície no espaço. A fronteira de uma face é formada por um ou mais ciclos.

Uma **casca** (*shell*) é um conjunto conexo de faces e arames. Uma casca pode ser formada por um vértice isolado e, nesse caso, ela é chamada de casca pontual.

Uma **região** (*region*), célula de dimensão 3, é um conjunto conexo do espaço. A fronteira de uma região é formada por uma ou mais cascas.

Um **uso de face** é um dos lados de uma face. Quando a face está na fronteira de duas regiões, cada um dos seus usos pertence a cascas distintas, uma para cada região. Cada uso de face é limitado por um ou mais usos de ciclo. Cada uso de ciclo é limitado, alternadamente, por **usos de arestas** e **usos de vértices** (no caso de ciclo pontual). O uso de aresta corresponde

ao uso de uma aresta por um uso de ciclo. Geralmente, o uso de vértice corresponde ao uso de um vértice por um uso de aresta.

O armazenamento do uso de elementos topológicos visa a simplificar a travessia da estrutura de dados para a busca de adjacências entre os elementos topológicos. A rigor, com o armazenamento dos usos, o armazenamento dos elementos face, ciclo, aresta e vértice seria dispensável. No entanto, é conveniente representá-los, pois neles podem ser armazenados atributos e informações geométricas.

3.2.2 Estrutura de Dados

A Figura 3-4 detalha a organização da estrutura de dados. Para sua melhor compreensão, deve ser explicada a convenção adotada para os nomes dos ponteiros.

Os nomes terminados em **_next** e **_last** são ponteiros de listas duplamente encadeadas. Os nomes terminados em **_ptr** são ponteiros de elemento para elemento. Os mnemônicos usados para os nomes dos elementos topológicos e dos usos são: **m** para modelo (*model*), **r** para região (*region*), **s** para casca (*shell*), **f** para face (*face*), **l** para ciclo (*loop*), **e** para aresta (*edge*), **v** para vértice (*vertex*), **fu** para uso de face (*faceuse*), **lu** para uso de ciclo (*loopuse*), **eu** para uso de aresta (*edgeuse*) e **vu** para uso de vértice (*vertexuse*). Na estrutura de dados, **upptr** designa um ponteiro para um elemento mais alto na estrutura topológica e **downptr**, para um elemento mais baixo.

Todos os elementos da estrutura estão organizados em listas duplamente encadeadas. Isto é feito para tornar rápidas as manipulações de construção e consulta da estrutura. A **SE** é armazenada na estrutura ***Model***. Esta aponta para listas circulares, duplamente encadeadas de regiões, faces, arestas e vértices. Desse modo, torna-se rápida a busca de todos os elementos de um tipo específico.

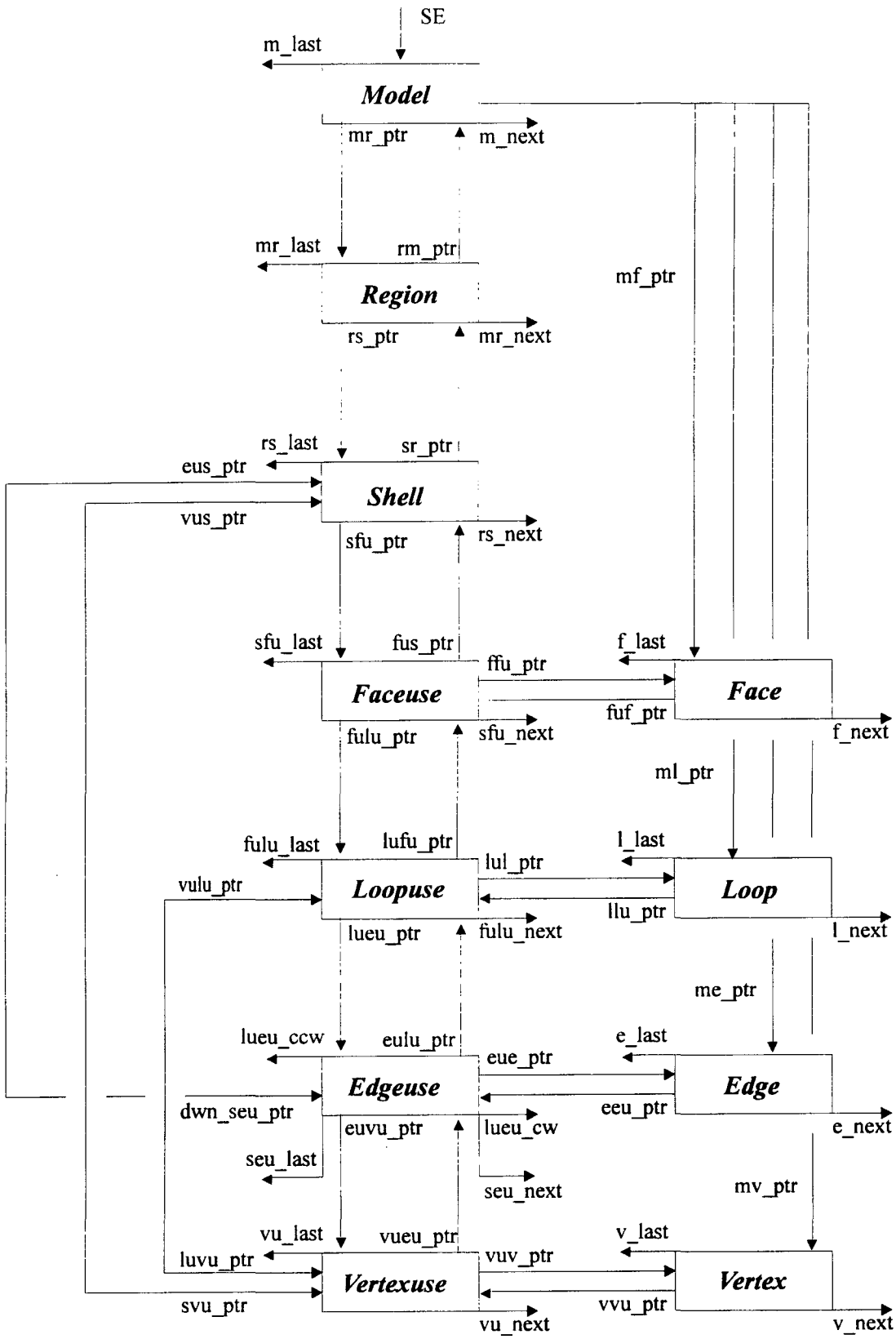


Figura 3-4 Organização da estrutura *radial-edge*.

A estrutura para **região** (*region*) possui ponteiro para seu modelo (**rm_ptr**) e para um lista de cascas (**rs_ptr**).

A estrutura para **casca** (*shell*) possui ponteiro para sua região (**sr_ptr**). Como uma casca pode ser formada por faces, arames ou por um vértice isolado, a estrutura casca pode apontar para um uso de face (**sfu_ptr**), uso de aresta (**dwn_seu_ptr**) ou uso de vértice (**svu_ptr**).

As estruturas para os elementos face (*face*), ciclo (*loop*), aresta (*edge*) e vértice (*vertex*) possuem ponteiros para um dos usos do elemento. Os demais usos de um elemento sempre podem ser encontrados, a partir do uso identificado, usando-se informações de adjacência.

A estrutura para **uso de face** (*faceuse*) possui ponteiro para sua casca (**fus_ptr**), para uma lista de usos de ciclo (**fulu_ptr**), para a sua face (**fuf_ptr**) e para o outro uso de face da mesma face (**fufu_mate_ptr**).

A estrutura para **uso de ciclo** (*loopuse*) possui ponteiro para seu uso de face (**lufu_ptr**), para seu ciclo (**lul_ptr**), para o uso de ciclo do outro uso de face da mesma face (**lulu_mate_ptr**). Como um ciclo pode ser formado por um conjunto de arestas e vértices ou por apenas um vértice isolado, a estrutura uso de ciclo pode apontar para um uso de aresta (**lueu_ptr**) ou um uso de vértice (**luvu_ptr**).

Como uma aresta pode ser um arame ou pode delimitar uma face, o campo **upptr** da estrutura uso de aresta pode apontar para uma casca (**eus_ptr**) ou para um uso de face (**eufu_ptr**). A estrutura **uso de aresta** (*edgeuse*) aponta para sua aresta (**eue_ptr**), para o outro uso de aresta na mesma face (**eueu_mate_ptr**), para o uso de vértice de seu vértice inicial

(**euvu_ptr**). Há ainda o ponteiro **eueu_radial_ptr** que aponta para o outro uso da mesma aresta no próximo uso de face em torno da aresta.

É importante notar que os ponteiros **eueu_radial_ptr** e **eueu_mate_ptr** fornecem meios de ordenar, radialmente, as faces que incidem em determinada aresta. Desse modo, é possível retratar condições *non-manifold* em uma aresta.

Como um vértice pode delimitar uma aresta ou formar um ciclo pontual ou uma casca pontual, o campo **upptr** da estrutura **uso de vértice** (*vertexuse*) pode apontar para um uso de aresta (**vueu_ptr**), para um ciclo (**vulu_ptr**) ou para uma casca (**vus_ptr**). A estrutura uso de vértice possui ainda ponteiro para seu vértice (**vvu_ptr**). Através do uso de vértice, é possível retratar condições *non-manifold* em um vértice.

3.2.3 Operadores Topológicos

Operadores topológicos são usados para criar e modificar a estrutura topológica de modelos geométricos. Os mais difundidos são os operadores de Euler.

Algumas características dos operadores de Euler podem ser citadas: através dos operadores, é mantida a consistência da estrutura topológica; todo operador de Euler possui um operador inverso, isto é, um operador que desfaz a ação original; uma camada topológica de alto nível pode ser definida através de uma seqüência de operadores de Euler.

Operadores de Euler convencionais não manipulam objetos *non-manifold*. Para esses objetos, foi criada uma extensão dos operadores de Euler. Weiler (1986) criou um conjunto de operadores, os **WOps**, para a manipulação da estrutura *radial-edge*. A lista (Tabela 3-2 e Tabela 3-3) desses operadores, com uma breve descrição de suas funções, é mostrada a seguir.

Tabela 3-2 Operadores que agem somente sobre as faces de uma SE.

mvfs	(make vertex, face and SP) cria uma SP com uma única face ilimitada.	**kvfs**	(kill vertex, face and SP)
mev	(make edge and vertex) cria uma aresta e um vértice em uma face.	**kev**	(kill edge and vertex)
mekr	(make edge, kill ring) cria uma aresta entre dois vértices, destruindo o ciclo pontual (*ring*) existente.	**kemr**	(kill edge, make ring)
mef	(make edge and face) cria uma aresta entre dois vértices pertencentes ao mesmo ciclo, dando origem a uma face.	**kef**	(kill edge and face)
mvr	(make vertex and ring) cria um ciclo pontual em uma face.	**kef**	(kill edge and face)

Tabela 3-3 Operadores para criar arames e adicionar faces ligadas a arestas ou arames específicos.

mmr	(*make SE and region*)	**km**	(*kill SE*)
	cria uma SE com uma única região ilimitada.		
msv	(*make shell and vertex*)	**kv**	(*kill vertex*)
	cria uma casca pontual em uma região.		
mev	(*make wire edge and vertex*)		
	cria um arame e um vértice em uma região.		
me	(*make wire edge*)	**ke**	(*kill edge*)
	cria um arame entre dois vértices de uma região.		
mf	(*make face*)	**kf**	(*kill face*)
	cria uma face, sendo que os vértices e arestas já existem na SE.		
esplit	(*split edges*)	**join**	(*join edges*)
	divide uma aresta (ou arame) em duas, criando um vértice entre elas.		

3.3 O Sistema de Subdivisões Espaciais

O **Sistema de Subdivisões Espaciais (SSE)** foi implementado em três camadas com funções bem definidas: uma camada é responsável pela topologia e ignora, completamente, problemas geométicos; outra é responsável pelo gerenciamento e manipulação de entidades geométricas e pela tradução de informações geométricas para informações topológicas; e a última camada é a de interação com o usuário.

Figura 3-5 Arquitetura do SSE.

O **SSE** foi criado para aplicações que criam sólidos de maneira não interativa. Um conjunto de retalhos de superfície é, de alguma maneira, gerado e é fornecido para o sistema, através de um arquivo. Os retalhos podem ser fornecidos em qualquer ordem e com interseção. Cada retalho a ser inserido determina um complexo geométrico completo de dimensão 3, com uma única face e uma única região, que deve ser compatibilizado com o complexo geométrico completo da SE já existente. Este procedimento, descrito no trabalho de Cavalcanti (1992), será revisto com mais detalhes no capítulo 5.

É possível armazenar subdivisões espaciais em meio permanente, utilizando-se o algoritmo de inversão. A SE é desfeita através da eliminação de todas as suas faces, arames e cascas pontuais, nesta ordem. Os WOps inversos dos utilizados na destruição da SE são colocados em uma pilha e, depois, desempilhados e armazenados em meio permanente. A execução dos WOps inversos em ordem inversa à da destruição recria a subdivisão espacial.

O SSE foi organizado de maneira a permitir a inclusão de novas geometrias para faces e arestas. A versão em uso neste trabalho trata faces planas e arestas retilíneas. Em sua tese, Cavalcanti (1992) identifica um grupo de funções geométricas necessárias para subdividir o espaço. Cada face e aresta é associada a funções próprias ao seu tipo de geometria. Assim, para que seja adotada uma nova geometria (geometria curva, por exemplo), basta que sejam implementadas novas funções geométricas capazes de tratar a nova geometria.

4. Exibição Gráfica de Informações

Comunicar, de forma clara, as informações contidas no modelo geológico e fazer que o usuário perceba o bloco diagrama como um objeto tridimensional são metas no desenvolvimento do protótipo.

Um intérprete pode desejar observar as formas e características intrínsecas (litologia, por exemplo) das unidades geológicas ou, ainda, observar somente as superfícies de contato entre as unidades. Como a estrutura é capaz de representar objetos formados por vários materiais, o bloco pode ser exibido como um conjunto de unidades geológicas e, como cada região é delimitada por cascas (por superfícies), o modelo geológico pode, também, ser exibido como um conjunto de horizontes.

A exibição gráfica das informações depende da estrutura de dados e de algoritmos de *rendering*. Nas próximas seções, são apresentadas algumas características da estrutura *radial-edge* que influenciam na visualização do modelo. Estas características e os recursos oferecidos pelo sistema gráfico utilizado (GL) determinaram a escolha dos algoritmos de *rendering* empregados no desenvolvimento do protótipo.

4.1 Características da Estrutura Radial-edge

Nesta seção, serão analisados dois pontos: a travessia do modelo para visualização e a exibição dos atributos do modelo.

