Universidade Estadual de Campinas

Faculdade de Engenharia Elétrica

Departamento de Engenharia de Computação e Automação Industrial

Técnicas básicas para interações 3D através do *mouse*

Autor: **Delia Perla Patricia Velásquez Alegre**

Orientadora: **Profa. Dra. Wu Shin-Ting**

Dissertação submetida à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas, para preenchimento dos pré-requisitos parciais para obtenção do Título de Mestre em Engenharia Elétrica.

5 de Janeiro de 1998

Formaram parte da Banca:

- *Prof^a. Dr^a. Wu Shin-Ting – Universidade Estadual de Campinas;*
- *Prof. Dr. Léo Pini Magalhães – Universidade Estadual de Campinas;*
- *Prof^a. Dr^a. Maria Cristina Ferreira de Oliveira – USP de São Carlos.*

Data da defesa: 11.10.95.

Sumário

Lista de Figuras

Convenções

- Foi convencionado o uso do tipo *itálico* para terminologia em inglês. Algumas palavras foram destacadas com o tipo sans serif para indicar relevância no texto.

- Termos como por exemplo *hardware*, *software* e *mouse* foram mantidos no seu original em inglês devido não a uma dificuldade de tradução, mas por representarem conceitos bem definidos e utilizados na literatura.

Resumo

Neste trabalho foi formalizado e implementado um conjunto de técnicas de interação 3D que permite ao usuário comunicar-se diretamente com um ambiente 3D – modelado por computador – através de dispositivos convencionais como o *mouse* e a tela bidimensional.

Estas técnicas permitem a identificação e posicionamento das entidades gráficas de objetos 3D no espaço do mundo real (WC) a partir do espaço bidimensional dos dispositivos convencionais (DC). As técnicas abordadas foram: identificação ordenada, movimentos circulares, particionamento do espaço imagem e plano de trabalho. Para auxiliar ao usuário nas suas interações com o espaço 3D foram usadas ferramentas visuais tais como grades e eixos 3D.

Para permitir a reutilização das técnicas em interfaces gráficas tridimensionais, a estratégia de desenvolvimento das mesmas se deu na forma de componentes de interface. Essas componentes são integrantes da biblioteca IQLT para interações com o ambiente 3D.

Uma destas técnicas – a técnica dos movimentos circulares – foi integrada à interface gráfica interativa *ProSIm* (*Prototipação de Síntese de Imagens Fotorealísticas*) apresentando o uso de uma estratégia para manipulação interativa da câmera.

Acredita-se que a formalização das técnicas de interação nas suas diferentes coordenadas de visualização, abre a possibilidade que elas sejam exploradas em diferentes contextos da manipulação, tanto no espaço objeto (WC) quanto no espaço imagem (DC), nos trabalhos futuros.

Abstract

A kit of 3D interaction tools that allow direct communication between the user and a computer modeled 3D enviroment has been formalized and implemented for conventional devices like mouse and bidimensional screen.

These tools allow identification and positioning of graphical entities of 3D objects on the real world space (WC) from the bidimensional space of conventional devices (DC). The interactive tools studied are: ordered identification, circular movemets, partitioning of image space and working plane. Visual tools like 3D grids and 3D axis were used to improve the user interaction with the 3D space.

To assure the reusability of these tools for 3D graphical interfaces the studied tools were implemented as widgets. These widgets are integrated in the toolkit named IQLT.

The circular movement tool interactive has been applied on the interactive graphical interface ProSIm (Photorealistic Images Sintesis Prototipation) to manipulate interactively the movements of camera.

We believe that the formalization of interaction tools in different visualization coordinates open the possibility to exploit the applicability of these tools in distinguished 3D contexts in future works.

Agradecimentos

Este é o meu muito obrigada a todas as pessoas que contribuiram para a realização deste trabalho.

- Agradeço profundamente a inteira e incondicional dedicação da amiga e orientadora Ting e reconhecer através deste trabalho a sua qualidade profissional na área de computação gráfica.
- Aos membros da Banca, Prof. Léo Pini, Profa. Maria Cristina de Oliveira e Prof. Clésio Tozzi, pela participação. De forma muito especial agradeço ao Prof. Léo pelo seu acompanhamento e apoio neste trabalho.
- À CAPES, como orgão que financiou durante 30 meses este trabalho de pesquisa;
- A Eric Bier (University of California) pelo envio dos seus trabalhos na área de modelamento geométrico.
- Aos meus amados pais, Juan e Delia pelo incentivo, amor e cuidados. E aos meus amados irmãos Andrea e Juanito pelo apoio incondicional.
- À pessoa que Deus me deu, Carlos Alberto, pela companhia, amor, e disposição permanente para a correção do texto.
- Aos irmãos em Cristo: Elton, Nelson, Márcio Leandro e Carlos, meus melhores amigos.
- Às minhas discípulas Célia e Vanda e discipuladores Andrea e Reginaldo pelo apóio espiritual nas horas difícieis.
- As várias pessoas que se envolveram diretamente com este trabalho, entre elas: Marcelo Malheiros (DCA) pela sua disposição na inclusão das técnicas no *ProSIm*. A Andrea (DCA), por ajudarme no esboço da primeira versão da dissertação. Flávio Navarro (DCA) pelo seu interesse e continuidade que dará ao meu trabalho; Mauricio Ferreira (DCA) e Marcelo Cordeiro (DCA) pelas palavras de incentivo; Reginaldo (DT) pela correção de uma primeira versão do texto; Leandro (LCAE) pelas "dicas" na programação de uma das técnicas; Ladislau (DCA) pela sua ajuda com o Latex; a Nina (DCA) pelas caronas;
- e sobretudo, Àquele a quem pertence toda honra, todo louvor, toda ciência, toda sabedoria e todas minhas ações de graças.

Louvai ao Senhor.
Louvai a Deus no seu santuário;
lovai-o no firmamento do seu poder.
Louvai-o pelos seus atos poderosos,
louvai-o conforme a excelência da sua grandeza.
Louvai-o com o som da trombeta,
louvai-o com saltério e harpa,
louvai-o com adufes e danças,
louvai-o com instrumentos de cordas e com flauta,
louvai-o com címbalos sonoros,
louvai-o com címbalos altissonantes.
Tudo o que têm fôlego louve ao Senhor.
Louvai ao Senhor.
Salmos 150:1-6

A Jesus Cristo,
o louvor do meu trabalho.

Capítulo 1

Introdução

" Peça a Deus que abençõe os seus planos,
e eles darão certo".
Provérbios 16:3.

Uma das principais preocupações na construção de interfaces gráficas para aplicações 3D está na obtenção de uma interação eficiente entre o usuário e o ambiente 3D. A comunicação entre o usuário e o ambiente tridimensional através dos atuais dispositivos de entrada e de saída figura entre as principais dificuldades para se obter uma interação eficiente; isto se deve à característica bidimensional dos dispositivos usuais de entrada (*mouse* 2D, mesa digitalizadora, *joystick*) e de saída (monitor) frente à tridimensionalidade do espaço e dos objetos da cena.

Uma abordagem dada ao problema de interação usuário-ambiente 3D tem sido através do desenvolvimento do *hardware* (ou desenvolvimento de novos dispositivos 3D, tais como, o *flyer mouse* [Ware88] e o *roller mouse* [Veno93]). A justificativa para a criação destes dispositivos tem sido a necessidade de se obter uma correspondência natural entre as ações dos usuários sobre os dispositivos e o movimento obtido no espaço 3D. Porém, muitos destes novos dispositivos são bastante específicos, relativamente caros e não portáteis.

Uma outra abordagem ao mesmo problema de interação tem sido através do desenvolvimento de ferramentas de *software* ou de técnicas de interação que permitem o mapeamento entre dispositivos de entrada bidimensionais e o espaço 3D. A justificativa para o uso

desta abordagem é de que os dispositivos de entrada 2D, aliados à evolução das técnicas de mapeamento, tem permitido interações tão ou mais amigáveis do que os novos dispositivos:

> " a interação com o espaço tridimensional a partir de dispositivos bidimensionais tem permanecido praticamente inexplorada. Este é um ramo que apresenta ainda fortes possibilidades de exploração..." [Conn92].

1.1 Objetivos do trabalho

Nosso trabalho visa uma abordagem por *software* fazendo uso de dispositivos convencionais, mais especificamente o *mouse* e a tela bidimensional, de ampla popularidade, baixo custo e fácil disponibilidade entre os usuários de sistemas gráficos. Em termos de *software*, a maioria das plataformas gráficas 2D disponíveis no mercado – tais como o *XView* [Hell90] e *XMotif* [Moti89] – previram o uso do *mouse* comum como dispostivo de entrada e a tela de visualização como dispositivo de saída. Em termos de recursos humanos, a maioria de programadores está bem treinada para desenvolver aplicativos com o uso do *mouse*. Em termos de *hardware*, o *mouse* e a tela de visualização são facilmente instalados em qualquer arquitetura. Estudos realizados demonstraram que entre os dispositivos de posicionamento existentes no mercado o *mouse* é um dos mais rápidos [Doug94].

O objetivo deste trabalho é desenvolver um conjunto de técnicas de mapeamento ou interações 3D para solução ao problema de comunicação usuário-ambiente 3D, visando o uso de dispositivos mouse e tela bidimensionais. Com o intuito de permitir a reutilização destas técnicas em interfaces gráficas tridimensionais, a estratégia de desenvolvimento das mesmas se deu na forma de componentes de interface. Como aplicação deste trabalho foi implementada a biblioteca de componentes 3D para interações e visualização do ambiente 3D (IQL**T**). Como aplicação propõe-se a integração de uma destas componentes na interface gráfica interativa *ProSIm* (*Prototipação de Síntese de Imagens Fotorealísticas*) apresentando o uso de uma estratégia para manipulação interativa da câmera. Também foram integrados os eixos como sistemas de referência e as grades como ferramenta auxiliar à percepção tridimensional.

1.2 Visão geral do trabalho

O trabalho é organizado em seis capítulos e quatro apêndices:

O capítulo 2 apresenta uma revisão bibliográfica das técnicas de mapeamento 3D e de visualização 3D. Novos dispositivos de entrada 3D e novos dispositivos de saída 3D são também comentados.

O capítulo 3 resume todos os conceitos e terminologia preliminares necessários à compreensão do trabalho.

O capítulo 4 é uma descrição detalhada da solução proposta, focalizando os algoritmos de mapeamento e as técnicas de visualização 3D.

O capítulo 5 descreve os detalhes da implementação, resultados e aplicações da solução proposta: uma biblioteca de componentes 3D para interações e visualização do ambiente 3D (IQL**T**).

O capítulo 6 contém a conclusão do trabalho e propõe novos caminhos de exploração.

Há ainda quatro apêndices. O apêndice A é um estudo dos dispositivos de posicionamento 2D mais usados no mercado. O apêndice B contém o manual de referência dos componentes. O apêndice C inclui o manual de programação dos componentes do IQL**T**. E o apêndice D descreve os *drivers* de comunicação para uso da biblioteca IQL**T** num sistema de suporte para desenvolvimento de interfaces: o *XView*.

Capítulo 2

Revisão bibliográfica

" Onde não há conselho frustram-se os projetos,
mas com a multidão de conselheiros eles se estabelecem".
Provérbios 15:22.

Este capítulo apresenta um resumo dos trabalhos mais relevantes existentes na literatura sobre interações diretas com o espaço 3D através de dispositivos de posicionamento (*locators*). A ação dos usuários sobre esses dispositivos, os efeitos e a representação gráfica dessas ações assim como as técnicas de visualização 3D, têm sido alvo de pesquisa para obter uma interação visual eficiente. Distinguem-se basicamente duas abordagens:

- soluções por *software* e
- soluções por *hardware*.

2.1 Entrada

2.1.1 Soluções por *software*: Técnicas de mapeamento

Na literatura foram encontradas algumas técnicas de mapeamento que são abordagens por *software* para resolver o problema de interação com o espaço 3D a partir de duas dimensões. Estas técnicas procuram correspondências entre as coordenadas (x', y') captadas pelos dispositivos 2D convencionais e os pontos (x, y, z) do espaço 3D.

Uso de movimentos circulares

Um dos primeiros estudos usando um dispositivo de posicionamento bidimensional – a mesa digitalizadora[1] – foi o trabalho de Evans [Evan81]. Neste trabalho, destaca-se principalmente o desenvolvimento de um dispositivo lógico, o *trackball 3D*, que procura simular o movimento de um dispositivo *trackball* em três dimensões a partir de um dispositivo de duas dimensões. Para isto, os movimentos horizontais 2D (x') são mapeados para movimentos em x no espaço 3D, movimentos verticais 2D (y') passam a ser entendidos como movimentos em y no espaço 3D e movimentos circulares (no sentido horário e anti-horário) são compreendidos como movimentos em z (avançando ou retrocedendo na direção de profundidade, respectivamente) (figura 2.1).

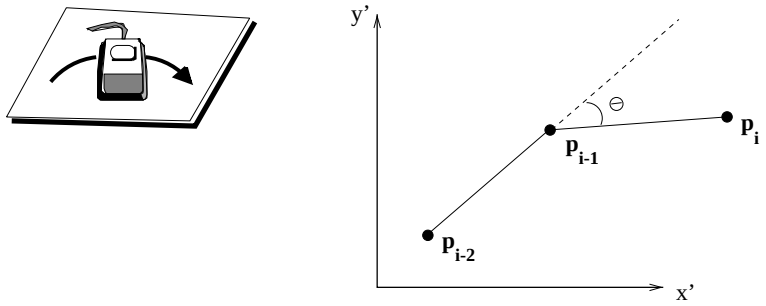

Figura 2.1: Curvatura do movimento do usuário em diferentes instantes de tempo.

A inconveniência deste método de mapeamento é que, independentemente do movimento do usuário, linear (em x e y) ou circular (em z), normalmente existirá uma variação (positiva ou negativa) no eixo z, o que nem sempre é desejável; principalmente quando se requer somente movimentos lineares em x ou y.

Inicialmente, a representação ou realimentação visual do cursor para movimentos tridimensionais, usada por Evans e nos primeiros trabalhos (como em AutoCad [Fole90]), foi bidimensional e as manipulações sobre a cena 3D eram consideradas como a composição (em planos bidimensionais) de vistas aéreas, frontais e laterais (figura 2.2). Os trabalhos posteriores demonstraram a importância de apresentar a cena como um todo (a cena num único plano bidimensional) e não como a composição das suas partes [Niel86] [Bier86] [Chen88]. Os fundamentos teóricos desta técnica serão abordados no capítulo 4.

[1] Aplicável também ao dispositivo *mouse*.

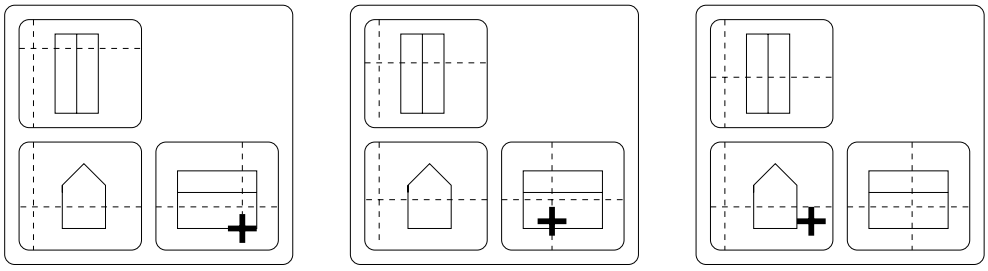

Figura 2.2: Vistas ortogonais bidimensionais.