4.1.1 Travessia do Modelo

O projeto de algoritmos de visualização de modelos armazenados na estrutura *radial-edge* é simplificado pela disponibilidade de faces, arestas e vértices do sólido. Os algoritmos para exibição do modelo podem se basear na travessia de faces (polígonos) ou de arestas.

O quadro (Tabela 4-1) abaixo mostra a relação existente entre os objetos geométricos, topológicos e geológicos que compõem o modelo do bloco diagrama.

Tabela 4-1 Relação entre os objetos geométricos, topológicos e geológicos.

Objetos geométricos	**Objetos topológicos**	**Objetos geológicos**
sólido	região (dimensão 3)	unidade geológica
superfície	face (dimensão 2)	horizonte
curva	aresta (dimensão 1)	sem significado

No protótipo, uma unidade geológica corresponde a uma região e um horizonte, a um conjunto de faces que fazem contato entre duas regiões. Como as unidades geológicas podem se estender por muitos metros tanto vertical quanto horizontalmente, são criadas, artificialmente, superfícies para fechamento do bloco diagrama, ou seja, são definidos os limites do bloco. A região fora desses limites corresponde à região externa no modelo topológico.

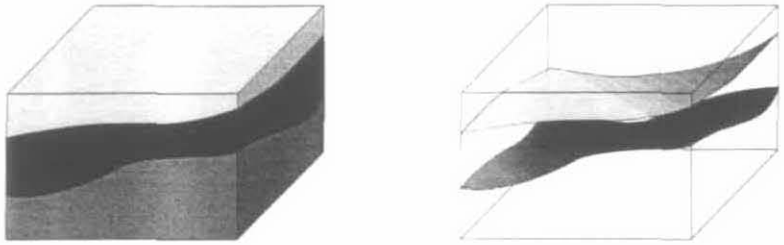
A visualização de unidades geológicas e de horizontes se baseia na exibição de faces (polígonos). A exibição do modelo como *wireframe*, por não ter significado geológico, só é utilizada, em casos especiais, quando se deseja rapidez de animação durante a manipulação.

Os algoritmos que percorrem a estrutura de dados para exibição, como pode ser visto a seguir, são bastante simples.

Para exibição das unidades geológicas de um bloco diagrama (das regiões do modelo), são percorridas todas as faces de todas as cascas da região externa (Figura 4-1(a) e Figura 4-2). Estas faces incluem eventuais faces de corte do modelo que são criadas em tempo de manipulação, conforme é explicado no próximo capítulo.

Para exibição dos horizontes (Figura 4-1(b) e Figura 4-3) de um bloco diagrama (faces do modelo), são percorridas todas as faces que não pertençam à região externa. O único trabalho adicional é verificar, para cada face, qual o uso de face voltado para o usuário.

Para exibição do modelo como *wireframe*, são percorridas todas as arestas do modelo.



(a) (b)

Figura 4-1 Exibição de unidades geológicas (a) e de horizontes (b).

4.1.2 Exibição de Atributos do Modelo

Todos os elementos topológicos e usos de elementos topológicos da estrutura *radial-edge* possuem um ponteiro para atributos, sendo que estes podem ser definidos de acordo com a necessidade da aplicação. No protótipo, as unidades geológicas são caracterizadas pelo tipo de rocha, ou seja, um material é associado a cada região da estrutura.

Esta possibilidade de atribuir características visuais a um determinado componente da estrutura é importante como forma de comunicação de informações do modelo.

4.2 Algoritmos de Rendering

Para que o usuário tenha sensação de volume, foram usados recursos de *rendering* tais como remoção de superfícies invisíveis e efeitos de iluminação. Neste trabalho, não se

pretende explicar os algoritmos de *rendering* utilizados, já que estes são oferecidos pela biblioteca gráfica.

4.2.1 Remoção de Superfícies Escondidas

Ao se usar uma estrutura baseada em aresta, como é o caso da *radial-edge*, deve-se levar em conta na escolha de um algoritmo de *rendering* que os polígonos são apresentados em qualquer ordem.

Algoritmos clássicos de remoção de superfícies escondidas, como os de *scan-line*, de *z-buffer* e de *ray-casting,* são aplicáveis na visualização de modelos armazenados na estrutura *radial-edge*. O algoritmo de *backface elimination* não é aplicável quando o objeto não é convexo ou quando há mais de um objeto em cena.

No protótipo foi utilizado o algoritmo de *z-buffer* oferecido pela biblioteca gráfica. O algoritmo de *z-buffer* não impõe limitações nem na organização da estrutura de dados nem na ordem de processamento dos polígonos. Este algoritmo, que se tornou um padrão de fato em computação gráfica, é de baixa complexidade, explora a arquitetura dos equipamentos computacionais disponíveis e, além disso, é adequado para implementação em *hardware*.

4.2.2 Efeitos de Iluminação

Efeitos de iluminação são usados para tratar objetos tridimensionais em um espaço bidimensional. Neste trabalho, o objetivo não é retratar, integralmente, a realidade, mas dar uma idéia do volume do objeto em estudo, através de sombreamento (diminuição de intensidade de cor).

Quando um objeto do mundo real é iluminado por alguma fonte de luz, um pouco dessa luz é absorvida pelo objeto e o resto é refletido. A luz refletida é interpretada pelo olho como forma e cor do objeto.

As características da fonte de luz determinam a direção, intensidade e cor da luz incidente. As características da geometria e do material do objeto determinam a direção, intensidade e cor da luz refletida.

Na implementação, são definidas as caracterísiticas da fonte de luz, a geometria do objeto é fornecida durante a travessia da estrutura e o material do objeto é dado pelo atributo associado a uma região.

A biblioteca gráfica GL oferece dois tipos de sombreamento: *flat shading*, que sombreia os polígonos uniformemente, e *Gouraud shading,* que varia a cor no interior dos polígonos, quando se deseja dar a estes uma aparência de superfície suave. No protótipo, foi empregado o modelo *flat*, por ser mais rápido e porque, em algumas situações (por exemplo, nas laterais do bloco diagrama), não é desejável suavizar a superfície.

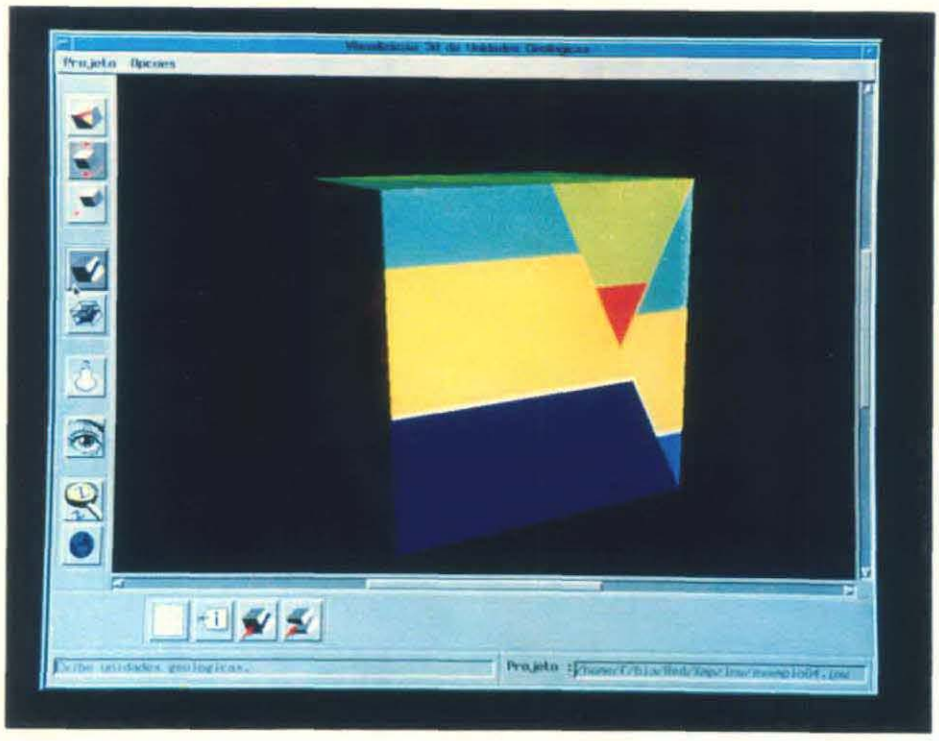
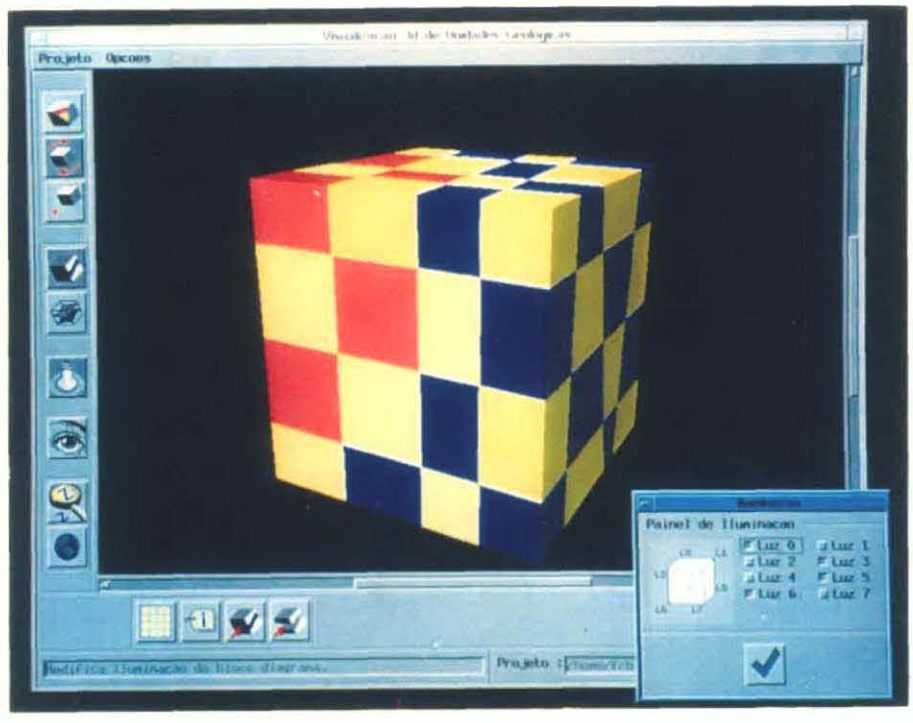
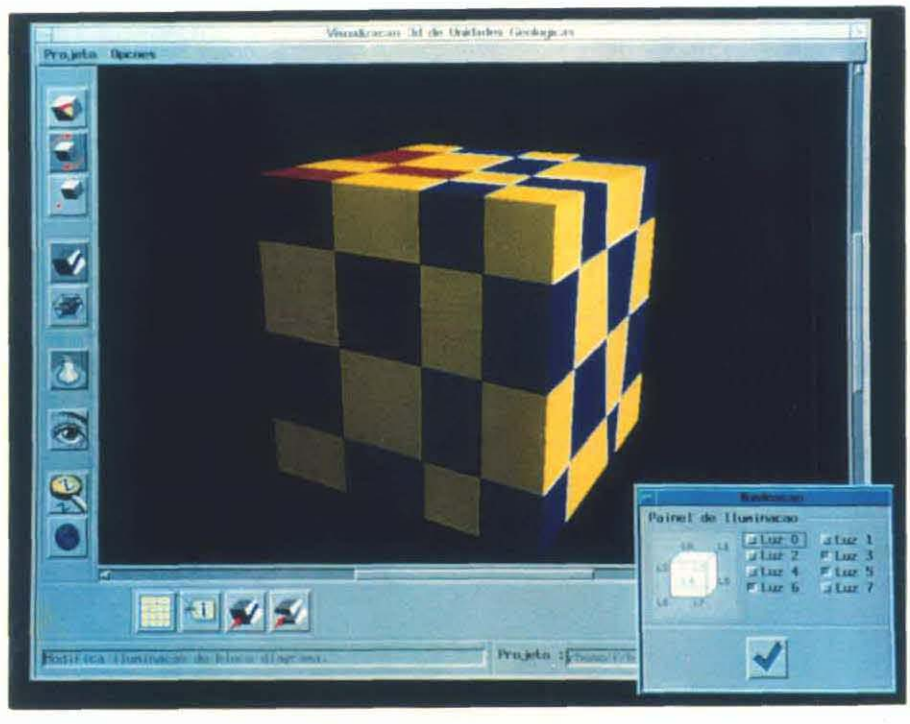


Figura 4-2 Exibição das regiões do modelo.

Figura 4-3 Exibição das faces do modelo que não pertencem à região externa.



(a)



(b)

Figura 4-4 Efeitos de iluminação: em **(a)**, a luz 0 está ligada; em **(b)**, está desligada.

5. Manipulação Interativa e Visualização de Objetos 3D

A escolha de modelos para manipulação é complexa pois envolve tanto procedimentos algébricos que controlam a projeção e o movimento como a interface com a visão 3D do usuário. Neste capítulo, é apresentada uma taxonomia dos diversos modelos de manipulação, criando-se subsídios para o desenvolvimento de novos tipos de controle.

Para que o problema da manipulação para visualização fique bem caracterizado, utiliza-se o conjunto de parâmetros de visualização do chamado **modelo de câmera**. Com base nestes parâmetros, são analisados os efeitos da movimentação de uma câmera na imagem de um objeto na tela.

Levando-se em conta a taxonomia proposta para os modelos de manipulação, são analisados alguns algoritmos existentes. Também são mostrados dois modelos simples de manipulação: um para rodar e mover um objeto com relação aos eixos da tela e outro para simular a movimentação da câmera dentro de um ambiente.

Propõe-se um modelo de manipulação, baseado na orientação corrente do objeto na tela, que é ideal para objetos que, a exemplo de blocos diagrama, se apresentam como uma caixa que envolve o domínio a ser simulado. A manipulação de visualização, utilizando este modelo para um objeto de forma genérica, pode ser feita através de sua caixa envolvente. Algumas avaliações sobre os três modelos de manipulação são feitas.

Propõe-se, também, uma estratégia de corte de objetos 3D, que envolve a manipulação do plano de corte e a metodologia para criação de subdivisões espaciais.

5.1 Parâmetros de Visualização em 3D

A abstração utilizada para a especificação dos parâmetros de visualização em 3D segue, quase que integralmente, um dos modelos definidos pelo *OpenGL* [Neider (1993)] e que aqui é referido como **modelo de câmera** (Figura 5-1). Neste modelo, uma região, com a forma de um tronco de pirâmide, denominada **frustum de visão**, limita a porção do espaço tridimensional que é vista através de uma área retangular associada a uma janela na tela de um computador.