Uso do particionamento do espaço da entrada

Nielson e Olsen [Niel86] propuseram em 1986 uma técnica mais intuitiva para posicionamento e manipulação de objetos no espaço tridimensional. Como visto na figura 2.3, o espaço bidimensional de entrada é particionado de forma a obter regiões correspondentes entre os movimentos do dispositivo de posicionamento bidimensional (*mouse*) e as componentes x, y, z do espaço tridimensional. Esta técnica de mapeamento considera como base de raciocínio a imagem projetada de um cursor tridimensional composta de três eixos ortogonais. Um deslocamento do *mouse* sobre o plano (espaço de entrada – $x'y'$) determina um vetor de deslocamento do *mouse* entre a sua posição anterior e a nova (movimentos relativos do *mouse*). Como o cursor tridimensional nada mais é do que uma realimentação visual dos movimentos do usuário no espaço de entrada, a nova posição do cursor tridimensional projetado na tela deve corresponder à nova posição do *mouse* no espaço de entrada. Esta associação entre as projeções dos eixos do cursor 3D e o vetor de deslocamento do *mouse* permite estimar o deslocamento efetivo do cursor no espaço 3D.

O algoritmo está baseado na projeção do cursor tridimensional sobre o plano da tela de visualização ($x'y'$). Como na projeção parte da informação é perdida, o método é impreciso na obtenção de posições 3D a partir do *mouse*.

Apesar desta inconveniência, o método é bastante intuitivo e flexível, pois podem ser desenvolvidas diferentes representações do cursor 3D (figura 2.4).

Na figura 2.5 destaca-se o uso do cursor “espaço cheio” no contexto de uma cena.

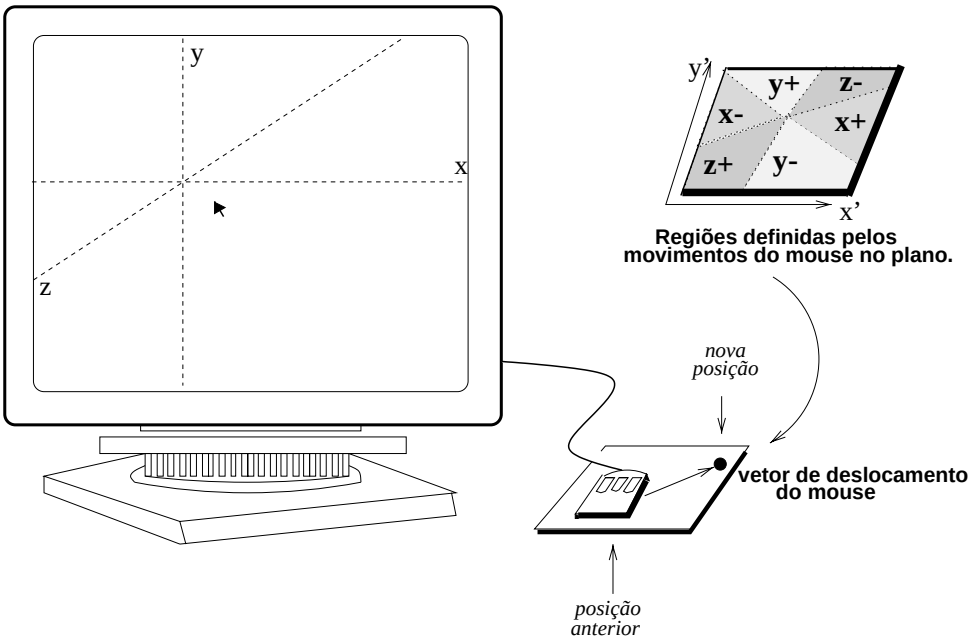

Figura 2.3: Correspondência entre movimentos do dispositivo e as coordenadas do espaço 3D.

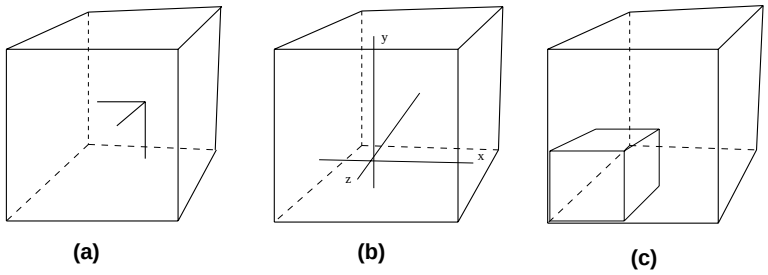

Figura 2.4: Representações gráficas do cursor 3D: (a) tríade (*triad*); (b) espaço cheio (*full space*) e (c) cúbico (*cubic*).

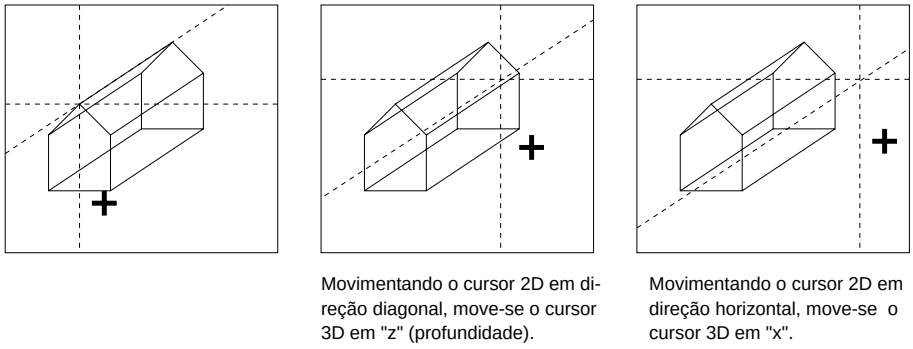

Figura 2.5: O cursor tridimensional de Nielson e Olsen.

Os fundametos teóricos desta técnica de particionamento serão abordados no capítulo 4.

Uso de movimentos em diferentes planos da cena

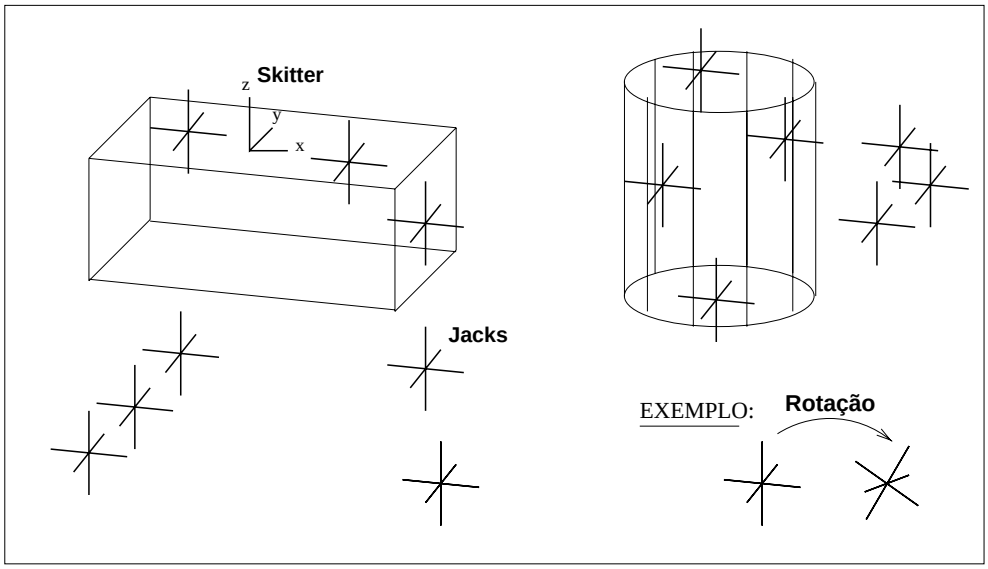

Figura 2.6: O cursor *skitter* e os *jacks* para transformações tridimensionais.

Bier [Bier86][Bier89][Bier90] elaborou um projeto para manipulação interativa de objetos tridimensionais usando como dispositivo de entrada o *mouse*. A manipulação de objetos 3D foi baseada em elementos da própria cena, tais como o uso da aresta de um objeto. Para Bier, a manipulação de objetos consiste dos seguintes passos:

- a seleção dos objetos a serem movimentados;
- a escolha da transformação;
- a especificação dos parâmetros de transformação.

Bier propõe o uso de sistemas de coordenadas cartesianas chamados *Jacks* para servirem como elementos bases para transformações 3D e também como referências para especificar os parâmetros de transformações 3D. Por exemplo, a transformação de rotação de um objeto 3D pode ser especificada através da rotação do *jack*. Para posicionamento interativo dos *Jacks* na cena usa-se um cursor tridimensional denominado *skitter* (figura 2.6), podendo ser:

- colocado sobre uma aresta, um vértice ou uma face especificada;
- posicionado sobre as superfícies dos objetos ou no centro dos objetos;
- posicionado sobre qualquer plano do espaço 3D.

A realimentação visual do *skitter* é um cursor com três eixos ortogonais. Um dos problemas encontrados no uso do *skitter* é que o seu deslocamento no espaço tridimensional acontece somente sobre os eixos que definem os três planos ortogonais. Ou melhor, o movimento do *skitter* ocorre sobre a direção dos seus eixos.

Uso de controladores virtuais

O trabalho de Chen [Chen88] é especialmente orientado ao problema de movimentos rotacionais no espaço 3D. A estratégia usada nesta técnica de mapeamento tem sido a introdução de controladores virtuais manipulados por um dispositivo de posicionamento bidimensional. Ele introduz novas formas de rotações de objetos tridimensionais através do uso de quatro controladores virtuais.

Na figura 2.7(a) tem-se um controlador que usa "deslizadores gráficos" (*sliders*) para realizar rotações em torno dos eixos x, y e z. O uso de deslizadores na tela permite ao usuário um melhor controle no ângulo de rotação em torno do eixo x, y ou z aplicado sobre o objeto. Em 2.7(b) tem-se uma variante na qual os deslizadores são sobrepostos à

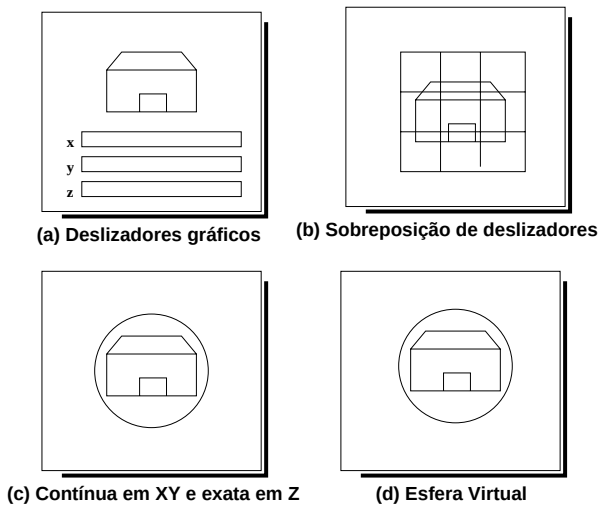

Figura 2.7: Controladores virtuais com o objeto no centro da tela.

imagem do objeto (*overlapping sliders*). Os movimentos em x, y e z foram representados por movimentos verticais, horizontais e circulares do *mouse*, respectivamente.

Em 2.7(c) usa-se um tipo de controlador denominado contínuo em XY e exato em Z. Contínuo em x e y porque um movimento de arrasto (*drag*) dentro do círculo deslocando o *mouse* da esquerda para a direita e de cima para baixo permitirá a rotação do objeto em torno dos eixos x e y, respectivamente. E movimentos de arrasto na diagonal, permitirão uma rotação proporcional sobre os eixos x e y. E exato em z porque um movimento de arrasto fora do círculo, rodará o objeto todo no sentido horário ou anti-horário dependendo do sentido do movimento, fazendo o objeto girar em torno do eixo z.

Em 2.7(d) o controlador usado é uma "esfera virtual" que simula os movimentos de um *trackball 3D*. O usuário pode imaginar ver o objeto dentro de uma casca de vidro em que movimentos verticais do *mouse* dentro do círculo provocam rotações do objeto em torno do eixo x, movimentos horizontais do *mouse* provocam rotações em torno do eixo y e movimentos circulares ao longo da esfera ou fora dela provocam movimentos em z.

Estes controladores passaram a ter cada vez mais aceitação na área de interação visual e foram aperfeiçoados a ponto de permitirem a construção e manipulação direta de objetos no espaço tridimensional [Conn92][Hern92][Snib92][Zele93].

Uso do sistema de referência do objeto

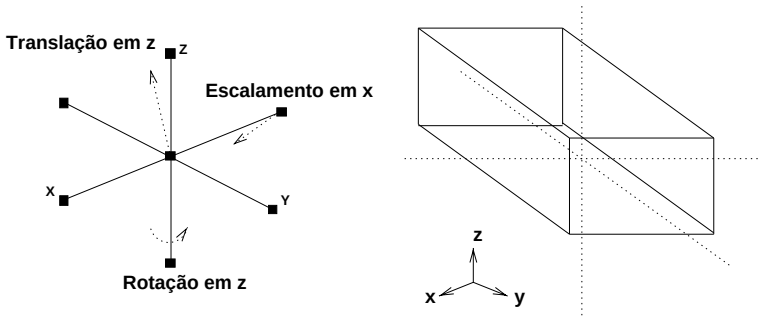

Figura 2.8: Manipulação direta baseado no sistema de coordenadas do objeto.

O mecanismo de interação com objetos 3D através de um *mouse* apresentado por Emmerik [Emme90] permite a definição de parâmetros de três tipos de transformação 3D: rotação, translação e escalamento. A manipulação de movimentos é baseada no sistema de coordenadas de referência do objeto. 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) (figura 2.8).

O usuário deve selecionar um ponto de controle e arrastá-lo (*drag*). O ponto de controle escolhido, a direção e a origem do movimento do *mouse* determinam os parâmetros da transformação. Por exemplo, para especificar um movimento de translação, o ponto de controle na origem do sistema é arrastado paralelamente a este eixo. Para rotação do objeto em torno de um eixo, o ponto de controle é escolhido em um dos dois eixos e arrastado em direção paralela ao terceiro.

2.1.2 Solução por *hardware*: Novos dispositivos de posicionamento

Esta seção apresenta uma amostra dos novos dispositivos de posicionamento 3D.

O dispositivo *flyer mouse*

Para Colin [Ware88], posicionar um objeto no espaço tridimensional é uma operação que requer seis variáveis de entrada: três para especificar posicionamento e três para espe-

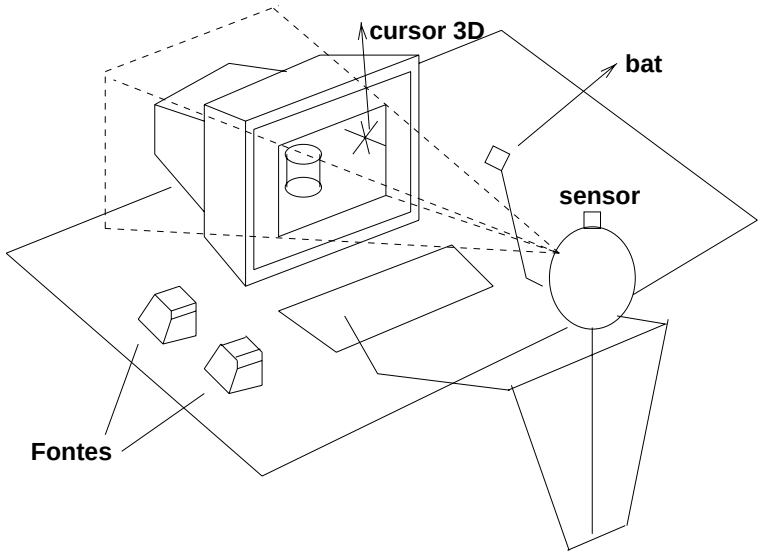

Figura 2.9: Realimentação visual aprimorada por sensor no corpo do usuário.

cificar orientação. Um objeto pode ser transladado na direção dos eixos x, y e z e pode ser rotacionado em torno dos três eixos para mudar sua orientação. O posicionamento de um objeto no espaço tridimensional com uma única interação exige um dispositivo que permita a entrada de seis variáveis.

Para seu propósito, Colin cria o *flyer mouse* ou *bat*, uma extensão lógica do *mouse* convencional. Este dispositivo codifica posições relativas, como o *mouse* convencional, mas capta a entrada de seis variáveis para manipulação do objeto. Logo, a manipulação do objeto requer de uma interface sensora que se valha não só das seis variáveis, como também de um protocolo de interação que funciona como uma espécie de "comunicador da interação".

Entre os modos de interação providos pelo dispositivo, constam os modos genéricos, especiais e o de posicionamento básico. No caso, por exemplo, de um posicionamento básico, um cursor na tela de visualização exibe as posições x, y e z do *bat*. O objeto pode ser movido quando o cursor é posicionado sobre ele e é pressionado o botão do *bat*. A partir daí, o movimento do *bat* determina uma nova posição e orientação.

Devido à sua baixa resolução, o *bat* não é apropriado para posicionamentos precisos. Porém, este não é um problema se considerarmos a instabilidade dos movimentos da mão. Para casos em que movimentos precisos são requeridos pode-se, alternativamente, mudar

o mapeamento de movimentos da mão para movimentos do objeto ou fazer ajustes na sensibilidade do cursor. Também é importante comentar que o *bat* permite movimentos cinestéticos[2].

O *bat* possui um sistema de realimentação visual que acrescenta maior realidade à interação com o ambiente tridimensional. Sofisticações adicionais foram realizadas permitindo uma melhor exploração e controle da câmera virtual através da inclusão de técnicas de visualização conhecidas como "metáforas de interação" [Ware90]. O *hardware* associado a estas técnicas é um sensor que é colocado sobre a cabeça do usuário para corrigir interativamente a visualização da cena a partir das mudanças de posição do usuário (figura 2.9).

Desenvolvimentos posteriores usando o *bat* produziram um sistema altamente interativo (JDCAD) [Jian94] que dispõe de menus tridimensionais, técnicas de seleção de objetos e outros recursos significativos. Com isso, o modelamento de objetos é realizado em um tempo muito mais curto do que pelos sistemas tradicionais.

O dispositivo *roller mouse*

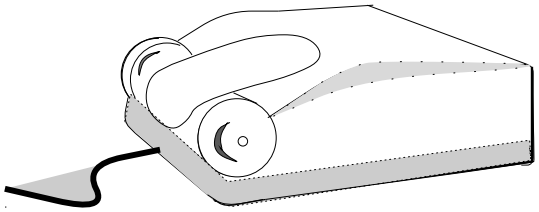

Figura 2.10: O *roller mouse*

O *roller mouse* ou *mouse 3D*, que possui um mecanismo de manipulação direta, foi desenvolvido por Venolia [Veno93] e é baseado num esquema de construção similar ao do *mouse* convencional. A diferença é que o *hardware* do novo *mouse* consiste de uma esfera para codificação dos dados de entrada (situada na base do dispositivo), um botão para interações e duas rodas laterais. A realimentação visual do *roller mouse* é representada por um cursor tridimensional em formato de um cone sólido (numa imagem com *rendering*). Este dispositivo, que controla o movimento do cursor, possui dois tipos de movimento:

[2]Correspondência entre os movimentos do dispositivo com os movimentos do cursor no espaço tridimensional.

movimento do corpo do *mouse* e movimento das rodas do *mouse*. O primeiro movimento controla o cursor tridimensional nas direções vertical e horizontal e o segundo movimento permite a aproximação e afastamento do cursor na direção de profundidade. Existem duas formas de mapeamento:

- mapeamento orientado à câmera;
- mapeamento orientado à tela.

No primeiro, as rodas são mapeadas para o vetor perpendicular ao plano de projeção da cena, e os vetores vertical e horizontal do *mouse* são mapeados para os eixos x e y do plano de projeção da cena. No segundo, posiciona-se o cursor 3D sobre um pixel da imagem e, movimentando o corpo e rodas do *mouse 3D*, controla-se a profundidade do pixel.

Auxiliado visualmente pelo cursor tridimensional, o usuário pode selecionar objetos e realizar operações sobre eles. Uma vez selecionado o objeto, o sistema permite rotacioná-lo e transladá-lo no espaço tridimensional. O usuário pode movimentar o objeto nas três dimensões pressionando o botão do *mouse* e movimentando suas rodas e corpo.

Um mecanismo de interação tridimensional associado ao cursor é o *snap-to*, um método para atração ou captura de objetos no espaço. O seu funcionamento é análogo a um atrator magnético: a força de atração é inversamente proporcional à distância entre o objeto e o cursor. As formas de atração apresentam-se ao usuário nas suas diferentes combinações de face, aresta e vértice.

Vários autores – como Venolia [Veno93], Slater [Slat92] e Zhai [Zhai94] – consideram que a representação visual do cursor no contexto de modelamento geométrico não necessariamente deve ser a mesma que para o contexto de *rendering*. O importante é que a representação visual do cursor tridimensional seja harmoniosa com a imagem do contexto. Um exemplo disso é o tipo de representação visual construída para o *roller mouse*: um cursor "cônico sólido" tridimensional. Outro exemplo é um cursor tridimensional "poliédrico sólido" com "face sensitiva" proposto por Slater. Um terceiro exemplo é um cursor chamado *silk*, projetado por Zhai, que é um tipo de cursor transparente com volume tridimensional utilizado para tarefas de aquisição de objetos no espaço 3D. Todos estes são exemplos de cursores tridimensionais apropriados para imagens com *rendering*.

2.2 Saída

2.2.1 Solução por *software*: Técnicas de visualização

Embora cada uma das técnicas de interação 3D anteriormente vistas (como por exemplo as técnicas de Evans [Evan81], Nielson [Niel86] e Bier [Bier90]) implementem as suas próprias estratégias para visualizar objetos 3D, elas não se têm orientado para o estudo de recursos para manipulação da câmera e realce do espaço tridimensional. Considera-se que a adição destas técnicas de visualização às técnicas de mapeamento da seção anterior (seção 2.1) pode tornar mais real o ambiente de interações 3D. Nesta seção, apresentam-se algumas técnicas de visualização e a sua aplicação em trabalhos de interações 3D. Entre as técnicas a serem vistas têm-se:

- técnicas para realce visual do ambiente 3D e;
- técnicas para representações gráficas do cursor 3D.

Uso de técnicas para realce visual

Phillips [Phil92] descreve uma técnica para melhorar o processo de manipulação direta pelo deslocamento automático e adequado da câmera virtual. Muitas das melhores técnicas para manipulação direta, sensitivas ao ângulo de visualização[3], não usam nenhum mecanismo que permita a manipulação da visualização. Estas técnicas podem ser aprimoradas, permitindo que o usuário coordene o deslocamento do ângulo de observação da cena 3D durante o processo de manipulação. Em alguns casos este processo é automatizado. Isto significa que o sistema pode por si só evitar situações de degeneração nas quais translações e rotações são difícieis de realizar devido à inconveniência do ângulo de visualização da câmera. Esta técnica usa uma metáfora visual[4] chamada "câmera na mão" que manipula a câmera virtual sobre a cena 3D. O espaço da câmera pode ser visto como contendo a cena de objetos tridimensionais e parâmetros de visualização, entre os quais está incluído o observador. A metáfora consiste em imaginar a cena posicionada sobre a mão do observador.

[3] Por exemplo as descritas por: Evans [Evan81], Nielson [Niel86] e Chen [Chen88] (seção 2.1.1).
[4] Desenvolvida por Ware em 1990 [Ware90] para usá-la com o dispositivo *bat* descrito na seção 2.1.2.

A cena é movida enquanto o botão do *mouse* estiver pressionado e pára quando o botão for liberado.

Num outro trabalho realizado por Staples [Stap93] propõe-se um estudo para enriquecimento do vocabulário visual das interfaces tridimensionais e das formas de representação da profundidade espacial nestas interfaces. O estudo sugere que a escolha de recursos como perspectiva, cores, luzes, variação do claro-escuro, opacacidade, transparência e sombras não devem ser aleatórios. É o caso, por exemplo, do estudo de Hagen [Hage91], citado inclusive neste trabalho, que propõe a criação de imagens realistas baseado nas regras convencionais da perspectiva.

Uso de realimentadores visuais

Em 1992, Osborn [Osbo92] realizou um trabalho para visualização e construção de objetos tridimensionais a partir de vistas bidimensionais do objeto. No trabalho, manipulam-se objetos 3D de modo a oferecer ao usuário diferentes ângulos de visualização e definem-se vistas aéreas, frontais e laterais do objeto. O trabalho é uma aplicação prática e melhorada das técnicas de mapeamento usadas por Chen (seção 2.1.1). Neste trabalho, usam-se dois dos quatro controladores virtuais descritos por Chen: (1) "contínuo em XY e exato em Z" e (2) a "esfera virtual". Dizemos que a técnica foi melhorada porque, ao invés de ser usado um círculo para uso da primeira e da segunda técnica, o usuário usa as fronteiras do próprio objeto a ser manipulado. As bordas do objeto atuam como controladores do movimento rotacional. Além disso, o que desta técnica queremos destacar é o uso de diferentes formatos bidimensionais para representação do cursor 2D de acordo com o comportamento do *software*.

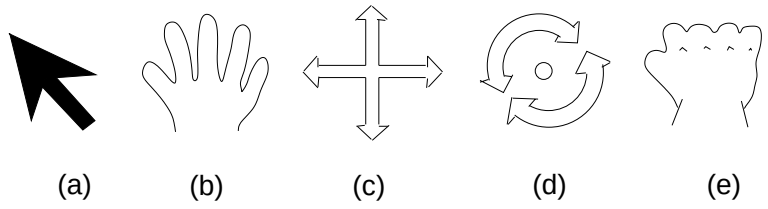

Figura 2.11: Cursores bidimensionais como realimentadores visuais do tipo de interação.

Na figura 2.11(a) apresenta-se um cursor do tipo *arrow* que é usado para seleção, por

exemplo, de menus, botões e deslizadores. Em 2.11(b) apresenta-se o formato de cursor *hand* o qual é usado para indicar a possibilidade de manipulação direta. Em 2.11(c) apresenta-se um tipo de cursor *crossing arrows* para quando a manipulação direta é inicializada e em 2.11(d) mostra-se o formato para o qual muda o cursor – formato *spining-arrows* – quando se está manipulando o objeto em torno dos eixos x e y. Em 2.11(e) apresenta-se um formato *grabbing hand* que é usado quando o objeto é rotacionado em torno do eixo z.

Baseado no mesmo princípio de usar realimentadores visuais para destaque da interação, Houde [Houd92] realiza um trabalho fundamentado no uso de diferentes posições das mãos. Neste trabalho, as mãos são usadas como cursores bidimensionais para facilitar e destacar o tipo de manipulação efetuado numa interface. Houde considera que a associação destes cursores a técnicas de mapeamento 3D permitem a implementação de interfaces gráficas bastante intuitivas.

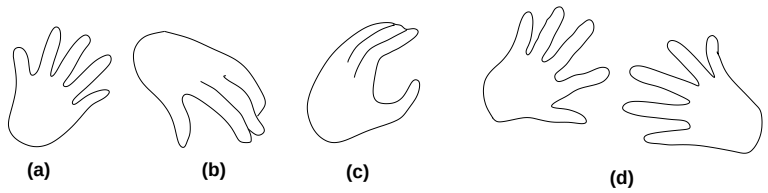

Figura 2.12: Uso das mãos como cursores bidimensionais para interações 3D.

Na figura 2.12(a) uma posição de uma mão aberta indica "estado de pronto para uma interação". Em 2.12(b) mostra-se uma mão em ato de "pressionando algum objeto perpendicular a um plano" indicando assim a tendência de agarrar o objeto. Em 2.12(c) apresenta-se a mão em posição de "pegar", para indicar que um objeto esta sendo levantado. Em 2.12(d) duas mãos, uma frente à outra, indicam rotação do objeto.

2.2.2 Solução por *hardware*: Novos dispositivos de exibição

Display holográfico

Diamand [Diam94] descreve o desenvolvimento de um sistema gerador de figuras tridimensionais que, por meio de tela holográfica[5], permite a observação de tais figuras

[5] A tela holográfica foi desenvolvida pelo Prof. Lunazzi em 1987 e apresentada ao público em 1989 em complemento à descoberta de que a fotografia colorida de alguns hologramas conserva a informação

sem óculos especiais, como no caso de imagens holográficas convencionais. A geração de imagens é controlada por computador e utiliza luz branca e elementos ópticos holográficos para definir a profundidade (coordenada z) de cada ponto da imagem. Trata-se de uma proposta inovadora que apresenta como principal vantagem em relação aos sistemas similares a possibilidade de se alcançarem grandes dimensões de exibição através de projeção.

2.3 Resumo

Neste capítulo foram apresentadas as duas abordagens para solucionar o problema de manipulação no espaço 3D: por *software* e por *hardware*. A abordagem por *software* descreve o desenvolvimento de técnicas de interação e visualização 3D e a abordagem por *hardware* descreve o desenvolvimento de novos dispositivos.

Por motivos já explicados no capitulo 1 (tais como popularidade e baixo custo do dispositivo *mouse* e pouca exploração das técnicas baseadas neste dispositivo [Conn92]), este capítulo deu uma ênfase maior à solução por *software*, deixando a abordagem por *hardware* como complemento para a compreensão do problema. A tabela 2.13 apresenta o resumo das técnicas, acreditando que um estudo generalizado de cada uma delas pode permitir um maior alcance das técnicas na manipulação 3D. Os conceitos preliminares para estudo de algumas destas técnicas serão vistos no próximo capítulo.

tridimensional.

Técnicas de Mapeamento ou Interações 3D
Técnica baseada em movimentos circulares do *mouse*: As componentes x e y de uma posicão (x, y, z) são obtidas a partir de um ponto 2D do *mouse* e a componente z é obtida a partir de três pontos de um movimento circular do *mouse* [Evan81].
Técnica baseada no particionamento do espaço de entrada (ou imagem): para as componentes x+, x-, y+, y-, z+, z- do espaço 3D particionam-se as áreas de movimento do *mouse* em 6 regiões de correspondência 3D [Niel86].
Técnica baseada no movimento sobre diferentes planos da cena: permite manipulação direta de objetos 3D. As transformações 3D são desenvolvidas fazendo uso de sistemas de referência (*jack's*). O deslocamento no espaço é através de um cursor 3D (*skitter*). O movimento do *skitter* é sobre planos da cena [Bier86] [Bier90].
Técnicas baseadas no uso de controladores virtuais: • deslizadores gráficos; • deslizadores sobrepostos na cena; • movimentos do *mouse* contínuo em x ey e exato em z; • esfera virtual. Estas técnicas permitem movimentos rotacionais do objeto através do *mouse* [Chen88].
Técnica baseada no sistema de referência do objeto: são definidos pontos virtuais de controle para especificação de transformações 3D [Emme90].
Técnicas de Visualização 3D
Técnicas para realce visual: • deslocamento automático da câmera virtual [Phil92]. • técnicas que exploram recursos tais como perspectiva, cores, luzes, e sombras [Stap93] [Hage91].
Técnicas para representação gráfica de interações 3D: • uso de diferentes representações gráficas para caracterizar uma interação 3D [Osbo92] [Houd92].