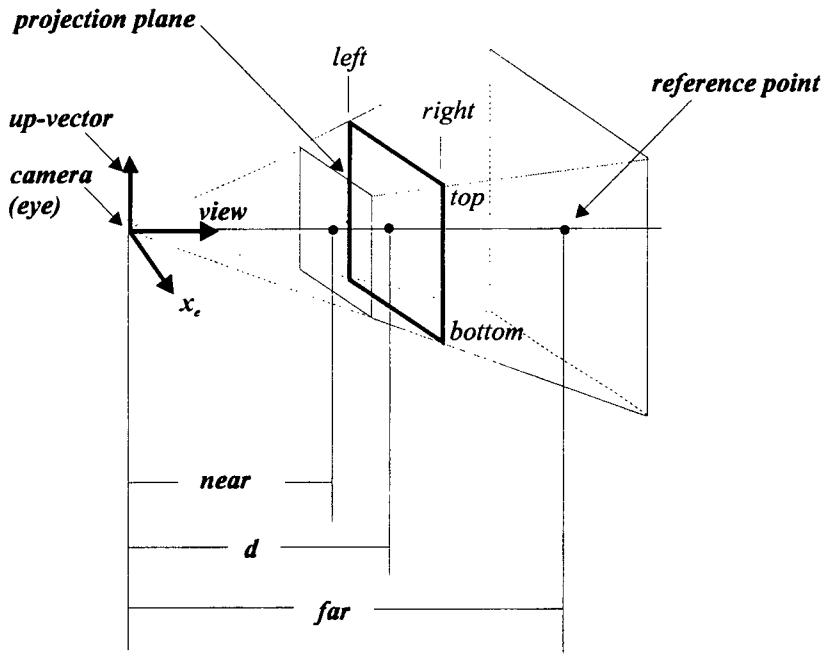


Figura 5-1 Modelo de câmera e frustum de visão.

Os parâmetros necessários para posicionar a câmera no espaço de modelagem são:

- posição da câmera (olho):

  (eye_x, eye_y, eye_z)

- posição do ponto de referência (um ponto no espaço de modelagem para onde a câmera mira):

  (ref_x, ref_y, ref_z)

- vetor de orientação vertical da câmera (*view up-vector* – ***vup***):

  (vup_x, vup_y, vup_z).

O vetor orientado da posição do olho para o ponto de referência é definido como **vetor de visão** (***view***). Os nove parâmetros mostrados acima definem o **sistema de coordenadas do olho**, que é um sistema de coordenadas que tem o ponto do olho como origem, eixo z_e orientado no sentido oposto ao vetor de visão, eixo y_e localizado no plano formado pelos vetores ***view*** e ***vup*** (voltado para o mesmo sentido de ***vup***). Desta forma, os vetores unitários que definem o sistema de coordenadas do olho podem ser calculados como:

$$\boldsymbol{view} = (ref_x, ref_y, ref_z) - (eye_x, eye_y, eye_z)$$

$$z_e = -\boldsymbol{view} / \|\boldsymbol{view}\|$$

$$x_e = (\boldsymbol{vup} \times z_e) / \|\boldsymbol{vup} \times z_e\|$$

$$y_e = z_e \times x_e.$$

O frustum de visão (Figura 5-1) pode ser definido de diversas maneiras e, neste trabalho, ele é determinado pelos limites da janela de visão (*left, right, bottom, top*) em um plano de projeção no sistema de coordenadas do olho e pelas distâncias dos planos de cerceamento anterior (*near*) e cerceamento posterior (*far*) ao olho, no sentido do vetor de visão. O plano de projeção corresponde à janela na tela do computador e é definido pela sua distância (*d*) ao olho, também no sentido do vetor de visão. Os planos de cerceamento anterior e posterior e o plano de projeção são perpendiculares ao eixo z_e e situam-se no seu lado

negativo. Deve-se observar que, enquanto os parâmetros da janela de visão são abscissas com valores reais negativos ou positivos, os parâmetros *far*, *near* e *d* são por definição distâncias positivas.

5.2 *Movimentos Básicos da Câmera*

O modelo de câmera proporciona uma maneira natural para se controlar a visualização de um ambiente ou de um objeto tridimensional. A identificação deste controle natural vai fornecer subsídios para formas mais eficientes do controle de visualização discutido mais tarde. Esta seção está baseada no trabalho de Wawrzynek (1991).

Para exemplificar o controle natural do modelo de câmera, uma seqüência de figuras, Figura 5-2 a Figura 5-8, mostra a manipulação de visualização de um objeto simples, um cubo, a partir de uma posição inicial da câmera, Figura 5-2, que mira o centro do cubo. Os exemplos ilustram movimentos básicos da câmera no modelo *Screen-based*, que será explicado nas próximas seções. O sistema de coordenadas do olho é, inicialmente, paralelo ao sistema de coordenadas do objeto. A superfície de uma esfera imaginária centrada no ponto de referência passando pela câmera é o lugar geométrico das posições da câmera que modificam a orientação do objeto na tela sem alterar a sua posição.

A Figura 5-3 mostra o resultado de uma rotação em torno de um eixo paralelo ao seu eixo vertical e passando pelo ponto de referência. A Figura 5-4 mostra o mesmo para um eixo paralelo ao eixo horizontal da câmera. Nestes dois casos, de uma forma geral, os únicos parâmetros de visualização atualizados são a posição da câmera e a orientação de seu eixo vertical (***vup***). A posição do ponto de referência fica inalterada. Observe que a rotação da câmera em um sentido resulta como imagem uma rotação do objeto no sentido oposto.

No caso da translação, Figura 5-5 e Figura 5-6, tanto a posição do olho quanto a posição do ponto de referência se alteram, e a orientação vertical da câmera fica inalterada. Note, nas imagens geradas, que a translação da câmera em um sentido resulta em uma translação do objeto no sentido oposto.

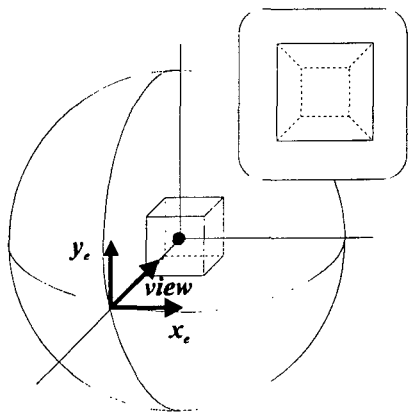


Figura 5-2 Um objeto simples visto pela posição inicial da câmera.

A Figura 5-7 mostra o resultado de uma rotação da câmera em torno de seu eixo axial, o que implica uma rotação no sentido oposto da imagem do objeto. A Figura 5-8 mostra o efeito da aproximação da câmera ao ponto de referência. Neste caso, a distância do plano de projeção à câmera é mantida inalterada, resultando em uma ampliação da imagem do objeto.

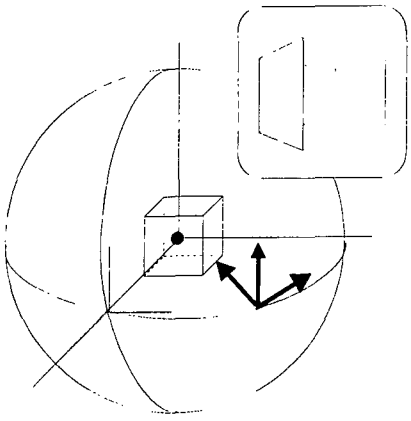


Figura 5-3

Rotação da câmera em torno de eixo vertical.

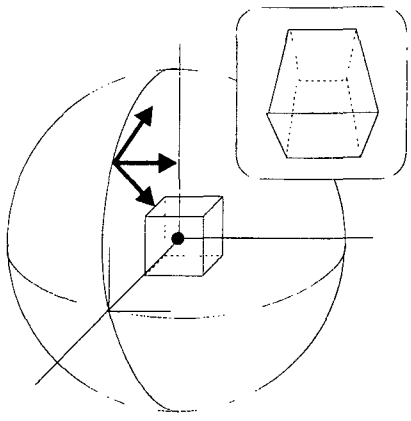


Figura 5-4

Rotação da câmera em torno de eixo horizontal.

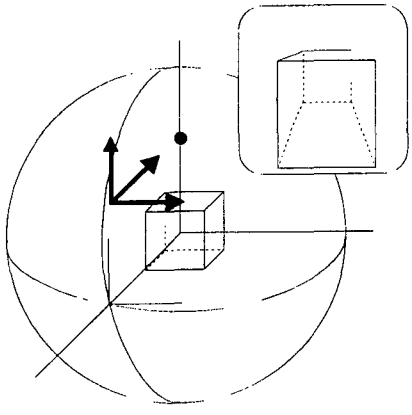


Figura 5-5

Translação vertical da câmera.

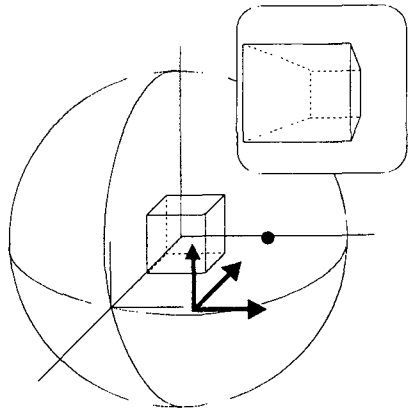


Figura 5-6

Translação horizontal da câmera.

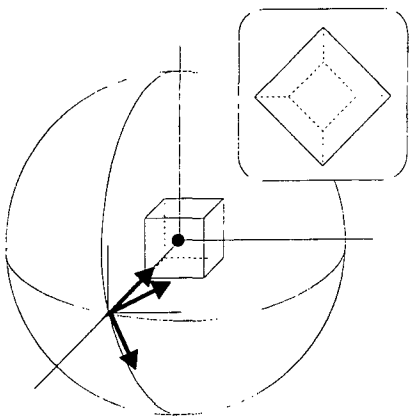


Figura 5-7

Rotação axial da câmera.

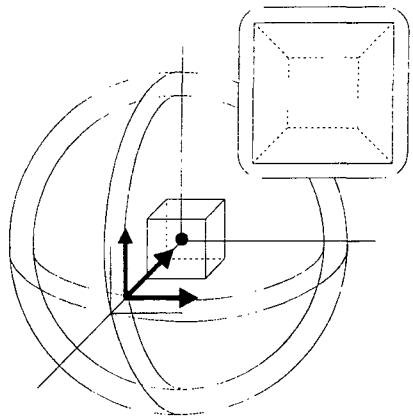


Figura 5-8

Translação radial da câmera.

5.3 Uma Taxonomia para o Controle de Visualização em 3D

A identificação dos movimentos básicos da câmera, com os seus efeitos na imagem do ambiente sendo visualizado, fornece subsídios para a especificação dos tipos de controles básicos de visualização em três dimensões. A especificação do controle de visualização depende do tipo de referencial de movimento, do tipo de movimento desejado e do ajuste de perspectiva e planos de cerceamento. Isto é mostrado na Figura 5-9.

Em alguns tipos de aplicação, é mais natural a rotação e a translação do objeto de modelagem em relação aos eixos da tela, adotando-se o ponto de referência como o centro de rotação. A manipulação deste tipo é denominada de *Screen-based.* Em outras aplicações, se deseja passear dentro de um ambiente criado no computador, sendo os movimentos sempre nas direções dos eixos da câmera e as rotações sempre centradas na posição da câmera. Esta é a manipulação que tem o referencial de movimento baseado na câmera e é dita do tipo *Walk-*

through. Em outros casos, ainda, é interessante a manipulação de movimentos baseada no sistema de eixos do objeto sendo visualizado e o controle é do tipo *Object-based*.

Ortogonalmente ao tipo de referencial de movimento, existe o tipo de movimento desejado: rotação (*Rotation*) ou translação (*Translation*). E, independentemente do tipo de referencial e do tipo de movimento, ainda é possível controlar a distância da câmera ao ponto de referência (*Eye Distance*) e as distâncias dos planos de cerceamento anterior e posterior (*Clipping Planes*) e do plano de projeção ao olho.

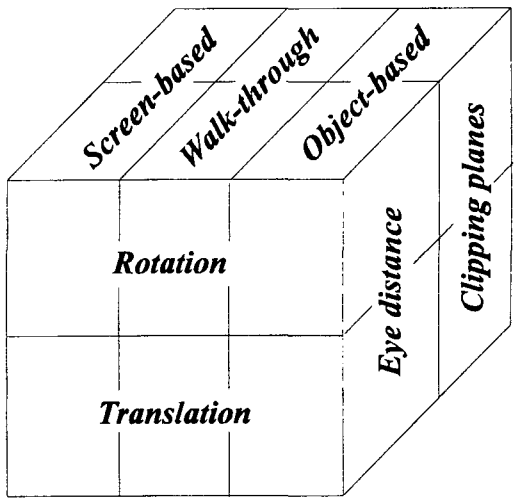


Figura 5-9 Uma taxonomia para o controle de parâmetros de visualização.

Além dos tipos de controle mostrados na Figura 5-9, existe o controle dos limites laterais do volume de visão ou da janela de visão (*left, right, bottom, top*). Este tipo de controle envolve uma manipulação de escala da imagem do objeto, um *pan* da imagem ou um *zoom* em uma região do ambiente. Por ser essencialmente bidimensional, este tipo de controle não foi considerado nesta taxonomia.

Diversas técnicas existentes para manipulação interativa de visualização de objetos 3D podem ser classificadas segundo a taxonomia proposta. Nesta seção, alguns modelos existentes são analisados. Nas seções seguintes, são propostos modelos básicos de manipulação.

A **esfera virtual** [Chen et al. (1988)] simula, com o uso de *mouse*, o mecanismo de um *trackball* capaz de girar livremente em torno de qualquer eixo do espaço 3D. O objeto é mostrado, na tela, dentro de um círculo que representa a esfera virtual. Ao se rolar a esfera, é girado também o objeto. Para definir a rotação, o usuário deve fornecer dois pontos. Um movimento dentro do círculo define uma direção tangencial à rotação. Um movimento fora do círculo define uma rotação em torno do eixo z_e. A esfera virtual é um exemplo de manipulação *Screen-based*, pois é feita em relação aos eixos da tela e adota o centro da esfera como ponto de referência e centro de rotação.

O ***arcball*** [Shoemake (1992)] é outra técnica para rotação de objetos 3D através do uso de *mouse*. Uma esfera envolve o objeto mostrado na tela. Para girar o objeto, o usuário define pontos na projeção da esfera. Como *feedback*, um arco é desenhado para mostrar o efeito do arrasto (*drag*). No *arcball*, utilizando-se fundamentos matemáticos, é feito um cuidadoso mapeamento entre o deslocamento do *mouse* e a rotação. Evitando-se a histerese (mudança de posição de um objeto quando o *mouse* retorna à sua posição inicial), é mais fácil desfazer uma seqüência de rotações. A rotação no *arcball* pode ser livre ou restrita a eixos especificados pelo usuário. Esses eixos podem ser selecionados nos sistemas de coordenadas do objeto, do mundo ou do olho, podem ser normais a arestas ou superfícies ou podem ser eixos de articulação do modelo. Observa-se, portanto, que o *arcball* permite os controles de rotação *Screen-based* e *Object-based*. Um aspecto negativo do *arcball* é a razão entre o movimento do *mouse* e o movimento do objeto, que é denominada *C/D ratio*. O formalismo matemático que

permite que rotações sejam comutativas (sem histerese) impõe que o objeto rode duas vezes mais que o cursor na esfera. Isto retira do usuário a sensação de "pegar e mover" (*pick and drag*) o objeto.