Figura 2.13: Resumo das técnicas de interação e visualização 3D.

Capítulo 3

Conceitos Preliminares

" Quem tem sabedoria, busca o conhecimento
pois ele é mais proveitoso do que a prata
e dá mais lucro do que o ouro..."
Provérbios 3:13.

No capítulo anterior foi realizada uma revisão das principais técnicas associadas ao problema de comunicação entre as coordenadas do dispositivo de entrada (x', y') e os pontos do espaço 3D onde se encontram os objetos (x, y, z). Para melhor compreensão do problema e estabelecimento das bases de solução, são abordados neste capítulo:

- um modelo de visualização;
- um modelo para representação de objetos 3D, o SST;
- a descrição de um núcleo de recursos gráficos tridimensionais, o IQL.

A partir do modelo de visualização 3D objetiva-se o delineamento das bases para o estudo da recuperação do espaço tridimensional do mundo real (WC) a partir do espaço bidimensional do dispositivo (DC). Existem vários modelos para computar transformações de visualização 3D. Neste trabalho, aborda-se especificamente o modelo de visualização adotado no sistema gráfico PHIGS+ (*Programmer's Hierarquical Interactive Graphics System*) [PHIG90] [Fole90]. Serão descritas as etapas e os sistemas de coordenadas, assim como as definições das transformações requeridas para exibição na tela (DC) de uma cena 3D (WC).

Neste capítulo é também apresentada uma visão da estrutura de dados das entidades gráficas para representação de objetos 3D na cena, o SST (*Segment STorage*). Do mesmo modo existem vários modelos para representar objetos 3D. O SST utilizado neste trabalho foi inspirado na organização do modelo CSS (*Central Segment Storage*) também do PHIGS+. Neste modelo dá-se ênfase às vantagens da sua organização e à capacidade de manipular individual ou conjuntamente entidades 3D da cena.

Finalmente, será descrito o módulo IQL, destacando-se o reaproveitamento de alguns recursos tridimensionais para o desenvolvimento das técnicas de interação 3D. Também será apresentada uma visão das atuais bibliotecas de suporte e o uso delas na solução implementada.

3.1 Transformação de visualização

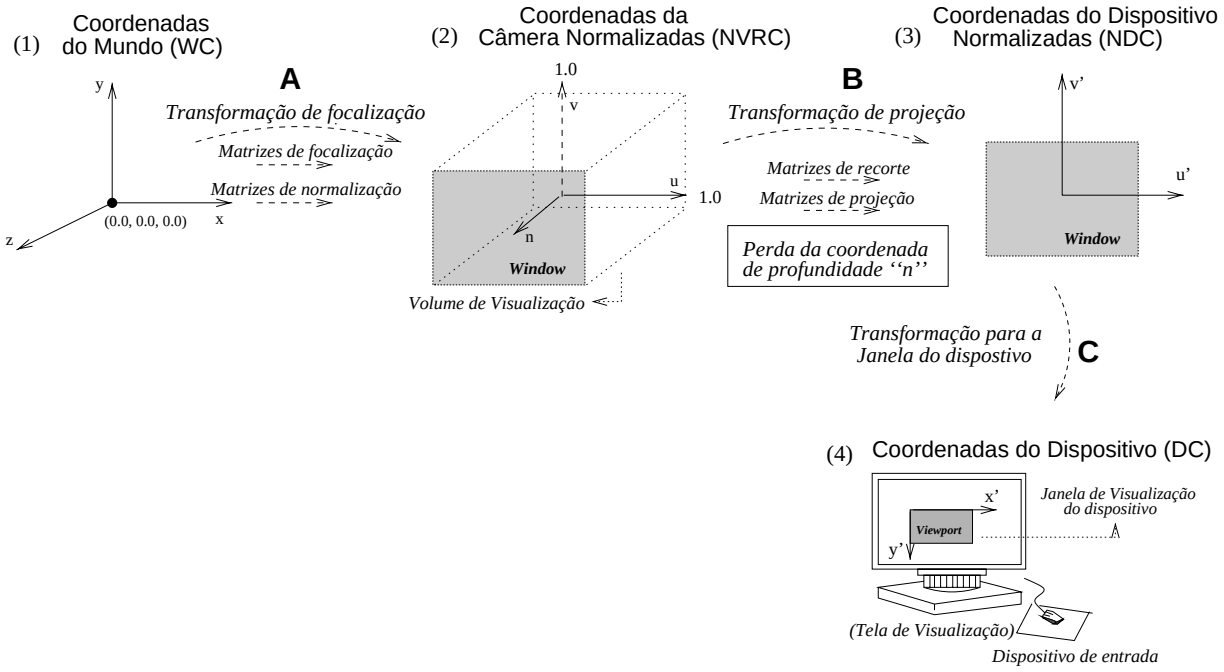

Figura 3.1: Sistema de coordenadas num processo de transformação de visualização.

As transformações de visualização permitem exibir uma cena 3D projetada sobre monitores 2D (figura 3.1). A sequência de sistemas de coordenadas a seguir tem como

propósito tornar claras as diferentes etapas do processo de visualização[1]:

1. coordenadas 3D do mundo (WC), denotado pelos eixos x, y e z;
2. coordenadas 3D da câmera normalizadas (NVRC), com eixos u, v e n;
3. coordenadas 2D do dispositivo normalizadas (NDC), com eixos u' e v';
4. coordenadas 2D do dispositivo (DC), denotado por x', y'.

A passagem de um sistema de coordenadas para outro implica transformações de:

A. focalização e normalização da cena (WC para NVRC);

B. recorte e projeção da cena (NVRC para NDC) e

C. exibição da cena (NDC para DC).

Focalização e normalização da cena

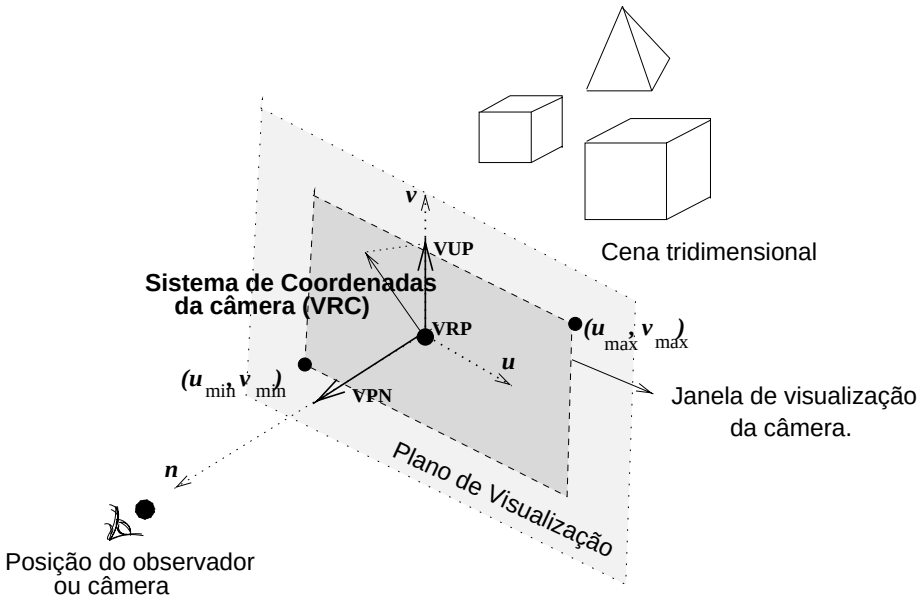

Figura 3.2: Focalização da cena pelo observador ou câmera.

[1] Esta divisão está detalhada no livro *Computer Graphics* - capítulos 5 e 6 [Fole90].

A etapa de focalização da cena consiste em definir um ângulo de visualização da cena através de um observador ou câmera fotográfica. A cena é vista através de um plano de visualização. Este plano, que é o plano de visualização da câmera, é definido por um ponto chamado ponto de referência de visualização (VRP) e por um vetor normal ao plano que passa por este ponto (VPN). Para visualizar a cena é necessário definir uma janela (*window*) no plano de visualização que será mapeada para a janela do dispositivo (*viewport*). Qualquer ponto do espaço WC que seja projetado fora da janela de visualização não será exibido no dispositivo.

Os eixos u, v e n do sistema VRC da câmera são definidos com base no plano de visualização. O ponto de origem do sistema de coordenadas da câmera é o VRP. Um dos eixos, o eixo n, é definido na mesma direção do vetor normal (VNP). O outro, o eixo v, é definido a partir do vetor que indica a inclinação da imagem (VUP). E por último, o eixo u é obtido a partir do produto vetorial de v e n. A janela de visualização (*window*) é especificada a partir dos valores das coordenadas u e v definidos sobre o plano de visualização: $(u_{min}, v_{min}), (u_{max}, v_{max})$ (figura 3.2).

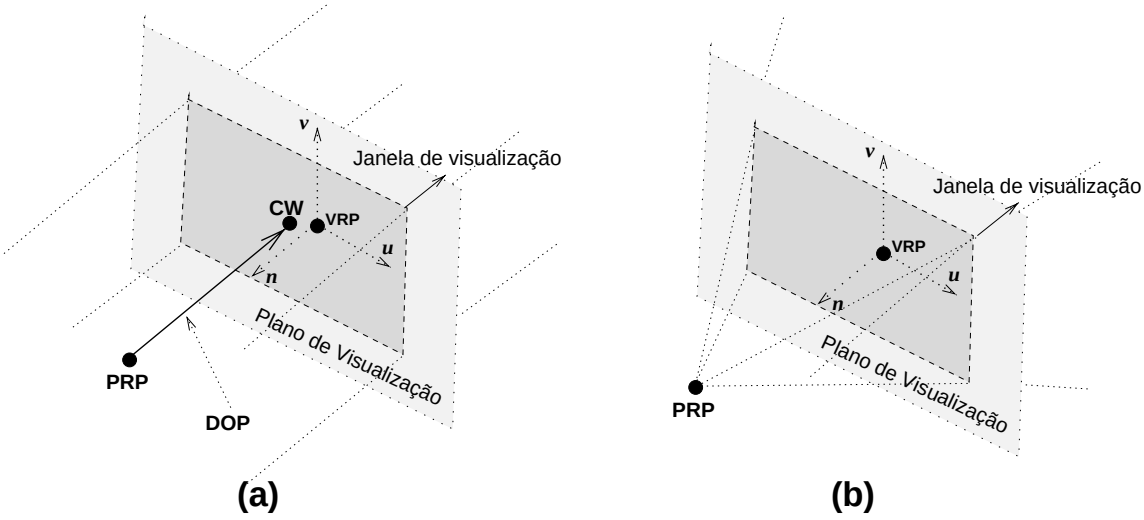

Figura 3.3: Direção de projeção (a) paralela; (b) perspectiva.

Uma projeção é dita paralela se o ponto de projeção dos objetos da cena é no infinito. Neste caso, a direção de projeção da cena sobre o plano (DOP) é definida como o vetor que liga o ponto de projeção (PRP) ao centro da janela (CW) (figura 3.3(a)).

A projeção é considerada perspectiva se o ponto de projeção dos objetos da cena é sobre um ponto finito. Neste caso, o PRP é o centro de projeção (figura 3.3(b)).

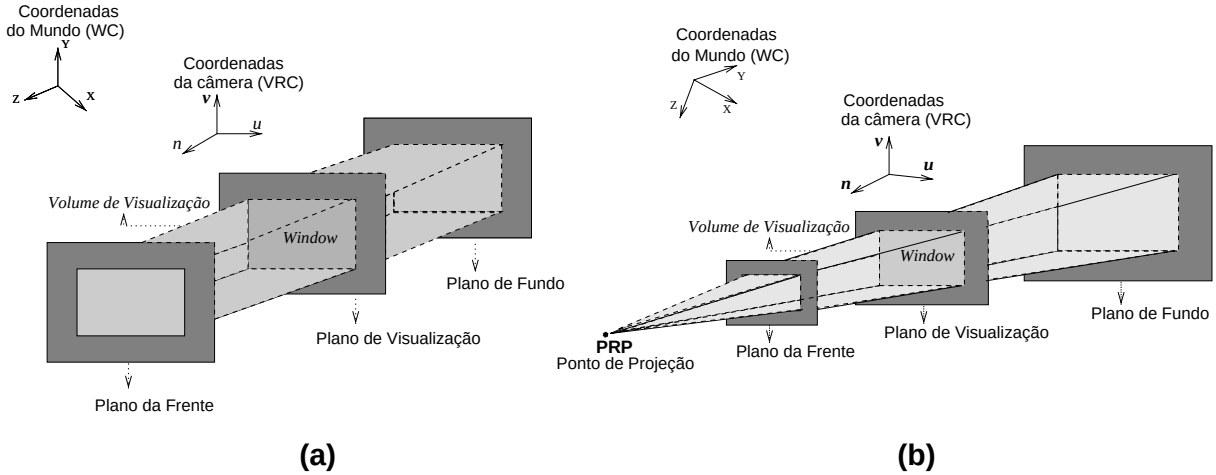

Figura 3.4: Modelo de visualização (a) paralela; (b) perspectiva.

No processo de visualização pode-se tratar somente uma porção finita do mundo (WC) definindo-se um volume de visualização em WC. O volume de visualização, tanto num modelo de projeção paralela como perspectiva, fica delimitado pelos planos de frente, de fundo e laterais. Numa projeção paralela os planos são: $x = -1$, $x = 1$, $y = -1$, $y = 1$, $z = 0$, $z = -1$ e o volume de visualização definido é um paralelepípedo (figura 3.4(a)). Numa projeção perspectiva os planos são: $x = z$, $x = -z$, $y = z$, $y = -z$, $z = -z_{min}$, $z = -1$ e o volume definido é uma pirâmide de base retangular (figura 3.4(b)).

Todo o processo de transformação de focalização e normalização da cena envolve uma sequência de matrizes[2]:

1. translada-se (T_{foc1}) o ponto VRP à origem do sistema WC;
2. rotacionam-se (R_{foc1}) os eixos u, v e n do sistema VRC de tal modo que eles coincidam com os eixos x, y e z do sistema WC, respectivamente;
3. após estas transformações, é necessário fazer uma transformação de cisalhamento (SH_{foc1}) em torno do eixo z de tal forma que a direção de projeção (DOP) seja paralela ao eixo z;

[2][Fole90], pg. 260-267.

4. o volume de visualização "retificado" pelo cisalhamento é então deslocado (T_{foc2}) para a origem do sistema NVRC e normalizado (N_{foc1}) com dimensões $2 \times 2 \times 1$ de forma a obter um volume de visualização canônico.

No primeiro passo ocorre apenas uma translação dada por

$$T_{foc1} : T(x_o, y_o, z_o) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_o & y_o & z_o & 1 \end{bmatrix} \tag{3.1}$$

onde (x_o, y_o, z_o) são as coordenadas de VRP.

No segundo passo usam-se as propriedades das matrizes ortogonais [Fole90]. Esta propriedade permite que o sistema ortogonal u, v e n seja rotacionado de forma que os seus eixos coincidam respectivamente com os eixos x, y e z do sistema de coordenadas WC através da matriz de rotação R_{foc1} (figura 3.5). Os vetores coluna R_x, R_y e R_z da matriz R_{foc1} correspondem aos vetores unitários que ao serem multiplicados por R_{foc1} são transformados em vetores $(1,0,0)$, $(0,1,0)$ e $(0,0,1)$, os vetores no sistema WC.

$$R_{foc1} = \begin{bmatrix} r_{1x} & r_{1y} & r_{1z} & 0 \\ r_{2x} & r_{2y} & r_{2z} & 0 \\ r_{3x} & r_{3y} & r_{3z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

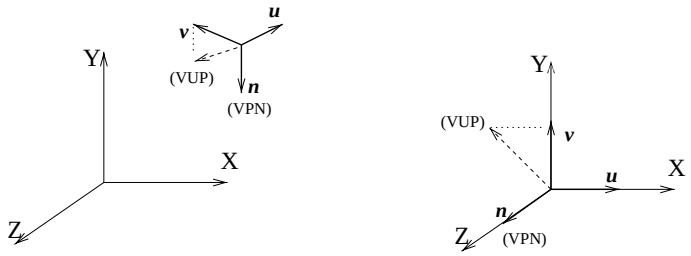

Figura 3.5: Rotação do sistema VRC para o sistema WC.

Portanto, para que VPN coincida com z $R_z = \begin{bmatrix} r_{1z} & r_{2z} & r_{3z} & 0 \end{bmatrix}^T = \frac{VPN}{|VPN|}$

Para que o produto vetorial de VPN e VUP coincida com o eixo x, deve-se ter,

$$R_x = \left[\begin{array}{cccc} r_{1x} & r_{2x} & r_{3x} & 0 \end{array} \right]^T = \frac{VUP \times VPN}{|VUP \times VPN|}$$

E, finalmente, $R_y = \left[\begin{array}{cccc} r_{1y} & r_{2y} & r_{3y} & 0 \end{array} \right]^T = R_z \times R_x$,

é o vetor que deve coincidir com o eixo y após a transformação.

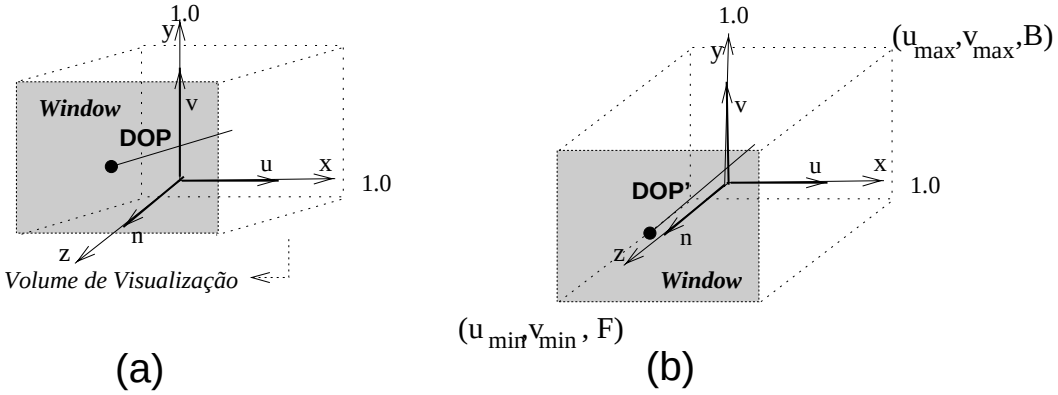

Figura 3.6: Através da matriz de cisalhamento o vetor DOP é retificado para um vetor DOP' paralelo ao eixo z.

O terceiro passo é o cisalhamento do volume de visualização ao longo do eixo n tal que a direção DOP seja paralela com o eixo z (figura 3.6(b)). Através da matriz de cisalhamento o vetor DOP é retificado para o vetor DOP' paralelo ao eixo z. Sejam o centro da janela (CW) e o ponto de projeção (PRP) definidos como,

$$\begin{aligned} CW &= \left[\begin{array}{cccc} \frac{u_{max}+u_{min}}{2} & \frac{v_{max}+v_{min}}{2} & 0 & 1 \end{array} \right] \\ PRP &= \left[\begin{array}{cccc} prp_u & prp_v & prp_n & 1 \end{array} \right]. \end{aligned}$$

então,

$$\begin{aligned} DOP &= \left[\begin{array}{cccc} dop_x & dop_y & dop_z & 1 \end{array} \right] = CW - PRP \\ &= \left[\begin{array}{cccc} \frac{u_{max}+u_{min}}{2} & \frac{v_{max}+v_{min}}{2} & 0 & 1 \end{array} \right] - \left[\begin{array}{cccc} prp_u & prp_v & prp_n & 1 \end{array} \right]. \end{aligned}$$

E, para retificar a direção de projeção, utiliza-se a matriz

$$SH_{foc1} : SH(sh_x, sh_y) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ sh_x & sh_y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

onde sh_x e sh_y devem ser obtidos tal que as componentes $x(dop'_x)$ e $y(dop'_y)$ do vetor DOP' sejam nulas. Assim,

$$\begin{aligned} DOP' = \begin{bmatrix} 0 & 0 & dop_z & 1 \end{bmatrix} &= DOP.SH_{foc1} \\ &= \begin{bmatrix} dop_x & dop_y & dop_z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ sh_x & sh_y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

onde,

$$\begin{aligned} sh_x.dop_z = 0 \qquad & e \qquad sh_x = -\tfrac{dop_x}{dop_z}, \\ sh_y.dop_z = 0 \qquad & e \qquad sh_y = -\tfrac{dop_y}{dop_z}. \end{aligned}$$

Após o cisalhamento as fronteiras do volume são $u_{min} \leq x \leq u_{max}$, $v_{min} \leq y \leq v_{max}$ e $B \leq z \leq F$; onde F e B são, respectivamente, as distâncias dos planos de frente e de fundo do volume, especificadas em relação ao sistema VRC. A partir destes dados pode-se determinar a transformação do sistema VRC para NVRC. Duas transformações básicas são necessárias:

1. translação, $T_{foc2} = T(-\frac{u_{max}+u_{min}}{2}, -\frac{v_{max}+v_{min}}{2}, -F)$ e
2. normalização para um cubo de dimensões $2 \times 2 \times 1$

$$N_{foc1} : N(n_1, n_2, n_3) = \begin{bmatrix} n_1 & 0 & 0 & 0 \\ 0 & n_2 & 0 & 0 \\ 0 & 0 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

onde, $n_1 = \frac{2}{u_{max}-u_{min}}$, $n_2 = \frac{2}{v_{max}-v_{min}}$ e $n_3 = \frac{1}{F-B}$.

Assim, a etapa de focalização e normalização denotada pela letra A (figura 3.1) pode ser representada pelo produto matricial,

$$\boxed{\mathsf{A} = T_{foc1}.R_{foc1}.SH_{foc1}.T_{foc2}.N_{foc1}} \tag{3.4}$$

Recorte e projeção da cena

As etapas que se seguem envolvem transformações de coordenadas normalizadas da câmera (NVRC) para coordenadas normalizadas do dispositivo (NDC). Em primeiro lugar, é necessário fazer um recorte (*clipping*) da cena, ou seja, manter apenas os pontos que estão incluídos na região do volume de visualização. Assim, as coordenadas das entidades visualizáveis estão delimitadas entre os valores $-1 \leq x \leq 1$, $-1 \leq y \leq 1$, $-1 \leq z \leq 0$.

Em segundo lugar, é necessário projetar (P) os pontos do interior do cubo sobre o plano uv para exibição na tela. Para fazer a projeção, basta tornar nula a coordenada n, do espaço NVRC, que representa a profundidade da câmera. Para isto, multiplica-se pela matriz (figura 3.1)

$$\mathsf{B} = P_{uv} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

Exibição da cena

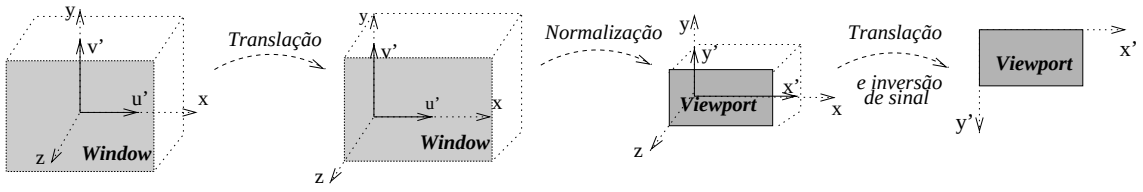

Figura 3.7: Passos para exibição da cena.

A exibição da cena consiste em visualizar a cena projetada em coordenadas normalizadas do dispositivo para uma janela do dispositivo de visualização (*viewport*). Isto significa

mapear a imagem de coordenadas NDC para coordenadas DC da tela[3].

O enquadramento do volume nas novas dimensões da janela pode ser pensado como um processo de três passos (figura 3.7):

1. primeiro é necessário que o volume seja transladado (T_{exi1}) para a origem. Como o volume é canônico, a translação é, $T_{exi1} = T(1, 1, 1)$;
2. no segundo passo o volume é normalizado (N_{exi1}) conforme as dimensões da *viewport*. Assim, $N_{exi1} = (\frac{x_{vmax} - x_{vmin}}{2}, \frac{y_{vmax} - y_{vmin}}{2}, \frac{z_{vmax} - z_{vmin}}{1})$;
3. e o terceiro passo é uma outra translação (T_{exi2}) e mudança de sinal[4] nas coordenadas y. Assim, $T_{exi2} = T(0, -y_{vmax}, 0)$.

Assim, o processo de coordenadas NDC para coordenadas DC, denotado pela letra C (figura 3.1) pode ser representado pelo produto matricial:

$$\boxed{\mathsf{C} = T_{exi1}.N_{exi1}.T_{exi2}} \tag{3.6}$$

A figura 3.8 mostra o resultado visual em cada uma destas etapas. O fluxograma de visualização à direita descreve as transformações necessárias. A cena exibida é uma roda. Na etapa A obtem-se a focalização e normalização da cena a partir de um certo ângulo. Na etapa B recorta-se e projeta-se a cena; e na etapa C mostra-se a imagem final na janela do dispositivo. As equações 3.4, 3.5 e 3.6 são, respectivamente, os produtos de matrizes a serem aplicados em cada etapa.

Na próxima seção será vista a estrutura de dados utilizada para exibição desta cena.

3.2 SST: Uma estrutura de dados para entidades gráficas 3D

O SST [IQL90] é uma estrutura de dados para representação das entidades gráficas dos objetos no espaço WC. Ele foi desenvolvido pelo pelo grupo de Computação Gráfica no

[3]Embora as coordenadas NDC e DC sejam bidimensionais, trabalha-se normalmente num espaço 3D. A terceira dimensão adicionada é considerada uma coordenada abstrata.

[4]É normalmente convencionado que o sistema de coordenadas do dispositivo (DC) seja de mão-esquerda e a sua origem fique no canto superior esquerdo do dispositivo físico.

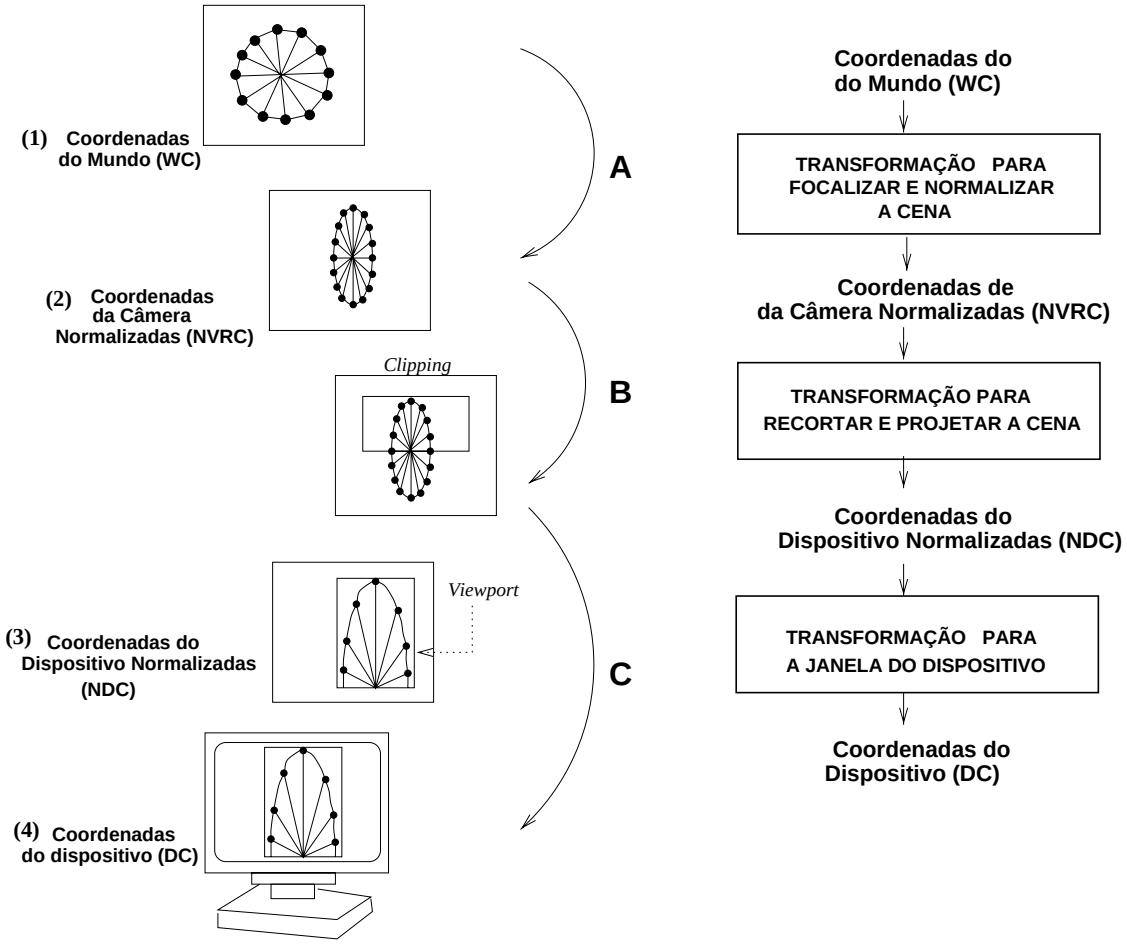

Figura 3.8: Processo de transformação de visualização.

IGD da Sociedade Fraunhofer Gesellschaft (FhG), na Alemanha. A estratégia de modelagem de objetos no SST considera que um modelo para representação de uma cena 3D deve permitir a construção de objetos complexos na cena a partir de entidades gráficas simples[5]. As entidades gráficas básicas no SST são segmentos, arestas e vértices.

De modo a entender um objeto como a composição de entidades mais simples é necessário guardar a informação a respeito dos relacionamentos existentes entre elas. Por exemplo, um cubo só é reconhecido como cubo se suas arestas possuem um relacionamento específico entre si.

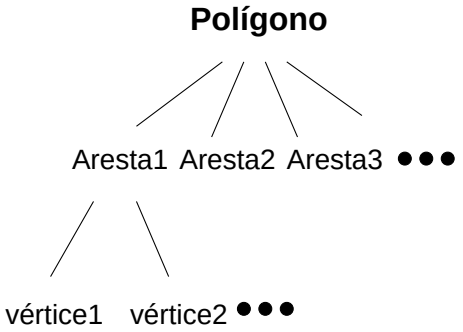

Figura 3.9: Composição da entidade gráfica polígono.