O mecanismo de manipulação direta de objetos 3D com o uso de dispositivos de entrada 2D apresentado por Emmerick (1990) permite a definição de parâmetros de três tipos de transformação 3D (rotação, translação e escala). A manipulação de movimentos é baseada no sistema de eixos do objeto (*Object-based*). São definidos sete pontos virtuais de controle no sistema de coordenadas local do objeto: um ponto na origem do sistema e seis nos eixos (um em cada semi-eixo). O usuário deve selecionar (*pick*) um ponto de controle e arrastá-lo (*drag*). O ponto de controle escolhido, a direção e a orientação do movimento do *mouse* determinam os parâmetros da transformação. Para definir translação, o ponto central de controle é arrastado paralelamente a um dos eixos. Para mudança de escala, o ponto de controle é selecionado em um dos eixos e arrastado paralelamente a este eixo. Para rotação do objeto em torno de um eixo, o ponto de controle é escolhido em um dos outros dois eixos e arrastado em direção paralela ao terceiro.

5.4 Modelos Básicos de Manipulação

Nesta seção são propostos modelos simples do tipo *Screen-based* e *Walk-through*, baseados nos movimentos básicos de câmera apresentados na seção 5.2. A manipulação de distância da câmera ao ponto de referência (*Eye Distance*) e a manipulação de distância dos planos de cerceamento anterior e posterior (*Clipping Planes*) e de projeção ao olho são comuns a todos estes tipos de controle e, por serem simples manipulações unidimensionais de distâncias, não são abordadas neste trabalho.

Para implementar os modelos básicos de manipulação, três operadores geométricos são utilizados: um para translação e dois para rotação. O operador *Translate*(t_x, t_y, t_z, p_x, p_y, p_z) transforma um ponto dado $\boldsymbol{p}$ segundo os parâmetros de translação t_x, t_y e t_z fornecidos. O primeiro operador de rotação, *RotateAxisOrg* (*angle*, r_x, r_y, r_z, p_x, p_y, p_z), transforma um ponto dado girando-o em torno de um eixo $\boldsymbol{r}$ arbitrário passando pela origem. O outro operador de rotação, *RotateAxisPnt*(*angle*, r_x, r_y, r_z, c_x, c_y, c_z, p_x, p_y, p_z), é semelhante ao primeiro, exceto que a rotação se dá em torno de um eixo passando por um outro ponto $\boldsymbol{c}$ dado. O segundo operador de rotação é implementado em função dos outros dois operadores da seguinte forma:

Translate($-t_x$, $-t_y$, $-t_z$, p_x, p_y, p_z)

RotateAxisOrg(*angle*, r_x, r_y, r_z, p_x, p_y, p_z)

Translate(c_x, c_y, c_z, p_x, p_y, p_z)

Os parâmetros de visualização que são atualizados pelas transformações geométricas são a posição do olho (câmera), a posição do ponto de referência e o vetor de orientação vertical da câmera. Conforme foi mostrado anteriormente, estes parâmetros definem os vetores unitários da base de coordenadas do olho em relação ao sistema de coordenadas de modelagem: $\boldsymbol{x_e}$, $\boldsymbol{y_e}$, $\boldsymbol{z_e}$. Maiores informações sobre as transformações geométricas necessárias para a exibição de objetos 3D na tela são descritas no apêndice B .

Na apresentação dos modelos básicos de manipulação, procura-se sempre que possível explorar o *mouse* como um dispositivo bidimensional de controle. Para tanto, considera-se que o movimento do *mouse* ocorre na janela de visão como se a tela fosse o próprio plano de projeção.

Na manipulação de rotação, o movimento do *mouse* registrado pelo sistema de interface em pixels (m_x, m_y) é normalizado em relação ao tamanho da janela de visão através de:

$$\delta_x = m_x / hsize$$

$$\delta_y = m_y / vsize,$$

onde *hsize* e *vsize* são os tamanhos horizontal e vertical da janela em pixels. Esta transformação define o vetor de movimento do *mouse* normalizado para rotação

$$\delta_m = (\delta_x, \delta_y).$$

Na manipulação de translação, o movimento do *mouse* é normalizado em relação aos tamanhos laterais da janela de visão no plano de projeção, isto é:

$$\Delta_x = \delta_x (right - left)$$

$$\Delta_y = \delta_y (top - bottom).$$

Esta transformação define um vetor de movimento do *mouse* normalizado para translação

$$\Delta_m = (\Delta_x, \Delta_y).$$

5.4.1 Modelo *Screen-based* Básico

Este modelo manipula a posição da imagem de um objeto na tela em relação a eixos que têm direções paralelas à própria tela. As direções da tela são sempre conhecidas no espaço de modelagem e correspondem às direções do sistemas de coordenadas do olho, visto que o plano de projeção é sempre paralelo ao plano $x_e y_e$ do olho. Portanto, as direções de rotação e translação do objeto neste modelo são as direções horizontal, vertical e axial da câmera.

É natural associar o movimento horizontal do *mouse* na tela a uma rotação em torno do eixo vertical da câmera e o movimento vertical com uma rotação em torno do eixo horizontal da câmera. Além disso, um movimento do *mouse* para a esquerda deve resultar em uma rotação do objeto na tela no mesmo sentido e um movimento para a direita deve resultar na rotação do objeto para a direita. O análogo pode ser dito para movimentos do *mouse* na direção vertical. Entretanto, conforme foi observado anteriormente, uma rotação da câmera em um sentido sempre resulta na rotação no sentido inverso da imagem do objeto na tela. Portanto, para se obter a rotação desejada, deve-se sempre rodar no sentido inverso ao movimento do *mouse*.

Para se explorar o *mouse* como um dispositivo de controle bidimensional, pode-se compor o seu movimento horizontal e vertical em um único vetor, cuja direção perpendicular no plano da tela define o eixo de rotação. Assim, considerando um movimento normalizado do *mouse* δ_m, o eixo de rotação r corresponderá ao vetor $(-\delta_y, \delta_x)$ na tela. A transformação deste vetor para o plano de projeção definido no espaço do objeto é trivialmente dada por:

$$r = \delta_x y_e - \delta_y x_e.$$

Na rotação em torno de um eixo da tela no modelo *Screen-based* básico, o centro de rotação é o ponto de referência (*reference pt.* - Figura 5-1).

Uma maneira simples de se definir o ângulo de rotação é considerar que um movimento do *mouse* de um extremo a outro da janela na horizontal ou na vertical corresponde a uma rotação de 180° ou seja

$$angle = -180^{\circ}\sqrt{\delta_x^2 + \delta_y^2}$$

Nota-se que este cálculo é feito entre *callbacks* do sistema de interface para eventos de movimento de *mouse*. Assim sendo, o ângulo é computado para pequenos incrementos da trajetória do *mouse* que pode ser uma curva qualquer. Ou seja, a direção de rotação e o valor do ângulo são valores instantâneos. O resultado final é uma integral do movimento.

Uma vez calculado o ângulo de rotação, a atualização dos parâmetros de visualização consiste apenas em rodar a posição da câmera (olho) em torno do eixo de rotação passando pelo ponto de referência e em rodar o vetor da orientação vertical da câmera em torno do eixo de rotação. Isto é mostrado abaixo:

RotateAxisPnt(angle, r_x, r_y, r_z, ref_x, ref_y, ref_z, eye_x, eye_y, eye_z*)*

RotateAxisOrg(angle, r_x, r_y, r_z, vup_x, vup_y, vup_z*).*

A rotação em torno do eixo axial da câmera pode ser associada à uma rotação do *mouse* em torno do centro da tela. Isto define o ângulo de rotação (Figura 5-10) utilizando um dispositivo bidimensional de controle. O eixo de rotação é o eixo z_e da câmera. E os únicos parâmetros de visualização a serem atualizados são as componentes do vetor da orientação vertical da câmera, tal como mostrado abaixo:

RotateAxisOrg(-angle, z_{ex}, z_{ey}, z_{ez}, vup_x, vup_y, vup_z*).*

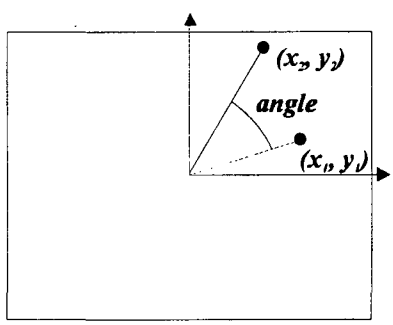


Figura 5-10 Cálculo do ângulo de rotação em torno do eixo axial da câmera.

Quanto ao controle de translação no modelo *Screen-based*, uma análise semelhante à que foi feita para o controle de rotação pode ser feita. Neste caso, o vetor de movimento do *mouse* na janela é transformado para movimentos no plano de projeção no espaço de coordenadas do olho. O controle de translação pode ser dividido em translações em direções paralelas ao plano de projeção e translações perpendiculares a este plano.

No primeiro caso, o vetor de translação t fica definido em função do vetor normalizado de movimento do *mouse* Δ_m na tela. Este vetor é transformado para o plano de projeção definido no espaço do objeto por:

$t = \Delta_x \; x_e + \Delta_y \; y_e$.

Os parâmetros de visualização a serem atualizados são as posições da câmera e do ponto de referência, conforme indicado abaixo, onde a troca de sinal dos parâmetros de translação é feita para transladar o objeto no sentido do movimento do *mouse*:

Translate($-t_x$, $-t_y$, $-t_z$, eye_x, eye_y, eye_z)

Translate($-t_x$, $-t_y$, $-t_z$, ref_x, ref_y, ref_z).

Com este este procedimento o objeto acompanha o movimento do *mouse* na tela.

Finalmente, a translação na direção perpendicular à tela se dá na direção do eixo z_e da câmera e pode ser definida em função do tamanho Δ do vetor Δ_m do movimento do *mouse* no plano de projeção (usando a tela como um mero potenciômetro):

$t = \Delta z_e$

Translate($-t_x$, $-t_y$, $-t_z$, eye_x, eye_y, eye_z)

Translate($-t_x$, $-t_y$, $-t_z$, ref_x, ref_y, ref_z).

5.4.2 Modelo *Walk-through* Básico

Neste modelo, os movimentos sempre se dão nas direções dos eixos da câmera e as rotações são sempre centradas na posição da câmera. O controle de manipulação neste modelo é bastante semelhante ao controle no modelo *Screen-based*, apenas que os movimentos do *mouse* devem indicar a direção de movimento e rotação da câmera e, portanto, o efeito na imagem do ambiente sendo visualizado na tela deve ser o inverso do que é obtido no modelo anterior.

Nas rotações da câmera em torno de um eixo paralelo ao plano de projeção (plano da tela), este efeito inverso é alcançado quando se roda o ponto de referência em torno do olho. Assim, para um movimento do *mouse* normalizado, fazem-se as seguintes operações geométricas para atualizar os parâmetros de visualização:

$\boldsymbol{r} = \delta_x\, y_e - \delta_y\, x_e$.

RotateAxisPnt(angle, r_x, r_y, r_z, eye_x, eye_y, eye_z, ref_x, ref_y, ref_z*)*

RotateAxisOrg(angle, r_x, r_y, r_z, vup_x, vup_y, vup_z*)*.

A rotação em torno de um eixo perpendicular à tela é feita da mesma forma que no modelo *Screen-based*, apenas que o sentido do ângulo de rotação deve ser invertido.

Para determinar o vetor de translação, o movimento do *mouse* é traduzido para o espaço de modelagem, com o mesmo sentido. Isto é mostrado em seguida, primeiro, para as translações em direções no plano de projeção:

$t = \Delta_x\ x_e + \Delta_y\ y_e$.

Translate(t_x, t_y, t_z, eye_x, eye_y, eye_z*)*

Translate(t_x, t_y, t_z, ref_x, ref_y, ref_z*)*.

E, finalmente, para translações perpendiculares ao plano de projeção, tem-se:

$t = \Delta z_e$

Translate(t_x, t_y, t_z, eye_x, eye_y, eye_z)

Translate(t_x, t_y, t_z, ref_x, ref_y, ref_z).

5.5 Manipulação Natural de Objetos com Caixa Envolvente

Nesta seção, é proposto um modelo de manipulação baseado no objeto (*Object-based*) que é ideal para objetos que apresentam um caixa envolvente natural. A motivação para a definição deste tipo de manipulação está em aplicações na área de Geologia, onde o objeto a ser manipulado é um conjunto de unidades geológicas, chamada de bloco diagrama, que corresponde ao domínio do meio rochoso que está sendo modelado. A fronteira destes objetos corresponde praticamente à caixa envolvente dos seus pontos. Entretando, o modelo proposto se aplica para qualquer tipo de objeto, sendo necessária apenas a visualização de sua caixa envolvente.

O modelo é bastante intuitivo e está baseado na idéia que para girar um objeto em torno de um dos seus eixo principais é preciso segurar duas arestas opostas paralelas a este eixo e dar a rotação desejada. No caso da translação na direção de um eixo principal, é preciso empurrar ou segurar uma face perpendicular ao eixo e dar o movimento desejado.

O problema básico está em associar estas idéias com uma manipulação interativa de um *mouse* em um dispositivo bidimensional. Uma possível solução, e de fácil implementação, é proposta a seguir. Parte-se do princípio que a caixa envolvente do objeto está desenhada na tela.

Para girar o objeto em torno de um dos eixos da caixa envolvente, passando pelo centro da caixa, o usuário pressiona o botão do *mouse* (*pick*) em uma das arestas da caixa na direção do eixo de giro e arrasta (*drag*) esta aresta no sentido da rotação desejada.

Para implementar esta metáfora de manipulação, imaginou-se que cada aresta contém uma direção $\boldsymbol{s}$ tangente de giro que está contida em um plano perpendicular à aresta. Isto é ilustrado na Figura 5-11. Estas direções são tais que, nos seus sentidos positivos, fazem um giro, que segue a regra da mão direita, em torno do eixo principal paralelo à aresta correspondente. Neste trabalho, optou-se por direções tangentes que fazem ângulos de 45° com os planos adjacentes das arestas.

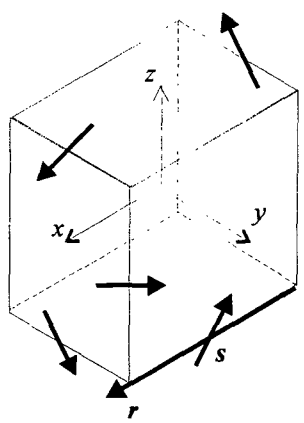


Figura 5-11 Arestas da caixa envolvente e as suas direções tangentes.

Movimentos do *mouse,* na tela, na direção da tangente $\boldsymbol{s}$ de uma aresta, projetada na tela, devem resultar em um máximo de rotação possível. Por outro lado, movimentos na direção perpendicular não devem resultar em rotação alguma. Isto é facilmente verificado, calculando-se o produto interno entre o vetor de movimento de *mouse* $\delta_{\boldsymbol{m}}$ e o vetor $\boldsymbol{s}$ projetado. Este produto interno é diretamente proporcional ao ângulo de giro.

Considerando $s_e = (s_{ex}, s_{ey}, s_{ez})$ o vetor s transformado para o espaço de coordenadas do olho (incluindo o efeito de perspectiva), o vetor unitário na direção da projeção deste vetor no plano da tela é dado por:

$v = (v_x, v_y)$.

$$v_x = s_{ex} / \sqrt{s_{ex}^2 + s_{ey}^2}$$

$$v_y = s_{ey} / \sqrt{s_{ex}^2 + s_{ey}^2}$$

O ângulo de rotação pode ser calculado em função do produto interno entre δ_m e v através da expressão:

$angle = -180° \ (\delta_x v_x + \delta_y v_y)$.

Nota-se que a rotação é ajustada para que um movimento do *mouse* de um extremo ao seu oposto na tela resulte em uma rotação total de 180°, a exemplo do que foi feito na 5.4.1. A troca de sinal é feita para girar o objeto no sentido do movimento do *mouse*. Define-se o vetor r como o eixo de giro na direção da aresta selecionada (Figura 5-11) e c como o centro da caixa envolvente. Dessa forma, têm-se as transformações geométricas que as posições do olho e do ponto de referência e a direção vertical da câmera devem sofrer para implementar as rotações neste modelo:

$RotateAxisPnt(angle, r_x, r_y, r_z, c_x, c_y, c_z, eye_x, eye_y, eye_z)$

$RotateAxisPnt(angle, r_x, r_y, r_z, c_x, c_y, c_z, ref_x, ref_y, ref_z)$

$RotateAxisOrg(angle, r_x, r_y, r_z, vup_x, vup_y, vup_z)$.

No caso da manipulação do objeto por translação na direção de um dos eixos principais da caixa envolvente, o usuário deve pressionar o botão do *mouse* em uma face

normal ao eixo e arrastar o objeto no sentido do movimento desejado. Desta forma a direção de movimento está indicada pela normal ***n*** da face selecionada (Figura 5-12).

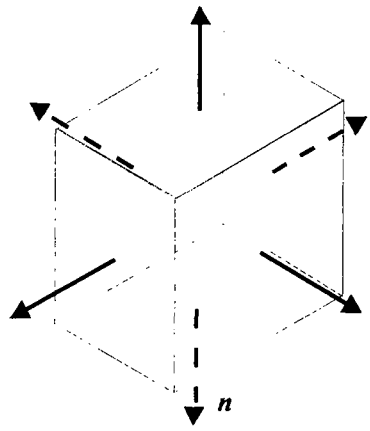


Figura 5-12 Normais das faces da caixa envolvente.

Considerando $\boldsymbol{n_e} = (n_{ex}, n_{ey}, n_{ez})$ o vetor ***n*** transformado para o espaço de coordenadas do olho (incluindo o efeito de perspectiva), o vetor unitário na direção da projeção deste vetor no plano da tela é dado por:

$$\boldsymbol{u} = (u_x, u_y).$$

$$u_x = n_{ex} / \sqrt{n_{ex}^2 + n_{ey}^2}$$

$$u_y = n_{ey} / \sqrt{n_{ex}^2 + n_{ey}^2}$$

E o vetor de translação pode ser calculado em função do produto interno entre Δ_m e u através da expressão:

$$t = (\Delta_x u_x + \Delta_y u_y)\, n.$$

Isto resulta nas seguintes transformações geométricas que implementam a manipulação de translação no modelo proposto:

Translate($-t_x$, $-t_y$, $-t_z$, eye_x, eye_y, eye_z)

Translate($-t_x$, $-t_y$, $-t_z$, ref_x, ref_y, ref_z).

A troca de sinal dos parâmetros de translação é feita para transladar o objeto no sentido do movimento do *mouse*.

5.6 Resultados da Taxonomia

Do estudo de vários modelos de manipulação resultou a taxonomia e a organização da álgebra das transformações propostas. Sobre a taxonomia, pode-se afirmar que ela permitiu um melhor entendimento dos diversos métodos existentes na literatura e possibilitou o desenvolvimento dos modelos básicos de manipulação que são descritos neste trabalho. Embora tenham sido efetuados poucos testes, nas avaliações preliminares feitas com manipulações de um objeto representado na tela, os modelos *Screen-based* e *Object-based* se mostraram mais eficientes, quando a manipulação é feita através do *mouse*. Quando a manipulação é feita através do teclado, o modelo *Screen-based* se mostra superior aos demais. O modelo *Screen-based* é melhor em uma movimentação livre e o *Object-based,* em um ajuste mais preciso da posição do objeto na tela. O modelo *Walk-through* parece ser mais adequado quando vários objetos compõem o ambiente.

Para trabalhos futuros espera-se, com base na taxonomia e na metodologia proposta aqui, avaliar cada uma das manipulações apresentadas junto a grupos de usuários.

5.7 Estratégia de Corte do Modelo

Uma estratégia para efetuar a seção no bloco diagrama envolve a escolha de um modo para manipulação do plano de corte e um procedimento para determinação das regiões na estrutura de dados (ou das unidades geológicas no bloco diagrama) cortadas pelo plano.

Aparentemente, a maneira mais intuitiva para a obtenção de um corte é indicar em que posição um plano, que pode ser livremente movimentado, deve seccionar o bloco diagrama. Um problema, no entanto, surge deste tipo de controle: o corte raramente é feito paralelamente ao plano de projeção. Assim, torna-se necessária a existência, na tela, de uma área para a exibição da seção, o que dificulta o estabelecimento de um vínculo entre a manipulação do plano e o resultado do corte.

Uma idéia semelhante à apresentada por Osborn e Agogino (1992), que não necessita uma área auxiliar para exibição do resultado do corte, foi adotada no protótipo. Para seccionar o bloco, optou-se por manter o plano de corte sempre paralelo à tela. O objeto é posicionado através de rotação ou translação. O plano de corte pode ser movido para frente e para trás, através do *mouse*, para dar a impressão ao usuário de estar fatiando (*slicing*) o bloco.

O ponto central da estratégia adotada é sugerir ao usuário uma sensação "sólido-volumétrica" para o bloco diagrama. Embora o bloco diagrama seja representado pela estrutura de dados como um conjunto de regiões (de sólidos), a sua exibição na tela é sempre feita desenhando-se as superfícies que delimitam as regiões do bloco. A estratégia sob teste atinge o seu objetivo, ao criar, automaticamente e em tempo real de execução, as superfícies que delimitam as regiões cortadas. Ao serem exibidas estas superfícies junto com as demais que não foram cerceadas, tem-se a sensação de ser obtido sempre um sólido como resultado do corte.

Na implementação, o plano de cerceamento anterior é utilizado como plano de corte. Os movimentos do plano para frente ou para trás equivalem a, respectivamente, diminuir ou aumentar a distância do plano de cerceamento anterior à câmera (Figura 5-13), sendo mantida a distância do plano de projeção à câmera.

O algoritmo de cerceamento (*clipping*) da biblioteca gráfica (Figura 5-13), que verifica quais partes do objeto estão dentro do frustum de visão, é aproveitado pelo procedimento de corte para eliminar a visão de porções do objeto que estão entre o plano de cerceamento anterior e a câmera. A próxima etapa é, então, substituir a parte tornada invisível pelo que passará a ser denominado **modelo da seção**.

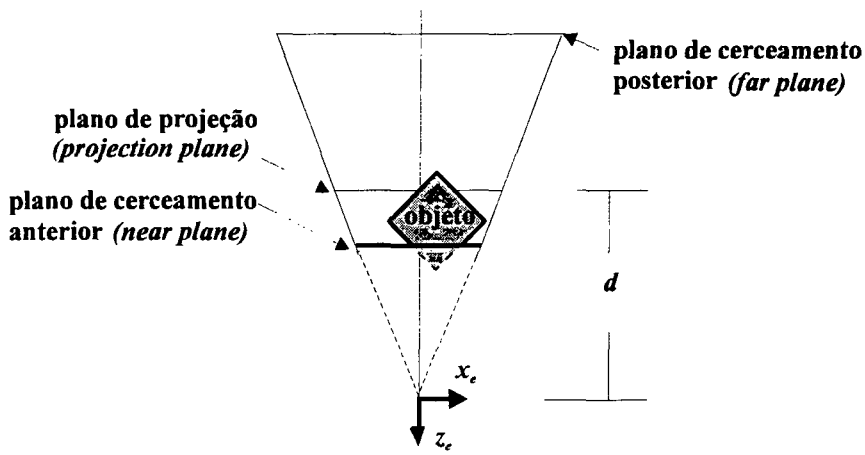


Figura 5-13 Cerceamento do objeto.

O modelo da seção é construído com a metodologia de subdivisões espaciais. O modelo é armazenado na estrutura de dados *radial-edge* e são utilizados os operadores topológicos do SSE na criação da seção.

Para compreender a construção do modelo da seção (Figura 5-14), é importante rever o processo de construção de SGCs descrito nos trabalhos de Cavalcanti (1992). No SSE, uma subdivisão espacial é criada adicionando-se, um a um, os retalhos de superfície que a definem.

Um retalho de superfície determina um único complexo geométrico completo, de dimensão 3, com uma única região, que é ilimitada, e com uma única face. Este complexo deve ser compatibilizado com a subdivisão espacial já existente. Para isto, o retalho deve ser refinado, isto é, subdividido em retalhos inteiramente contidos em uma região (retalhos simples) da subdivisão espacial original. Ao final do refinamento, o retalho, possivelmente, possui várias faces (retalhos simples) que devem ser costuradas na SE, através de operadores topológicos.

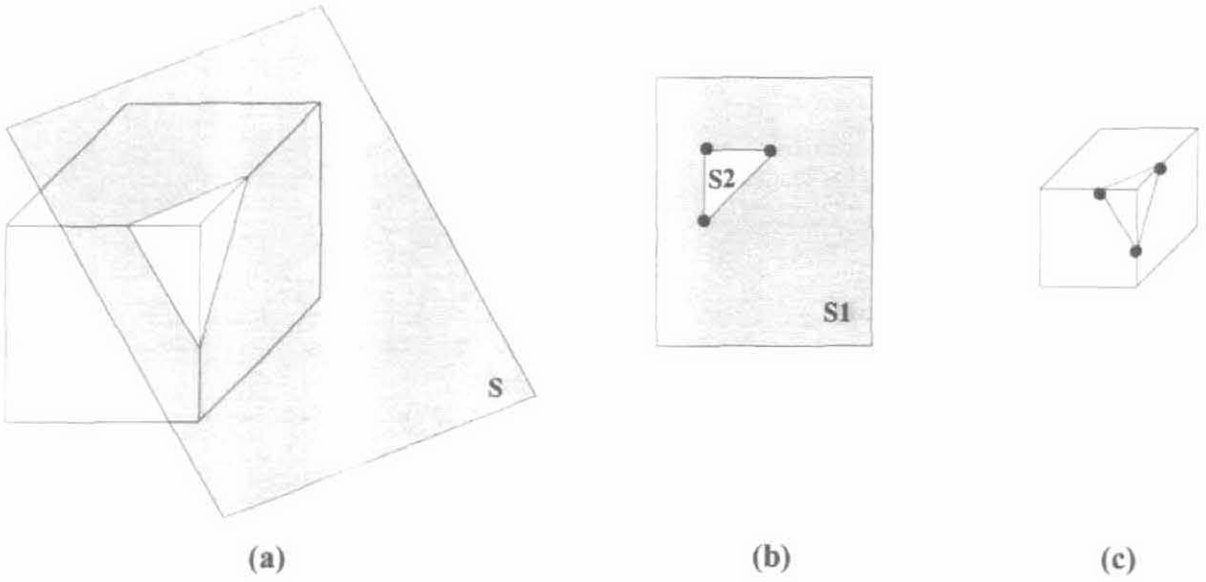


Figura 5-14 Na inserção do retalho S à SE (a), S é decomposto em retalhos mais simples (b) e adicionados à SE com operadores topológicos.

Para construir o modelo da seção, foi criado um **retalho da seção** (Figura 5-15). Este retalho é uma área retangular definida em um plano paralelo ao plano de cerceamento anterior e bastante próximo a ele. O retalho não é definido no próprio plano de cerceamento, para que a imagem da seção não seja cerceada durante a exibição do bloco. Como os limites da janela no plano de projeção e a distância da câmera ao plano de projeção são conhecidos, os limites do retalho da seção podem ser, facilmente, obtidos. Na Figura 5-15, para simplificar a

explicação, os planos de projeção e de cerceamento anterior são coincidentes. A distância Δ_z entre o plano de cerceamento e o plano do retalho da seção representa uma pequena proporção p da distância entre os planos de cerceamento anterior e posterior.

$$\Delta_z = p\ (far - near)$$

$$\Delta_x = 0.5\ p\ (right - left)$$

$$\Delta_y = 0.5\ p\ (top - bottom)$$

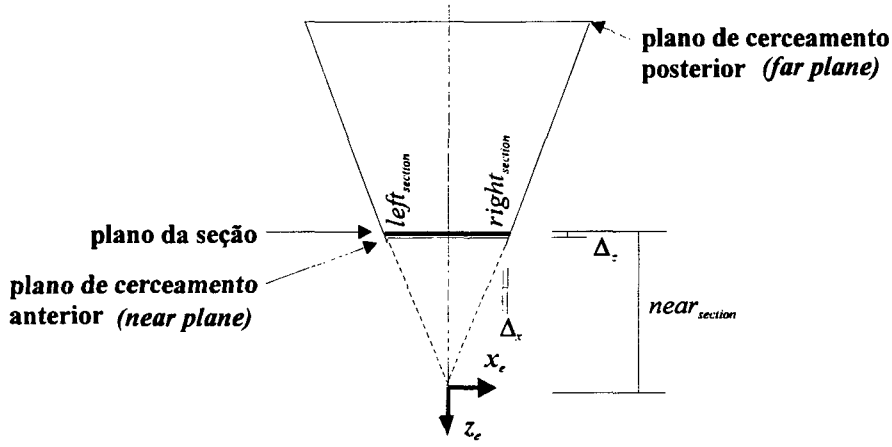


Figura 5-15 Determinação do retalho da seção.