As entidades que o SST considera possuem um relacionamento hierárquico. O segmento é formado de arestas que por sua vez são definidas a partir de vértices. Ao mesmo tempo, um segmento pode fazer parte de um outro segmento, construindo-se assim hierarquias de entidades simples até entidades mais complexas. A figura 3.9 mostra uma árvore representando este relacionamento hierárquico para a definição de um segmento que corresponde a uma entidade gráfica polígono. O SST contém funções que permitem ler, atualizar, copiar e eliminar dados nesta estrutura.

Pela possibilidade de agrupamento das entidades gráficas, o SST permite manipular como um todo um conjunto de entidades gráficas, como por exemplo, mudar a aparência de um objeto ou de dois objetos separados na cena através de uma única operação.

Com o SST também é possível criar níveis de abstração para um determinado objeto.

[5]Entidades gráficas simples são elementos básicos para a composição de objetos gráficos mais complexos a partir delas. No padrão gráfico PHIGS+ as entidades simples são, por exemplo, vértices (*polymark*), arestas (*polyline*), polígono (*fill area*) e polígonos (*fill area set*).

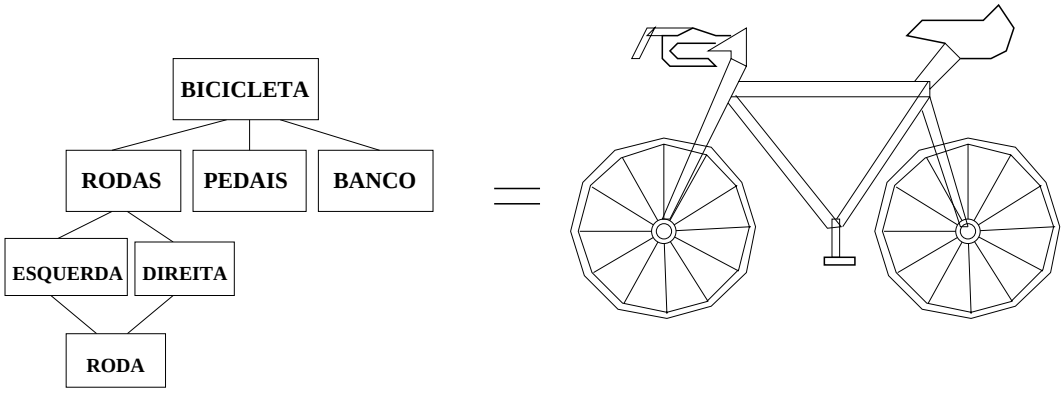

Figura 3.10: Construção de um objeto através de segmentos.

Por exemplo, uma bicicleta pode ser definida como o agrupamento de duas rodas, um guidão, dois pedais e um banco (figura 3.10). Por sua vez, cada uma dessas partes do objeto também pode ser entendida como o agrupamento de entidades simples: uma roda pode ser entendida como um agrupamento hierárquico de arestas e vértices (figura 3.11). Diferentes níveis de abstração para um objeto facilitam interação separada com cada uma das suas partes. Para poder interagir com cada uma dessas partes de forma não ambígua, criou-se internamente códigos de identificação para dar referência às mesmas.

Considerando que um segmento é uma entidade possível de ser duplicada, um objeto (por exemplo, o elemento roda na estrutura da figura 3.10) pode ser duplicado tantas vezes quanto requerido através de um mecanismo chamado mecanismo de referência[6]. Através deste mecanismo, a atualização da informação ocorre de maneira bastante eficiente.

Todas as entidades definidas no SST possuem atributos geométricos e não geométricos, tais como cor, estilo de linha, visibilidade e recorte. Os atributos cor e estilo de linha modificam a aparência visual das entidades. O atributo visibilidade indica a exibição ou não da entidade na tela de visualização. O atributo recorte mostra se a entidade está ou não no volume de visualização.

A seguir será dada uma visão de alguns sistemas de suporte utilizados para o desenvolvimento de interfaces gráficas e exibição das entidades.

[6] Através da cópia física dos dados.

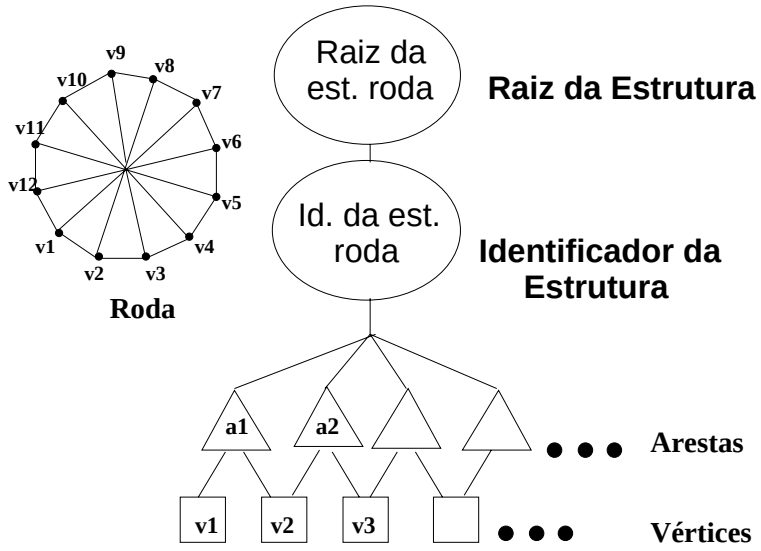

Figura 3.11: Composição hierárquica de arestas e vértices.

3.3 Bibliotecas de sistemas de suporte para desenvolvimento de interfaces gráficas

Segundo Foley [Fole90], existem vários níveis de sistemas de suporte usados atualmente na implementação de interfaces. Entre eles têm-se (figura 3.12):

- sistema gerenciador de janelas;
- bibliotecas de componentes de interface;
- sistema gerenciador de interface de usuário.

De acordo com a figura 3.12, o programa de aplicação acessa todos os níveis de bibliotecas e os programadores podem explorar os serviços que cada nível provê.

Um sistema gerenciador de janelas (*window-manager system*) baseado em modelos de interação básicos, tais como PHIGS e GKS, é basicamente um gerenciador de recursos. Através deste sistema podem ser atribuídos recursos tais como tela e dispositivos de interação. Hix e Hartson [Hix93] complementam

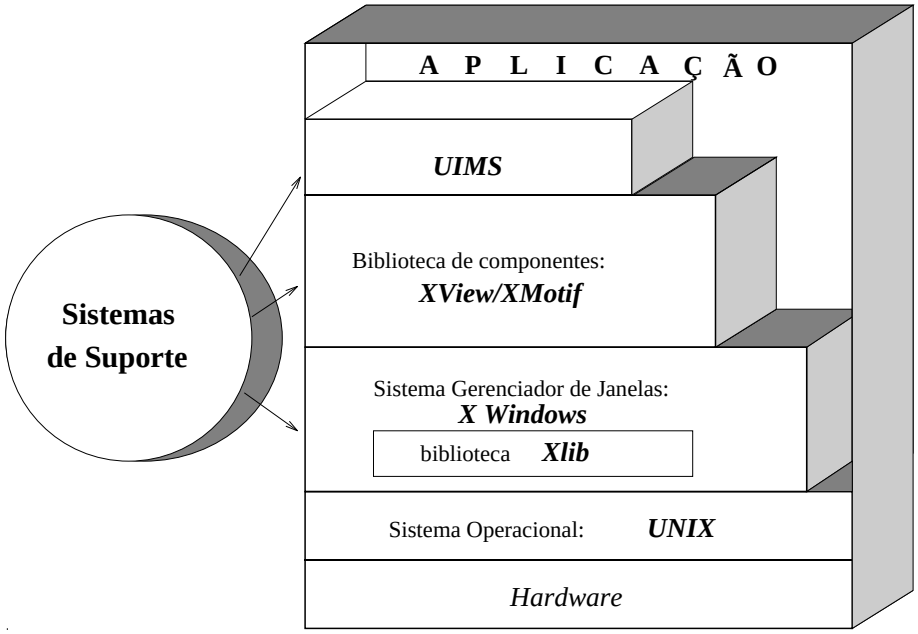

Figura 3.12: Sistemas de suporte para desenvolvimento de interfaces gráficas.

> "através de um sistema como este, um usuário pode interagir com várias aplicações, cada uma numa janela diferente. Cada vez que uma janela é ativada, recursos tais como *mouse* e teclado são atribuídos a essa janela.

A biblioteca de funções de um sistema gerenciador de janelas é composta de dois módulos

1. o módulo gerenciador de janelas e
2. o módulo de sistemas de janelas.

O primeiro módulo permite interações do usuário com as janelas e gerencia os recursos entre elas. O segundo contém as funções gráficas da biblioteca para edição de janelas. O módulo gerenciador é construído com os recursos que a biblioteca gráfica oferece.

No trabalho foi usado o sistema gerenciador de janelas *X-Window* [XWin86] e a biblioteca de funções *Xlib* [Nye90].

Um outro sistema de suporte à construção de interface é a biblioteca de componentes de interface (*toolkit*). Segundo Hix [Hix93], uma biblioteca de componentes para construção

de interfaces de usuário é uma biblioteca de funções (ou rotinas), usadas por programadores, para implementação de interfaces. Esta biblioteca contém códigos de diferentes técnicas de interação que suportam o uso de dispositivos de interação.

Uma componente de interface ou *widget* é um elemento básico que encapsula aparência e funcionalidade de uma técnica de interação e mantém informações sobre objetos gráficos e aplicações. Conner [Conn92] dá um conceito mais exato de componente de interface e a define como:

> "um elemento que encapsula geometria e comportamento para exibição das propriedades e informações dos objetos da aplicação".

O termo geometria refere-se à aparência do componente e o termo comportamento refere-se à sua funcionalidade. Exemplos de componentes de interface 2D são: barras de rolagem (*scroll bars*), janelas de desenho (*canvas*), botões (*buttons*) e cursores (*cursors*). Exemplos de componentes de interface 3D são: esfera virtual (*virtual trackball*) e *rack*.

Como mostrado na figura 3.12, uma biblioteca de componentes pode ser implementada sobre um sistema gerenciador de janelas. Na ausência deste, a biblioteca pode ser implementada diretamente sobre o módulo de funções gráficas[7].

Bibliotecas de componentes de técnicas de interação podem ser usadas não somente por programas de aplicação como também para desenvolvimento do módulo gerenciador de janelas. Usar a mesma biblioteca, em ambos os casos, é um enfoque bastante comum e de importância para unificar a aparência e funcionalidade, tanto dos múltiplos programas de aplicação como do próprio ambiente de janelas. Assim, neste trabalho utilizou-se conjuntamente com o sistema gerenciador *X-Window* a biblioteca de componentes *XView* [Hell90], do padrão OPENLOOK.

Finalmente, tem-se o sistema de suporte para gerenciamento de interface de usuário, chamado também UIMS. Segundo Hix, um UIMS é

> "um conjunto integrado de programas interativos voltados para todo o processo de desenvolvimento de interfaces incluindo projeto, representação, protótipos, execução, avaliação e manutenção de interface".

[7] Uma biblioteca baseada nos recursos da biblioteca *Xlib* é o *Stardust*, desenvolvido no Departamento de Ciências da Computação (DCC) da Unicamp [Furu92].

Para Foley, um UIMS tem como função básica implementar a forma da interface com o usuário e, em alguns casos, também parte de códigos internos. Os UIMS's podem aumentar a produtividade do programador, acelerar o processo de desenvolvimento e facilitar o refinamento iterativo da interface do usuário conforme a experiência adquirida durante o uso. Hierarquicamente, o UIMS está entre um programa de aplicação e uma biblioteca de componentes de interface. Quanto mais poderoso for o UIMS, menor será a necessidade de interação direta entre um programa de aplicação e os recursos do sistema operacional, do sistema gerenciador de janelas e da biblioteca de componentes.

Na próxima seção descreve-se o módulo IQL da biblioteca de suporte PRODIA, contendo recursos básicos desejáveis para o desenvolvimento das técnicas 3D.

3.4 O IQL: um módulo de recursos gráficos

O IQL (*Inquire Quick Line*) [IQL90] é o módulo de recursos gráficos tridimensionais de uma biblioteca de componentes de interface denominada PRODIA [PROD90]. Como o SST, PRODIA foi projetada na década 80 pelo grupo de Computação Gráfica no IGD da Sociedade Fraunhofer Gesellschaft (FhG), na Alemanha. O IQL é um módulo do PRODIA e, num nível mais baixo, é uma biblioteca de funções com alguns recursos gráficos bi e tridimensionais baseada na biblioteca *Xlib* (figura 3.13).

O IQL possui entre outros recursos:

- um modelo de entidades gráficas 3D, denominado SST (seção 3.2);
- um modelo de visualização 3D (seção 3.1);
- alguns realimentadores gráficos bidimensionais (diferentes cursores);
- alguns enfatizadores do contexto gráfico tais como grades 2D com atratores (figura 3.14);
- uma técnica de mapeamento: identificação ordenada (seção 4.4.2);
- algumas ferramentas visuais: grades e eixos 3D (seção 4.5.1).

Tanto o PRODIA quanto os módulos nele incluídos foram descritos em linguagem *C* e as funções da biblioteca *Xlib* foram usadas no desenvolvimento destes módulos. Com o

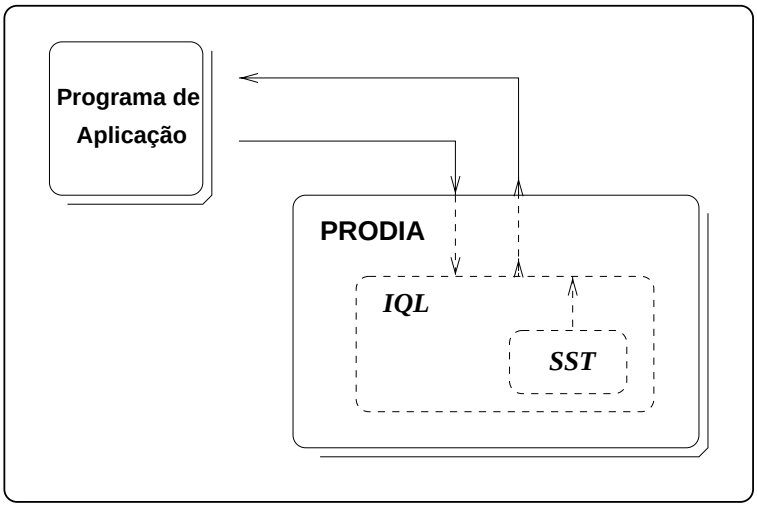

Figura 3.13: Descrição do módulo IQL no contexto do PRODIA. A criação de uma cena 3D no PRODIA requer da descrição da cena e chamada aos recursos através de um programa de aplicação.