Os limites do retalho da seção são dados por:

$$left_{section} = left - \Delta_x$$

$$right_{section} = right + \Delta_x$$

$$bottom_{section} = bottom - \Delta_y$$

$$top_{section} = top - \Delta_y$$

A distância do plano do retalho da seção ao olho é dada por:

$$near_{section} = near + \Delta_z$$

Determinado o retalho da seção, este deve ser compatibilizado com a subdivisão espacial existente, ou seja, com o modelo do bloco diagrama. O retalho é refinado, sendo

criadas faces (retalhos simples). A costura dos retalhos simples na subdivisão espacial não é feita neste caso, pois o que se deseja é manter o modelo do bloco inalterado e obter um modelo para a seção. Obtido o modelo da seção, ainda é necessário relacionar cada face à região por ela cortada. Isto é feito tomando um ponto interno à face e verificando em que região ele se encontra. Esta informação é guardada como atributo da face.

A estratégia descrita visou à obtenção de uma manipulação simples do plano de corte e à utilização das facilidades da metodologia de subdivisões espaciais. A construção do bloco diagrama e a da seção são feitas de maneira bastante semelhante: bloco e seção são tratados como subdivisões espaciais. Adotando-se a metodologia de subdivisões espaciais, não foi necessário criar estruturas especiais para o armazenamento da seção e, com a utilização dos operadores topológicos (WOps), garantiu-se a consistência topológica do modelo da seção.

6. O Protótipo de Visualização 3D de Unidades Geológicas

A implementação do protótipo foi realizada no ICAD, Laboratório de CAD Inteligente da PUC-Rio, e no Departamento de Exploração da Petrobrás.

O protótipo foi implementado em *workstations* RISC 6000 da IBM (ambiente Unix/Motif), desenvolvido em linguagem C e utiliza a biblioteca gráfica GL. Para o desenvolvimento da interface com o usuário, foi empregada a biblioteca IUP/LED [Levy (1993)] de criação e manipulação de diálogos.

6.1 Organização do Protótipo

A figura abaixo mostra o esquema da tela inicial do protótipo de Visualização 3D de Unidades Geológicas.

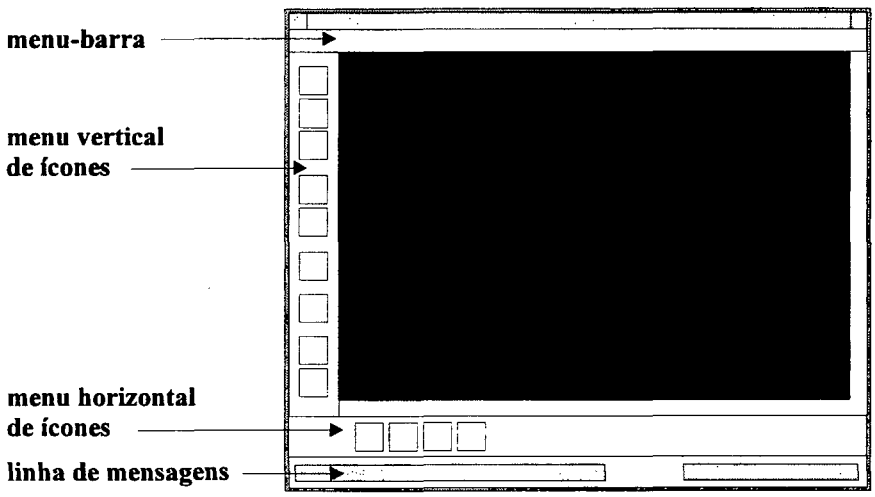


Figura 6-1 Esquema da tela inicial do protótipo.

No **menu-barra**, estão as opções disponíveis a qualquer instante: **Projeto** e **Opções**. Em **Projetos**, são definidos os arquivos que devem ser lidos ou gravados. Em **Opções**, são definidos modos para a exibição do modelo.

Na **área de desenho**, é mostrado o bloco diagrama.

No **menu vertical de ícones**, estão representadas as funções que atuam sobre o bloco diagrama ou sobre a janela de visualização. O menu disponibiliza as funções descritas abaixo:

- operações de rotação, translação e corte do bloco diagrama;
- representação do bloco como um conjunto de unidades geológicas ou como um conjunto de horizontes;
- escolha do tipo de referencial de movimento *Screen-based*, *Walk-through* ou *Object-based*;
- iluminação do bloco diagrama;
- modificação da janela de visualização através de *zoom* ou panorâmica.

No **menu horizontal de ícones**, estão as funções que precisam da seleção (*pick*) de unidades geológicas ou de horizontes. O menu oferece as funções:

- atribuição de litologia às unidades geológicas selecionadas;
- consulta a volume de unidades e a área de horizontes;
- destaque de unidades ou de horizontes;
- seleção de unidades (ou de horizontes) que não devem ser exibidos para que outras unidades (ou horizontes) do interior do bloco possam ser avistadas (avistados).

Na **linha de mensagens**, são fornecidas explicações sobre os ícones e ações que o usuário deve tomar.

6.2 Resultados Experimentais

O bloco diagrama das figuras seguintes (Figura 6-2 a Figura 6-7) foi construído a partir de dados obtidos do sistema de mapeamento MSUP da Petrobrás. O sistema calcula os valores de z para os nós de um *grid* de células retangulares. Para a construção do bloco diagrama, as células do *grid* foram subdivididas em triângulos. Foram fornecidos 1327 retalhos triangulares de superfície para a construção do modelo.

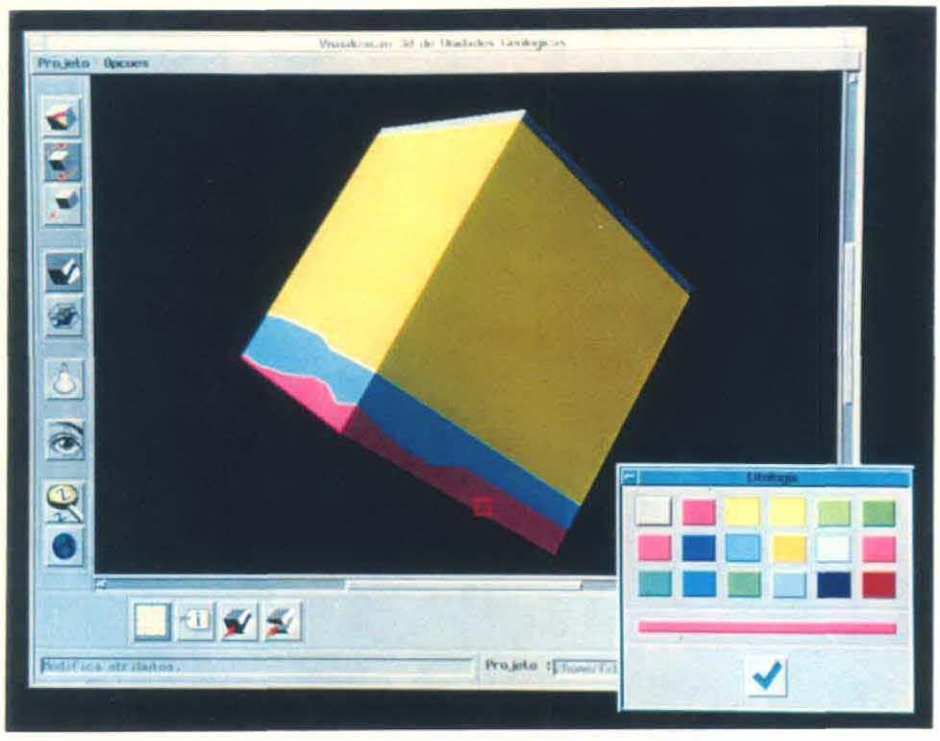


Figura 6-2 Atribuição de litologia à unidade geológica selecionada.

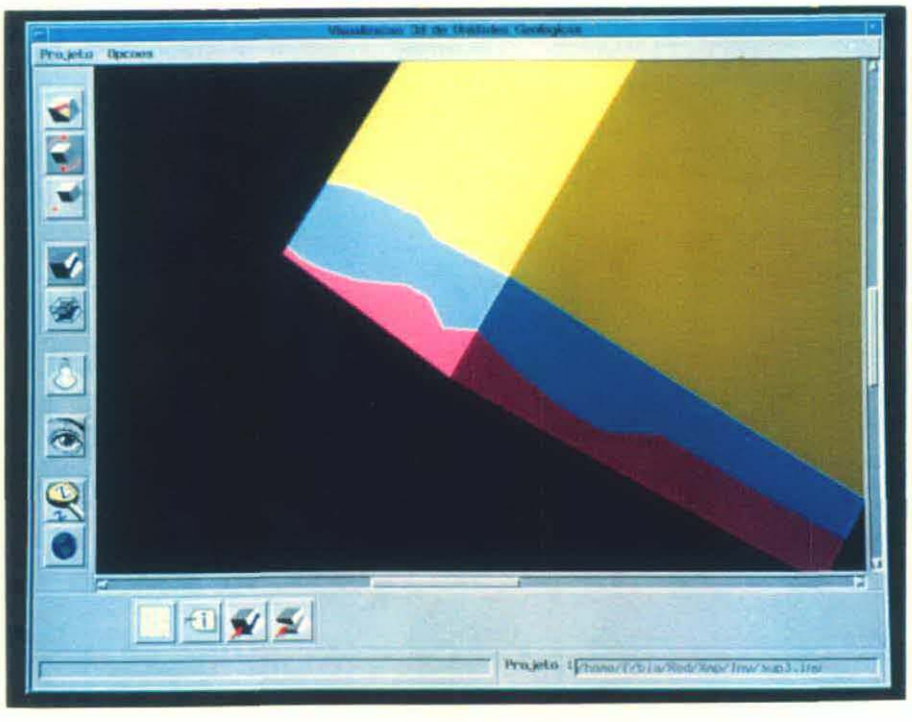


Figura 6-3 Ampliação de parte do bloco através da opção de *zoom*.

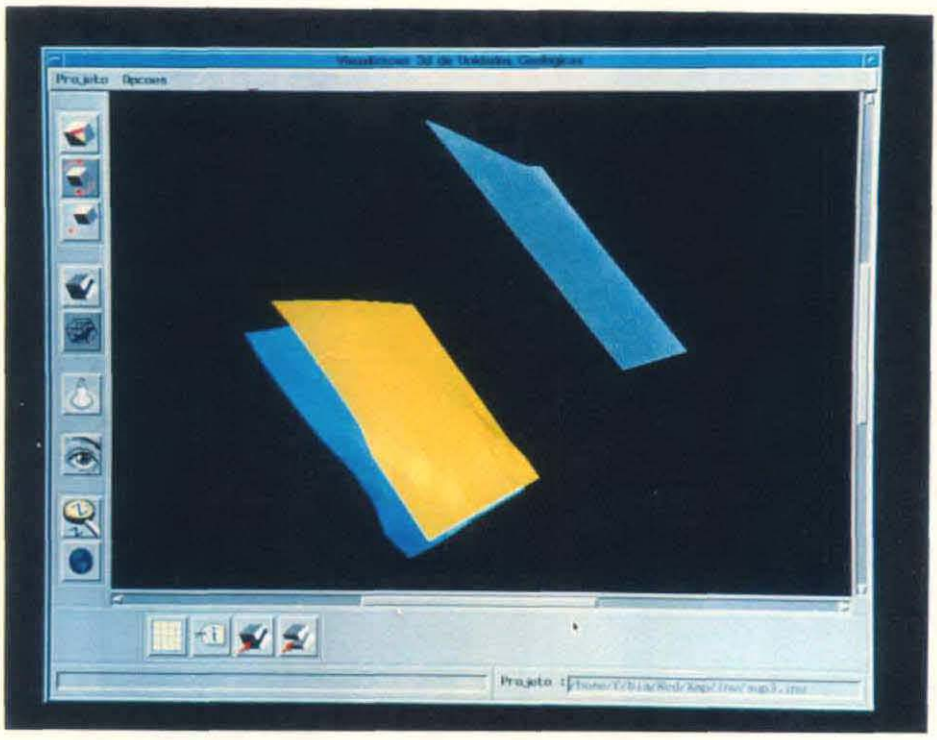


Figura 6-4 Exibição dos horizontes do modelo.

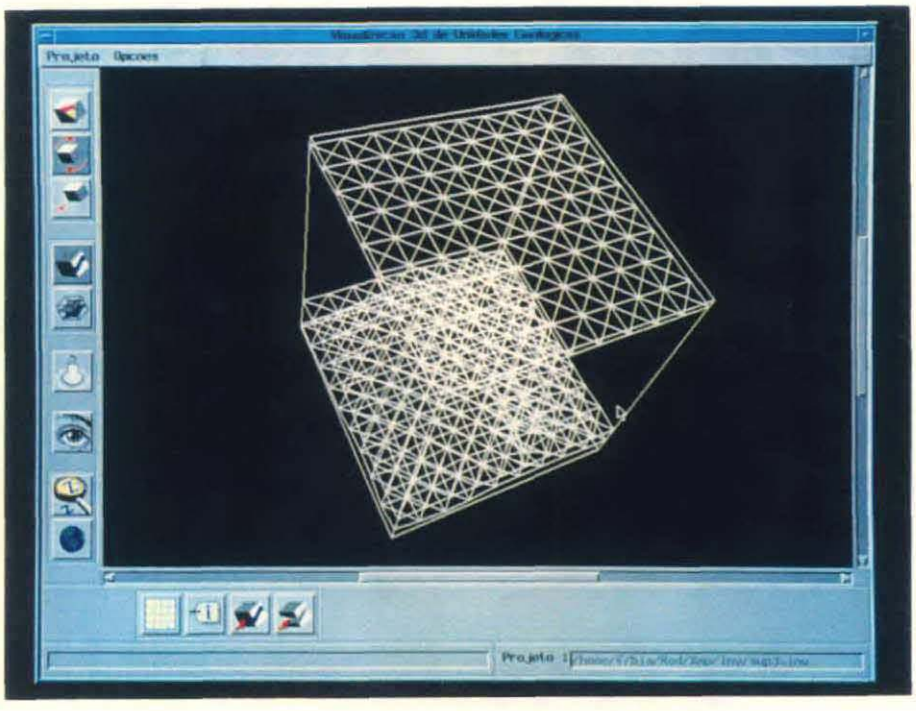


Figura 6-5 Representação do bloco diagrama através de arames.