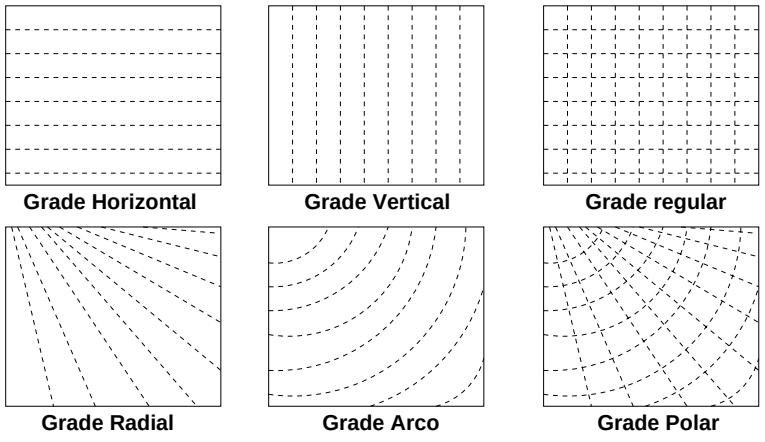

Figura 3.14: Um conjunto de grades 2D.

intuito de reaproveitar ao máximo os códigos de programação, foi efetuado o desacoplamento dos recursos do módulo IQL do PRODIA e as suas funções foram usadas como suporte para o desenvolvimento das técnicas 3D.

3.5 Resumo

Neste capítulo foram ressaltados alguns conceitos básicos para a formalização das técnicas de interação 3D, formando-se as bases para o mapeamento ao espaço 3D (WC) a partir de informações dadas pelo dispositivo (DC).

O modelo para transformação de visualização das entidades gráficas 3D utilizado no trabalho[8] foi baseado no padrão PHIGS+ [Fole90]. O modelo para representação das entidades de objetos 3D foi o SST, e os sistemas de suporte utilizados para implementação das técnicas foram as bibliotecas de funções *Xlib* e de componentes *XView*.

Com o propósito de economizar esforços no desenvolvimento das técnicas e estender o seu alcance para novas bibliotecas de recursos gráficos tridimensionais, foram desacopladas algumas funções do módulo IQL.

Todos estes fatores permitiram o desenvolvimento teórico e implementacional das técnicas propostas no próximo capítulo.

[8] Considerando-se basicamente o modelo de visualização paralela.

Capítulo 4

Uma proposta para a interação 3D através do *mouse* 2D

"Portanto, quer comais, quer bebais,
ou façais outra coisa qualquer,
fazei tudo para a glória de Deus".
I Coríntios 10:31.

4.1 Introdução

No capítulo 1 foi exposto o problema comum à maioria de usuários de sistemas gráficos 3D: conseguir interagir diretamente com um ambiente modelado por computador onde se encontram definidos objetos tridimensionais, usando dispositivos 2D. No capítulo 2 foram descritas sucintamente as diferentes abordagens para solucionar esse problema. E neste capítulo é discutida a solução implementada no contexto deste trabalho. A nossa solução visa atender os seguintes requisitos:

- ser apropriada para estações de trabalho providas de um dispositivo *mouse* 2D e monitor bidimensional;
- ter uma interface de fácil uso e
- ser portátil.

4.2 Problemas

▷ As entidades gráficas 3D de uma cena definida no espaço 3D podem ser projetadas na tela de visualização através de uma projeção paralela ou perspectiva. Devido às transformações serem sobrejetivas, um ponto na tela de visualização pode corresponder a um conjunto de pontos das entidades gráficas 3D originais. Portanto, a distinção das entidades gráficas 3D através de suas imagens é atualmente um trabalho difícil.

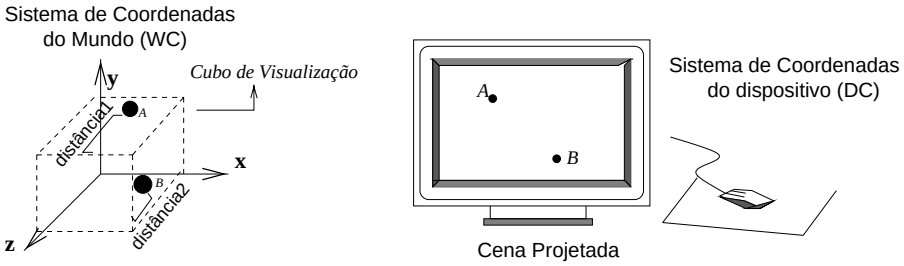

Figura 4.1: Problema da identificação de profundidade.

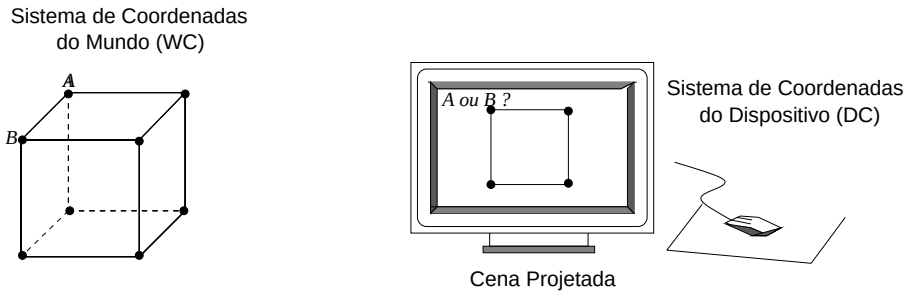

Figura 4.2: Ambiguidade na identificação.

Após a projeção da cena tridimensional sobre o plano da imagem, uma dimensão de cada entidade é perdida. Como se observa na figura 4.1, a partir da tela será impossível ao usuário, por exemplo, determinar se dois pontos A e B projetados em distintas posições da tela estão ou não no mesmo nível de profundidade[1]. Ou, como no caso da figura 4.2, onde se tem novamente os pontos A e B projetados sobre o mesmo ponto da tela, não podendo, neste caso, distinguir o ponto A do ponto B.

[1] Em relação ao plano de projeção.

De forma a garantir uma identificação não ambígua das entidades gráficas 3D, pretende-se estabelecer uma estratégia de mapeamento que permita diferenciar uma ou mais entidades gráficas do espaço 3D que foram projetadas para um mesmo ponto da tela de visualização.

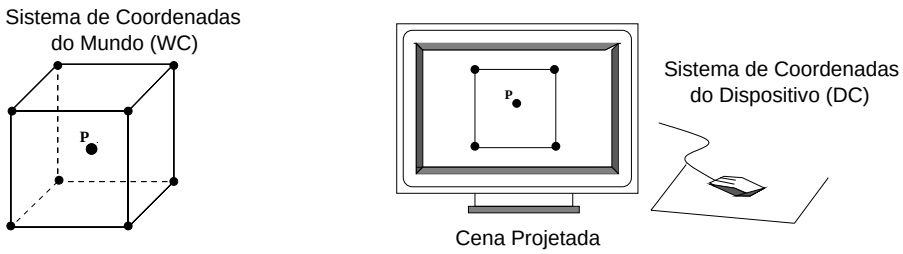

Figura 4.3: Posicionamento de um ponto P no interior de um objeto.

Do mesmo modo, para que o posicionamento de uma entidade gráfica no espaço 3D através do *mouse* seja possível, é necessário o estabelecimento de uma correspondência biunívoca entre as coordenadas do espaço 3D (WC) e as coordenadas do dispositivo (DC). Por exemplo, para posicionar um ponto P no interior de um cubo (figura 4.3) fazendo uso de um *mouse* 2D convencional seria necessário uma técnica especial, uma vez que ele não possui a informação de profundidade.

Estas dificuldades na interação, tanto em tarefas de identificação como de posicionamento, resumem-se a um único problema: a recuperação da coordenada de profundidade da cena, mais especificamente a recuperação da coordenada "n" do espaço NVRC. No caso da identificação, recupera-se a coordenada "n" de entidades já existentes na cena e, no caso do posicionamento, recupera-se a coordenada "n" de posições não necessariamente "ocupadas" por entidades gráficas.

Para ilustrar melhor o problema, a figura 4.4 apresenta a situação de recuperação da coordenada "n" com base num modelo inverso de visualização.

- Um segundo problema a ser considerado na interação com o espaço 3D é a necessidade de manipular estruturas de entidades gráficas 3D mais complexas. Para isto, é necessário que os relacionamentos entre entidades gráficas 3D, tais como pontos, curvas e superfícies, possam ser agrupadas num modelo geométrico mais complexo, de forma a poder manipulá-los como um todo a partir das suas projeções na tela.

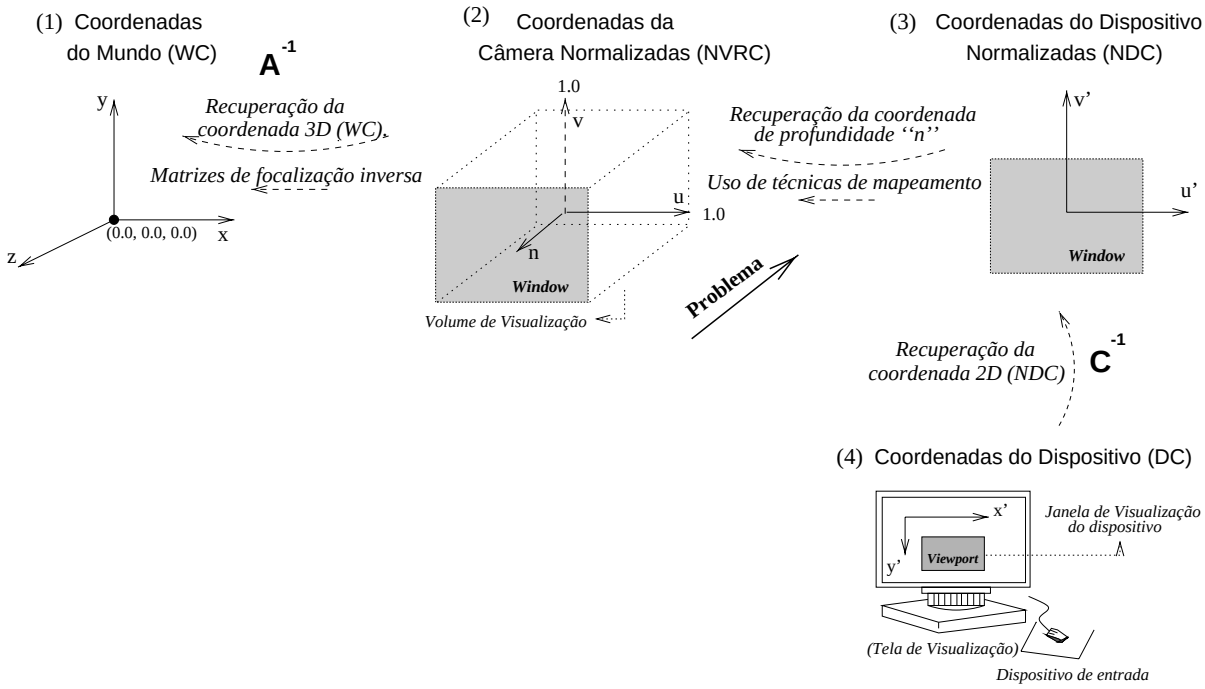

Figura 4.4: Problema de recuperação da coordenada "n".

- Um terceiro problema a ser enfrentado é a dificuldade na percepção do ambiente de interações tridimensionais a partir da tela. É necessário que o ambiente em que tarefas de identificação e posicionamento aconteçam seja compreensível para o usuário.

- Finalmente, um quarto problema latente em nosso meio de pesquisa é que bibliotecas de componentes – tais como *XView* e *XMotif* – são bastante usadas na construção de interfaces tridimensionais apesar de não possuirem recursos tridimensionais. Isto exige dos programadores o desenvolvimento das suas próprias rotinas de interação e visualização 3D. Esta situação sugere a importância de se ter acesso fácil às técnicas que auxiliem na construção de interfaces tridimensionais.

4.3 Uma proposta

- Para o problema de recuperação da profundidade, propõem-se:

- no caso de posicionamento 3D, implementar um conjunto básico de técnicas de interação 3D, apropriado para projeções paralelas:
  - a técnica dos movimentos circulares,
  - a técnica de particionamento do espaço imagem e
  - a técnica do plano de trabalho.

Propõe-se também um estudo detalhado das técnicas, de modo a distinguir as suas etapas de visualização e o contexto em que se pode aplicar cada uma delas.

- no caso de identificação 3D, propõe-se a reutilização de uma técnica desacoplada do PRODIA (seção 3.4):
  - a técnica de identificação ordenada.

Na seção 4.4.2 apresenta-se uma breve explicação do que esta técnica significa.

▷ Para o problema de modelamento das relações entre as entidades gráficas, propõe-se utilizar o modelo de entidades gráficas 3D, denominado SST (descrito na seção 3.2).

▷ Para realce do ambiente 3D, propõem-se algumas estratégias, como câmeras móveis, grades e eixos 3D para orientar melhor os usuários nas suas tarefas de interação 3D.

▷ E para o problema relativo às bibliotecas de componentes usadas para o desenvolvimento das interfaces gráficas tridimensionais, propõe-se a implementação destas técnicas na forma de componentes de interface. Com isto, encaminha-se a solução deste trabalho a uma forma de desenvolvimento reutilizável, fazendo-a integrável a qualquer plataforma.

4.4 Técnicas de mapeamento

4.4.1 Introdução

Através do uso de técnicas de mapeamento e de matrizes inversas de transformações de visualização paralela (seção 3.1), procura-se recuperar a informação da profundidade (mais especificamente, a coordenada "n" do espaço NVRC) que foi perdida durante o processo de projeção da cena para a tela de visualização (DC).

O que se pretende com o uso destas técnicas é que os dados fornecidos pelo dispositivo de entrada (DC) através de movimentos do usuário, isto é, as coordenadas x' e y' do *mouse*, possam ser mapeadas inversamente ao longo das diferentes etapas de visualização até a recuperação das coordenadas x, y e z do espaço real (WC).

Tomando como base o modelo de visualização inverso da figura 4.4, será desenvolvido a seguir um estudo das técnicas.

4.4.2 Técnica da identificação ordenada

Esta técnica permite a identificação de entidades gráficas 3D numa tela de visualização. As entidades correspondentes a todas as entidades da cena são previamente armazenadas tanto em coordenadas do mundo (WC) como em coordenadas do dispostivo (DC) de acordo com o modelo de entidades da seção 3.2, o SST. Fazendo uso deste modelo, o algoritmo de identificação ordena as entidades 3D a serem exibidas na tela de acordo com a sua distância em relação à posição do observador/câmera.

Por exemplo, para identificação de duas entidades projetadas em diferentes posições da tela, (figura 4.1), o usuário posiciona o cursor sobre um determinado *pixel* da entidade projetada (A ou B). Ao posicionar o cursor sobre o *pixel*, o algoritmo de identificação busca na árvore do modelo as coordenadas em WC da entidade gráfica 3D a que pertence esse *pixel*.

Quando se tem o caso em que o cursor está posicionado sobre os pontos A e B projetados sobre um mesmo ponto da tela (figura 4.2), a identificação das entidades é sequencial, conforme a ordem em que foram inicialmente armazenadas no modelo – desde a mais próxima até a mais distante ao observador.

4.4.3 Técnica de movimentos circulares

Esta técnica – baseada na técnica proposta por Evans [Evan81] aplicada inicialmente a movimentos rotacionais no espaço 3D – consiste basicamente num método vetorial para traduzir os movimentos bidimensionais circulares do *mouse* (DC) em movimentos na direção de profundidade da câmera (direção "n" - espaço NVRC).