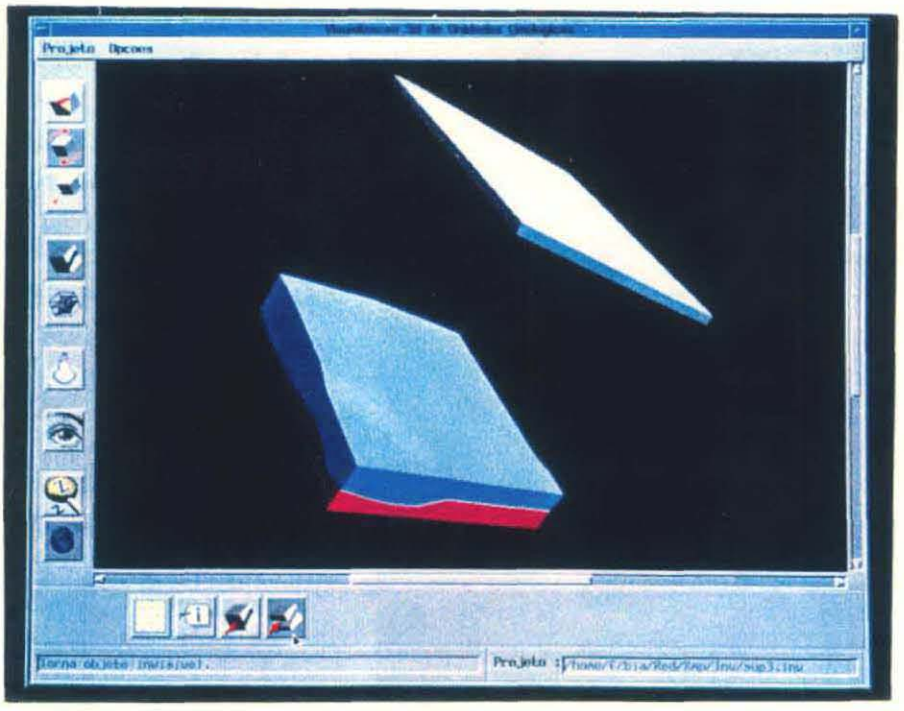


Figura 6-6 Uma unidade geológica é escondida, para que detalhes de outras unidades possam ser vistos.

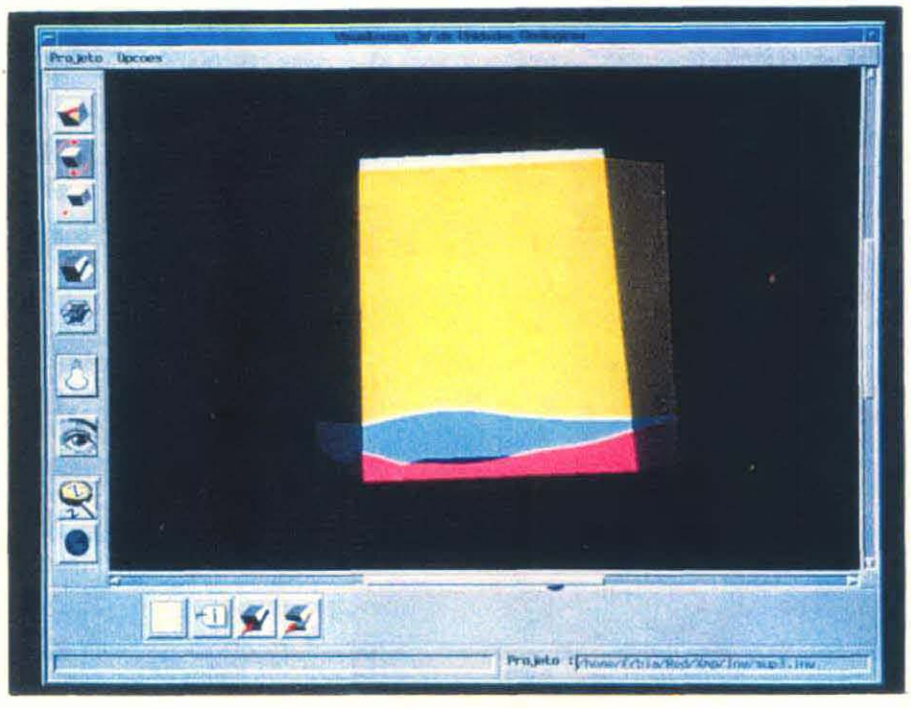


Figura 6-7 O corte revela a existência de uma unidade geológica no interior do bloco diagrama.

7. Conclusão

Visualização e modelagem 3D têm representado uma grande mudança na área de exploração e produção na indústria de petróleo. Para comprovar este fato, basta notar que, há apenas pouco tempo, a interpretação dos dados exploratórios era feita usando-se lápis e papel. Depois, com o uso de computador, mapas e seções passaram a ser feitos de maneira mais rápida e a partir de um volume maior de dados. A transição de ferramentas 2D para 3D é relativamente recente e teve seu desenvolvimento promovido, em grande parte, pelo aperfeiçoamento do *hardware* gráfico.

Problemas 3D têm sido de interesse das comunidades de geologia e de computação gráfica: para geólogos e geofísicos, trabalhar no espaço tridimensional significa ganho em produtividade na interpretação; para a comunidade de computação gráfica, novas e promissoras áreas de pesquisa.

A meta inicial da dissertação era avaliar se a metodologia para construção de subdivisões espaciais é adequada para representar modelos da superfície terrestre. Como conseqüência natural, surgiu a necessidade de visualizar os dados modelados. Além disso, para que o bloco diagrama fosse visto sob vários ângulos, tornou-se indispensável a manipulação interativa do volume pelo usuário.

Esse encadeamento de necessidades está presente em toda a dissertação, que foi organizada em capítulos que tratam de modelagem de sólidos, de exibição de informações gráficas e de modelos de manipulação. As conclusões do trabalho também estão relacionadas a estes itens.

A metodologia de subdivisões planares já foi empregada, com sucesso, na área de geologia e em sistemas bidimensionais de elementos finitos. A estrutura *radial-edge* foi

utilizada, também com êxito, por Martha (1989) na simulação de fraturamento de sólidos. O protótipo de Visualização 3D de Unidades Geológicas é a primeira experiência com a metodologia de subdivisões espaciais (formada pela estrutura *radial-edge*, operadores topológicos e operadores geométricos). Por isto, no desenvolvimento da aplicação o objetivo foi aproveitar, ao máximo, as características da estrutura topológica e os recursos oferecidos pela metodologia: na exibição do bloco diagrama, foram explorados os fatos de a estrutura armazenar, explicitamente, vértices, arestas, faces e regiões e de encontrar relacionamentos de adjacência de modo eficiente; na implementação do corte, o plano que secciona o bloco é tratado como um retalho de superfície e a metodologia foi empregada na definição dos limites, no plano, de cada região cortada.

A metodologia de subdivisões espaciais permite a representação de objetos *non-manifold* e de objetos heterogêneos em material, em dimensão e com estrutura interior. As técnicas de visualização empregadas possibilitaram ver estas características do bloco diagrama. Foram usados recursos como corte e eliminação de partes que escondem outras partes mais internas no bloco. A exibição gráfica foi, em grande parte, facilitada pelo emprego dos recursos de *rendering* da biblioteca gráfica GL.

Girar a cabeça para alterar o ponto de vista, mover a mão para apontar ou tocar um objeto são movimentos bem mais intuitivos do que rolar um *mouse*. Infelizmente, a tecnologia de realidade virtual ainda não está nem suficientemente desenvolvida nem economicamente viável para tornar-se, neste momento, uma interface com usuário comum em aplicações de interpretação geológica. Por esta razão, os modelos sugeridos e estudados propõem soluções para manipulação de objetos 3D através de dispositivos 2D. A taxonomia de manipulação apresentada organizou conhecimentos sobre interação com o usário e sobre a álgebra das transformações. Assim, foi possível uma maior compreensão dos diversos modelos de

manipulação descritos na literatura. A taxonomia pode servir como base para o desenvolvimento de novos modelos de manipulação.

Embora o estudo do protótipo tenha sido voltado para atender às necessidades de uma aplicação geológica, os resultados obtidos podem servir como ponto de partida para quaisquer aplicações que necessitem representar objetos heterogêneos e *non-manifold.*

Por ser uma primeira experiência com a metodologia de subdivisões espaciais, este trabalho traz subsídios para melhoramentos nos algoritmos tanto do protótipo quanto das bibliotecas do SSE.

A inclusão de algoritmos para geometria curva, citada no trabalho de Cavalcanti (1992), certamente seria de grande valia para aplicações geológicas.

O sistema Recon [Ferraz (1993)] reconstitui seções geológicas utilizando subdivisões planares, transformações geométricas e computação gráfica interativa. Uma sugestão para trabalho futuro é a extensão desse sistema para 3D, o que exigiria o estudo das transformações geométricas de reconstituição das unidades geológicas e a criação de uma interface que permita ao usuário editar o objeto 3D.

Os modelos sugeridos para rotação e translação de objetos tridimensionais, em uma avaliação preliminar, apresentaram bons resultados. A continuação desse estudo de manipulação é de grande interesse para a área de CAD. Trabalhos podem surgir uma investigação mais profunda das respostas dos usuários em cada modelo de manipulação.

Apêndice A. Transformações Geométricas para Visualização em 3D

Este apêndice é, praticamente, uma transcrição do artigo de Martha e Gattass (1994), que resume, de maneira didática, as matrizes de transformação geométrica que são utilizadas para visualização em 3D. O artigo foi a base para a taxonomia e os modelos de manipulação propostos no capítulo 5.

Neste trabalho, as matrizes de transformação 4x4 em coordenadas homogêneas são tais que pré-multiplicam um ponto do modelo a ser transformado (visualizado). Desta forma, a concatenação de uma matriz de transformação à matriz que acumula as transformações sempre se dá na forma de pré-multiplicação.

A.1. Parâmetros de Visualização em 3D

A abstração utilizada para a especificação dos parâmetros de visualização em 3D segue, quase que integralmente, um dos modelos definidos pelo *OpenGL* [Neider (1993)], o **modelo de câmera** (Figura A-1). Como este modelo já foi explicado no capítulo 5, neste apêndice, será feita somente uma rápida revisão.

Os parâmetros necessários para posicionar a câmera no espaço de modelagem são:

- posição da câmera (olho):

  (eye_x, eye_y, eye_z)

- posição do ponto de referência (um ponto no espaço de modelagem para onde a câmera mira):

  (ref_x, ref_y, ref_z)

- vetor de orientação vertical da câmera (*view up-vector* – ***vup***):

  (vup_x, vup_y, vup_z).

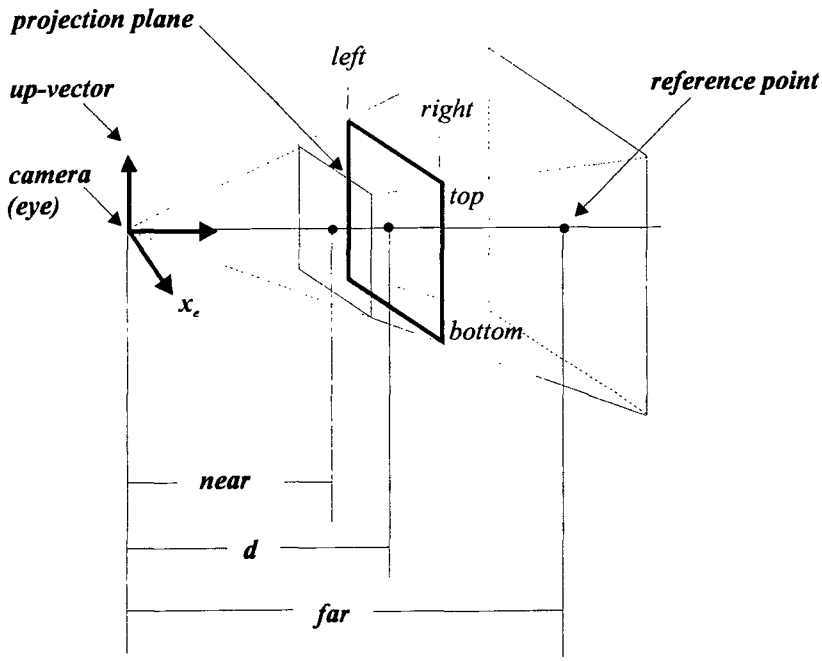


Figura A-1 Modelo de câmera e frustum de visão.

Os vetores unitários que definem o sistema de coordenadas do olho podem ser definidos como:

$$\boldsymbol{view} = (ref_x, ref_y, ref_z) - (eye_x, eye_y, eye_z)$$

$$\boldsymbol{z_e} = - \boldsymbol{view} / ||\boldsymbol{view}||$$

$$\boldsymbol{x_e} = (\boldsymbol{vup} \times \boldsymbol{z_e}) / ||\boldsymbol{vup} \times \boldsymbol{z_e}||$$

$$\boldsymbol{y_e} = \boldsymbol{z_e} \times \boldsymbol{x_e}.$$

A.2 Transformação do Espaço de Modelagem para o Sistema da Câmera

A primeira transformação geométrica sofrida pelos objetos de um ambiente para serem visualizados é uma mudança de base do sistema de coordenadas de modelagem (do ambiente) para o sistema de coordenadas do olho. Esta mudança de base envolve uma translação da origem do espaço de modelagem para a posição da câmera (olho) e uma rotação do sistema de eixos de modelagem para se igualar ao sistema de eixos do olho.

A matriz que faz a transformação de translação [*T*] é formada pelas coordenadas da posição do olho no espaço de modelagem:

$$[T]=\begin{bmatrix} 1 & 0 & 0 & -eye_x \\ 0 & 1 & 0 & -eye_y \\ 0 & 0 & 1 & -eye_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A matriz que faz a transformação de rotação [*R*] tem como linhas da sub-matriz 3x3 superior esquerda as componentes dos vetores unitários da base do sistema do olho descritos no sistema de coordenadas de modelagem:

$$[R]=\begin{bmatrix} x_{ex} & x_{ey} & x_{ez} & 0 \\ y_{ex} & y_{ey} & y_{ez} & 0 \\ z_{ex} & z_{ey} & z_{ez} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A mudança de base do espaço de modelagem para o espaço do olho pode ser grupada em uma única matriz de transformação, definida como **matriz de posicionamento de câmera**:

$[C] = [R][T]$.

A.3 Perspectiva Canônica e Espaço da Tela

Após a translação [T] e a rotação [R], os objetos estão prontos para serem projetados no plano de projeção, fazendo-se uma projeção cônica (perspectiva) ou ortográfica. A projeção cônica para objetos já transformados para o sistema do olho é denominada de perspectiva canônica. Este tipo de projeção é mostrado na Figura A-2 para a coordenada x_e, sendo que para a coordenada y_e o tratamento é análogo.

As coordenadas projetadas de um ponto qualquer (x_e, y_e, z_e) são dadas por:

$$x_p = \frac{d}{-z_e} x_e$$

$$y_p = \frac{d}{-z_e} y_e$$

$$z_p = -d$$

As coordenadas (x_p, y_p, z_p) são as coordenadas de um ponto no espaço de modelagem transformadas e projetadas no plano de projeção da tela. No entanto, não existe nenhuma informação sobre a profundidade do ponto para o interior da tela. Esta profundidade é uma informação importante pois possibilita a correta identificação dos objetos que estão mais próximos do plano de cerceamento anterior e que, por isso, são os objetos visíveis. É, então, criado um espaço de coordenadas (x_S, y_S, z_S), chamado de sistema de coordenadas da tela, tal que

$x_S = x_p$

$y_S = y_p$

$z_S = \textit{profundidade}$

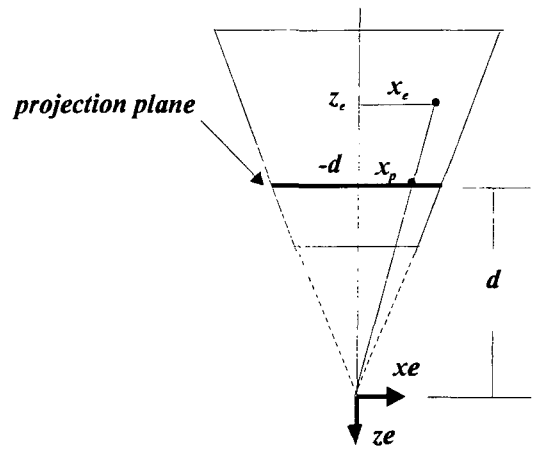


Figura A-2 Perspectiva canônica.

Na verdade, para desenhar na tela um ponto (x_S, y_S, z_S), precisa-se apenas usar as coordenadas x_S e y_S e ignorar a coordenada z_S. O desenho é uma **projeção ortográfica** do objeto transformado para o sistema de coordenadas da tela. Isto é, a transformação do objeto para o sistema de coordenadas da tela o distorce de tal maneira que o resultado da projeção ortográfica no plano $x_S y_S$ é igual à transformação de projeção cônica sobre o objeto.

A transformação do objeto para o espaço da tela é conveniente, pois faz que o processo de remoção de linhas e superfícies ocultas de uma imagem seja feito com base em retas perpendiculares ao plano de projeção (projeção ortográfica). Isto é, para saber se um ponto de uma linha ou superfície é obscurecido por outro basta comparar a profundidade z_S dos dois pontos: o que tiver a menor profundidade é o ponto que aparecerá na tela.

Entretanto, para que o resultado da transformação do objeto para o sistema de coordenadas da tela seja útil para a eliminação de linhas e superfícies escondidas, é necessário que se calcule a profundidade de uma linha, não somente para os seus pontos extremos, mas também para qualquer ponto intermediário. Para que a interpolação de um ponto intermediário no espaço da tela seja simples, é preciso que linhas retas e planos no sistema de coordenadas

do olho sejam transformados para linhas retas e planos no sistema de coordenadas da tela. Pode-se demonstrar [Newman-Sproull (1979)] que a relação que deve existir entre z_e e z_S para que estas condições sejam atendidas é $z_S = \alpha + \beta / z_e$, onde α e β são quaisquer números reais.

Existem infinitos valores de α e β que satifazem a esta imposição de planaridade. Isto é, dados dois valores quaisquer para α e β, tem-se que qualquer reta ou plano no sistema de coordenadas do olho vai se transformar em um plano ou reta no sistema de coordenadas da tela. De fato, o coeficiente α pode ser considerado uma translação de corpo rígido e β um coeficiente influenciando a quantidade de distorção por perspectiva.

Os valores de α e β podem ser escolhidos de tal forma que todos os valores z_e que estão dentro do frustum de visão no sistema de coordenadas do olho mapeiem para valores convenientes de z_S no sistema de coordenadas da tela. Uma solução proposta neste trabalho é manter os valores da coordenada z_e para os planos frontal e posterior de cerceamento no espaço da tela. Isto é, associa-se uma coordenada $z_S = -near$ para pontos com $z_e = -near$ e associa-se $z_S = -far$ para pontos com $z_e = -far$. O resultado disso é:

$$\alpha = -(far + near)$$

$$\beta = -(far\ near)$$

Resumindo, as coordenadas de um ponto no sistema de coordenadas do olho são transformadas para o sistema de coordenadas da tela da seguinte maneira:

$$x_s = \frac{d}{-z_e} x_e$$

$$y_s = \frac{d}{-z_e} y_e$$

$$z_s = -(far + near) - \frac{far \cdot near}{z_e}$$

Esta transformação pode ser expressa por uma matriz, denominada de **matriz de perspectiva canônica** [*P*], tal como mostrada abaixo, onde os símbolos (*n*, *f*) foram utilizados para representar (*near*, *far*). Nota-se que somente após a divisão pela quarta coordenada homogênea a transformação de um ponto por esta matriz resulta nas expressões acima.

$$[P] = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & (f+n) & (f\,n) \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

A.4 Normalização de Coordenadas

Para complementar as transformações geométricas, em geral, é feita uma **normalização** de coordenadas. Esta normalização vai ser útil para a implementação de algoritmos de cerceamento ou remoção de linhas e superfícies ocultas. A exemplo do que é feito no *OpenGL*, para esta normalização, a menor coordenada $z_S = -1$ corresponde ao plano $z_e = -near$ e a maior coordenada $z_S = +1$ corresponde ao plano $z_e = -far$. Nota-se que é feito um espelhamento em relação ao plano $x_S y_S$. Isto é dado para que pontos no plano anterior de cerceamento tenham a menor profundidade z_S, e pontos no plano posterior de cerceamento tenham a maior profundidade z_S.

Para projeções ortográficas, a **matriz de normalização de coordenadas** [*N*] pode ser deduzida a partir de uma transformação de translação do sistema de eixos para o centro do volume de visão, seguida de uma transformação de escala nas três direções, e finalmente de uma transformação de espelhamento em relação $x_S y_S$. Isto resulta na concatenação das duas

matrizes mostradas abaixo, onde adotaram-se os símbolos (*l*, *r*, *b*, *t*, *n*, *f*) para representar os limites do volume de visão (*left*, *right*, *bottom*, *top*, *near*, *far*):

$$[N]=\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & -S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & -\Delta_x \\ 0 & 1 & 0 & -\Delta_y \\ 0 & 0 & 1 & -\Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{onde} \begin{cases} S_x = \dfrac{2}{r-l} \\ S_y = \dfrac{2}{t-b} \\ S_z = \dfrac{2}{f-n} \end{cases} \quad \text{e} \quad \begin{cases} \Delta_x = \dfrac{r+l}{2} \\ \Delta_y = \dfrac{t+b}{2} \\ \Delta_z = -\dfrac{f+n}{2} \end{cases}$$

A matriz de normalização para projeções ortográficas que resulta é:

$$[N]=\begin{bmatrix} \dfrac{2}{r-l} & 0 & 0 & -\dfrac{r+l}{r-l} \\ 0 & \dfrac{2}{t-b} & 0 & -\dfrac{t+b}{t-b} \\ 0 & 0 & \dfrac{-2}{f-n} & -\dfrac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Para projeções cônicas, pode-se obter uma única matriz de transformação que engloba a perspectiva canônica [*P*] e a normalização de coordenadas [*N*]. Esta é a chamada **matriz de frustum** [*F*] = [*N*][*P*].

A concatenação de [*P*] e [*N*] para a obtenção de [*F*] pode ser entendida da seguinte maneira:

- A primeira transformação [*P*] a ser aplicada distorce o frustum de visão de um tronco de pirâmide para um prisma, onde as coordenadas máximas e mínimas z_e nos planos de cerceamento anterior e posterior são preservadas. As outras coordenadas máximas e mínimas deste prisma correspondem à janela de visão (*left*, *right*, *bottom*, *top*) definida no plano de projeção.

- A transformação [*P*] distorceu o frustum de visão para um prisma e fez o problema recair em uma normalização de projeções ortográficas. Portanto, basta aplicar a transformação [*N*] para normalizar as coordenadas.

A matriz [*F*] resultante é mostrada abaixo. No *OpenGL,* esta matriz é apresentada com o parâmetro *near* (*n*) no lugar do parâmetro *d* pois é assumido que o plano de projeção é o plano de cerceamento anterior.

$$[F]=\begin{bmatrix} \frac{2d}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2d}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Pode-se resumir a seqüência de operações geométricas que um ponto no espaço de modelagem sofre para ser transformado para o espaço normalizado da tela. Isto é feito através da seguinte concatenação de matrizes:

$\{p_S\} = [N][P][R][T]\ \{p\}$

$\{p_S\} = [F][C]\ \{p\}$

A concatenação com a matriz [*P*] não é feita no caso de projeções ortográficas.

Referências Bibliográficas

V. Bui-Tran, J. L. Pajon, P. Joseph e J. M. Chautru, *3D Reservoir Visualization*, Journal of Petroleum Technology, vol. 43, no. 11, pp. 1310-1314, 1991.

M. Chen, S. J. Mountford e A. Sellen, *A Study in Interactive 3D Rotation Using 2D Control Devices*, Proceedings of ACM SIGGRAPH, New York, vol. 22, no. 4, pp. 121-129, 1988.

P. C. P. Carvalho e L. H. Figueiredo, *Introdução à Geometria Computacional*, Publicação do IMPA, 1991.

P. R. Cavalcanti , P. C. P. Carvalho e L. F. Martha, *Criação e Modelagem de Subdivisões Planares*, Anais do IV SIBGRAPI, São Paulo, pp. 13-24, 1991.

P. R. Cavalcanti, *Criação e Manutenção de Subdivisões do Espaço*, Tese de Dourorado, Departamento de Informática, PUC-Rio, 1992.

P. R. Cavalcanti, P. C. P. Carvalho e L. F. Martha, *Representação de Objetos Heterogêneos através de Decomposição do Espaço*, Anais do VI SIBGRAPI, Recife, pp. 277-284, 1993.

P. R. Cavalcanti, P. C. P. Carvalho e L. F. Martha, *Construção de Decomposições do Espaço e Objetos Heterogêneos*, Anais do VII SIBGRAPI, Curitiba, pp. 279-285, 1994.

M. F. R. Ferraz, *Reconstituição de Seções Geológicas utilizando Subdivisões Planares, Transformações Geométricas e Computação Gráfica Interativa*, Dissertação de Mestrado, Departamento de Informática, PUC-Rio, 1993.

J. M. Gomes e L. Velho, *Implicit Objects in Computer Graphics*, Publicação do IMPA, 1992.

T. A. Jones, *Modeling Geology in Three Dimensions*, Geobyte, vol. 3, no. 1, pp. 14-20, 1988.

C. H. Levy, *IUP/LED: Uma Ferramenta Portátil de Interface com Usuário*. Dissertação de Mestrado, Departamento de Informática, PUC-Rio, 1993.

M. Mäntylä, *An Introduction to Solid Modeling*, Computer Science Press, 1988.

L. F. Martha, *Topological and Geometrical Modeling Approach to Numerical Discretization and Arbitrary Fracture Simulation in Three Dimensions*, PhD Thesis, Cornell University, Ithaca-NY, 1989.

L. F. Martha e M. Gattass, *Um Resumo das Transformações Geométricas para Visualização em 3D*, Comunicações do VII SIBGRAPI, Curitiba, pp. 9-12, 1994.

J. Neider, T. Davis e M. Woo, *OpenGL Programming Guide - The Official Guide to Learning OpenGL*, Release 1, Addison Wesley, 1993.

W. Newman e R. Sproull, *Principles of Interactive Computer Graphics*, 2nd edition, McGraw-Hill, 1979.

J. R. Osborn e A. M. Agogino, *An Interface for Interactive Spatial Reasoning* and *Visualization*, Proceedings of CHI'92, ACM Press, pp. 75-82, 1992.

F. P. Preparata e M. I. Shamos, *Computational Geometry*, Springer-Verlag, 1985

A. A. G. Requicha, *Representations of Solid Objects: Theory, Methods and Systems*, ACM Computer Surveys, vol. 12, no. 4, pp. 438-464, 1980.

A. A. G. Requicha e J. R. Rossignac, *Solid Modeling and Beyond* , IEEE Computer Graphics and Applications,vol. 12, no. 5, pp. 31-44, 1992.

A. A. G. Requicha e H. B. Voelcker, *Solid Modeling: A Historical Summary and Contemporary Assessment*, IEEE Computer Graphics and Applications, vol. 2, no. 2, pp. 9-24, 1982.

A. A. G. Requicha e H. B. Voelcker, *Solid Modeling: Current Status and Research Directions*, IEEE Computer Graphics and Applications, vol. 3, no. 7, pp. 25-37, 1983.

J. R. Rossignac e M. A. O'Connor, *SGC: A Dimension-Independent Model for Pointsets with Internal Structures and Incomplete Boundaries*, Geometric Modeling for Product Engineering, North Holland, pp. 145-180, 1990.

J. R. Rossignac e A. A. G. Requicha, *Constructive non-regularized Geometry*, Computer-Aided Design, vol. 23, no. 1, pp. 21-32, 1991.

P. Sabella e I. Carlbom, *An Object-Oriented Approach to the Solid Modeling of Empirical Data*, IEEE Computer Graphics and Applications, vol. 9, no. 5, pp. 24-35, 1989.

K. Shoemake, *Arcball: A User Interface for Specifying Three-dimensional Orientation Using a Mouse*, Proceedings of Graphics Interface'92, pp. 151-156, 1990.

M. van Emmerick, *A Direct Manipulation Technique for Specifying 3D Object Transformations with a 2D Input Device*, Computer Graphics Forum, vol. 9, pp. 335-361, 1990.

P. Wawrzynek, *Discrete Modeling of Crack Propagation: Theoretical Aspects and Implementation Issues in Two and Three Dimensions*, PhD Dissertation, Cornell University, Ithaca-NY, 1991.

K. Weiler, *Topological Structures for Geometric Modeling*, PhD Thesis, Rensselaer Polytechnic Institute, Troy-NY, 1986.

IBM AIX Version 3.2 for RISC System/6000, *Graphics Programming Concepts*, 1992.

"Visualização 3D de Unidades Geológicas Representadas por Subdivisões Espaciais" apresentada por **Beatriz Castier**, em 27 de abril de 1995, ao Departamento de Informática da PUC-Rio e aprovada pela Comissão Julgadora, formada pelos seguintes membros:

Prof. Luiz Fernando C. R. Martha
(Orientador)
Departamento de Engenharia Civil - PUC-Rio

Prof. Marcelo Gattass
Departamento de Informática - PUC-Rio

Prof. Paulo Cezar Pinto Carvalho
Instituto de Matemática Pura e Aplicada

Geraldo de Oliveira Santos
Petrobras - Petróleo Brasileiro S.A.

Prof. Luiz Henrique de Figueiredo
Instituto de Matemática Pura e Aplicada

Visto e permitida a impressão.
Rio de Janeiro, 18 / 10 / 95

Coordenador de Programas de Pós-Graduação
e Pesquisa do Centro Técnico Científico