Nesta técnica a componente x e y de um ponto (x, y, z) no espaço é obtida a partir de uma posição (x', y') do *mouse* e a componente z do ponto é obtida a partir dos movimentos circulares do *mouse*. Estes movimentos (horário e anti-horário) são interpretados como movimentos em "n" (de avanço ou retrocesso na direção de profundidade, respectivamente). O ângulo de rotação (θ) do movimento do usuário é determinado tomando três pontos (x'_{i-1}, y'_{i-1}), (x'_{i-2}, y'_{i-2}) e (x'_i, y'_i) em diferentes instantes no movimento do dispositivo, a partir dos quais obtêm-se dois vetores: $\vec{v_1}$ e $\vec{v_2}$ no espaço NDC (figura 4.5(c)). O movimento no espaço 3D na direção "n" é proporcional ao ângulo θ do movimento circular.

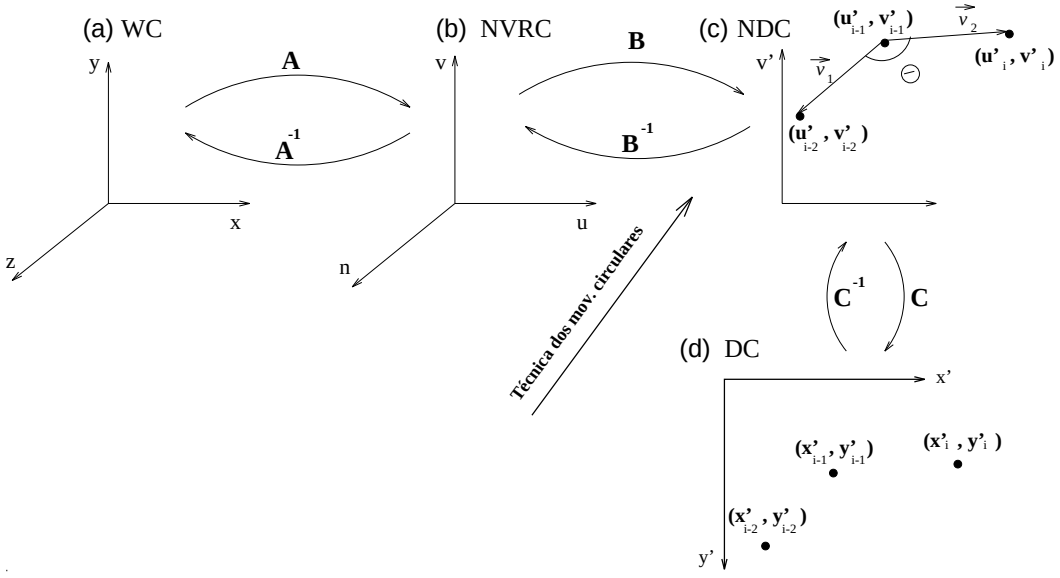

Figura 4.5: Técnica dos movimentos circulares.

A interpretação dos pontos de entrada do dispositivo (DC) em coordenadas NDC é dada pela equação,

$$\begin{bmatrix} u' & v' & 1 \end{bmatrix} = \begin{bmatrix} x' & y' & 1 \end{bmatrix} * C^{-1} \tag{4.1}$$

sendo que,

$$C^{-1} = \begin{bmatrix} C'_{00} & C'_{01} & C'_{02} \\ C'_{10} & C'_{11} & C'_{12} \\ C'_{20} & C'_{21} & C'_{22} \end{bmatrix}. \tag{4.2}$$

Disso resulta que

$$u' = x'C'_{00} + y'C'_{10} + C'_{20} \quad e \tag{4.3}$$

$$v' = x'C'_{01} + y'C'_{11} + C'_{21} \quad . \tag{4.4}$$

Para recuperação da coordenada "n" do espaço NVRC aplica-se a técnica dos movimentos circulares. O produto vetorial entre $\vec{v_1}$ e $\vec{v_2}$ nos dá não somente o módulo, como também o sentido de deslocamento na direção "n" do espaço NVRC. Então,

$$n = \vec{v_1} \times \vec{v_2} = \begin{bmatrix} i & j & k \\ v_{1_{x'}} & v_{1_{y'}} & 0 \\ v_{2_{x'}} & v_{2_{y'}} & 0 \end{bmatrix} = v_{1_{x'}} v_{2_{y'}} - v_{2_{x'}} v_{1_{y'}} \tag{4.5}$$

onde $v_{1_{x'}} = u'_{i-2} - u'_{i-1}$ e $v_{1_{y'}} = v'_{i-2} - v'_{i-1}$; $v_{2_{x'}} = u'_i - u'_{i-1}$ e $v_{2_{y'}} = v'_i - v'_{i-1}$. E, para definição das coordenadas u e v no espaço NVRC definiu-se as coordenadas (u'_i, v'_i) do último ponto de entrada do *mouse*. Como a projeção é paralela, um ponto (u, v, n) em coordenadas NVRC obedece às seguintes relações:

$$u = u'_i \quad e \tag{4.6}$$

$$v = v'_i \quad ; \tag{4.7}$$

substituindo as equações (4.6) e (4.7) nas equações (4.3) e (4.4) este ponto pode ser expresso em termos de coordenadas do dispositivo (DC). Assim,

$$\begin{aligned} u = &\quad x'C'_{00} + y'C'_{10} + C'_{20} \quad , \\ v = &\quad x'C'_{01} + y'C'_{11} + C'_{21} \quad e \\ n = &\quad (u_{i-2} - u_{i-1})(v_i - v_{i-1}) - (u_i - u_{i-1})(v_{i-2} - v_{i-1}). \end{aligned}$$

Ao multiplicar o ponto (u, v, n) pela inversa da matriz de focalização, A^{-1} (equação 3.4, seção 3.1), obtém-se um ponto (x, y, z) em termos de coordenadas do mundo (WC) (figura 4.5). Assim,

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} = \begin{bmatrix} u & v & n & 1 \end{bmatrix} * A^{-1} \quad , \tag{4.8}$$

isto é,

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} = \begin{bmatrix} x'C'_{00} + y'C'_{10} + C'_{20} & x'C'_{01} + y'C'_{11} + C'_{21} & v_{1_{x'}} v_{2_{y'}} - v_{2_{x'}} v_{1_{y'}} & 1 \end{bmatrix} * A^{-1} \tag{4.9}$$

sendo que,

$$A^{-1} = \begin{bmatrix} A'_{00} & A'_{01} & A'_{02} & A'_{03} \\ A'_{10} & A'_{11} & A'_{12} & A'_{13} \\ A'_{20} & A'_{21} & A'_{22} & A'_{23} \\ A'_{30} & A'_{31} & A'_{32} & A'_{33} \end{bmatrix} \tag{4.10}$$

Logo;

$$\begin{aligned} x = \; & x'(A'_{00}C'_{00} + A'_{10}C'_{01}) + y'(A'_{00}C'_{10} + A'_{10}C'_{11}) + A'_{20}(v_{1_{x'}}v_{2_{y'}} - v_{2_{x'}}v_{1_{y'}}) + \\ & (A'_{30} + A'_{00}C'_{20} + A'_{10}C'_{21}) \\ y = \; & x'(A'_{01}C'_{00} + A'_{11}C'_{01}) + y'(A'_{01}C'_{10} + A'_{11}C'_{11}) + A'_{21}(v_{1_{x'}}v_{2_{y'}} - v_{2_{x'}}v_{1_{y'}}) + \\ & (A'_{31} + A'_{01}C'_{20} + A'_{11}C'_{21}) \\ z = \; & x'(A'_{02}C'_{00} + A'_{10}C'_{01}) + y'(A'_{02}C'_{10} + A'_{12}C'_{11}) + A'_{22}(v_{1_{x'}}v_{2_{y'}} - v_{2_{x'}}v_{1_{y'}}) + \\ & (A'_{32} + A'_{02}C'_{20} + A'_{12}C'_{21}) \end{aligned} \tag{4.11}$$

onde x, y e z são as coordenadas de um ponto no espaço WC representadas em coordenadas do dispositivo DC.

4.4.4 Técnica de particionamento do espaço imagem

O particionamento do espaço imagem é uma técnica proposta por Nielson e Olsen [Niel86] que permite interpretar movimentos do *mouse* sobre um plano como movimentos no espaço 3D. Trata-se de uma técnica de interação 3D que requer um mecanismo visual – o cursor 3D – composto de três eixos ortogonais projetados na tela de visualização.

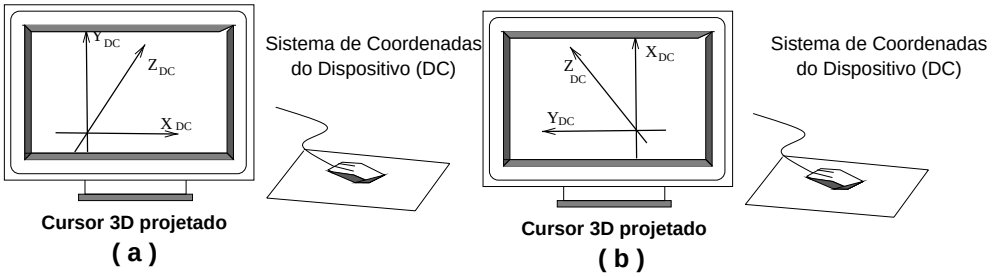

Figura 4.6: Diferentes projeções do cursor 3D na tela de visualização.

Uma vez fixada uma posição do observador em relação à cena tridimensional, a transformação de visualização do cursor 3D na tela de exibição permanece invariante. Nas

figuras 4.6(a) e 4.6(b) tem-se o cursor 3D projetado sob diferentes pontos-de-vista do observador. A projeção do cursor 3D particiona o espaço imagem (tela) em regiões de movimento do *mouse* (DC). Através destas regiões bidimensionais podem ser controlados os movimentos tridimensionais do cursor 3D.

A invariância da transformação de visualização do cursor 3D na tela é a base para o desenvolvimento desta técnica. A figura 4.7 explica o raciocínio desta técnica baseando-se no ponto-de-vista do observador da figura 4.6(a).

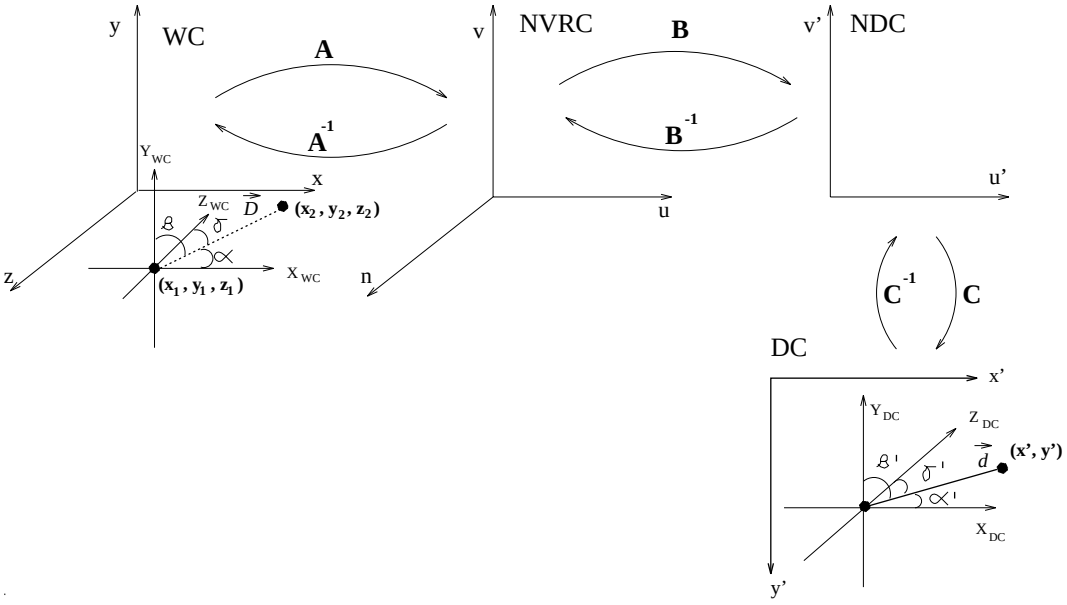

Figura 4.7: Técnica de particionamento do espaço imagem.

Como mostrado na figura 4.7, seja $\vec{d}$ a projeção sobre a tela (coordenadas do dispositivo – DC) de um vetor $\vec{D}$ em três dimensões (WC). Como visto na imagem ampliada da figura 4.8, sejam também α, β e γ os ângulos do no espaço WC que se formam entre os eixos do cursor 3D e o vetor de deslocamento $\vec{D}$. Sejam α', β' e γ' os ângulos no espaço bidimensional (DC) que se formam pela projeção do cursor 3D e do vetor $\vec{D}$ sobre o espaço DC.

O que se quer neste algoritmo é fazer uma boa estimativa do vetor de deslocamento

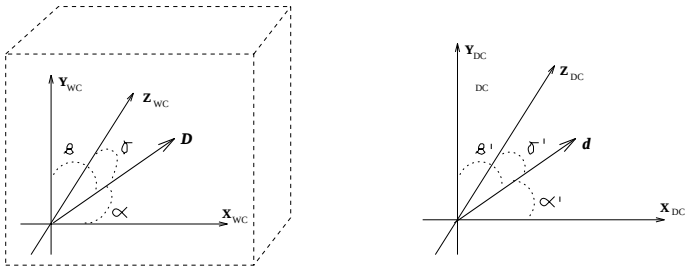

Figura 4.8: Vetor em três dimensões e sua projeção no plano $x'y'$.

$\vec{D}$ do espaço WC em função dos ângulos α', β' e γ'. Considera-se que,

$$\frac{\vec{D}}{|\vec{D}|} = (\cos\alpha, \cos\beta, \cos\gamma) \tag{4.12}$$

tenha a mesma direção do vetor $\vec{d}$. Assim,

$$\frac{\vec{D}}{|\vec{D}|} = \frac{\vec{d}}{|\vec{d}|} = \frac{(\cos\alpha', \cos\beta', \cos\gamma')}{\sqrt{(\cos\alpha')^2 + (\cos\beta')^2 + (\cos\gamma')^2}} \tag{4.13}$$

ou seja,

$$\frac{\vec{D}}{|\vec{D}|} = (\frac{\cos\alpha'}{|\vec{d}|}, \frac{\cos\beta'}{|\vec{d}|}, \frac{\cos\gamma'}{|\vec{d}|}) \tag{4.14}$$

onde, para facilitar futuros cálculos, pode-se considerar $\frac{\cos\alpha'}{|\vec{d}|} = m_x$, $\frac{\cos\beta'}{|\vec{d}|} = m_y$ e $\frac{\cos\gamma'}{|\vec{d}|} = m_z$. A magnitude do vetor $(\vec{D})$ é obtida a partir da nova posição (x', y') do *mouse* (DC) (figura 4.7). Através das equações (4.3) e (4.4) tem-se esta nova posição em coordenadas NDC. Novamente, como a projeção é paralela,

$$u = u' = x'C'_{00} + y'C'_{10} + C'_{20} \quad e \tag{4.15}$$

$$v = v' = x'C'_{01} + y'C'_{11} + C'_{21} \quad . \tag{4.16}$$

Por outro lado, a nova posição (u, v, n) no espaço NVRC pode ser obtida a partir das coordenadas do espaço WC através da matriz de focalização A (equação 3.4, seção 3.1). Portanto,

$$\left[\begin{array}{cccc} u & v & n & 1 \end{array}\right] = \left[\begin{array}{cccc} x_1 + D_x & y_1 + D_y & z_1 + D_z & 1 \end{array}\right] * A \quad , \tag{4.17}$$

onde, da equação (4.14), tem-se,

$$\begin{array}{l} D_x = |\vec{D}|m_x, \\ D_y = |\vec{D}|m_y, \\ D_z = |\vec{D}|m_z, \end{array} \quad e \quad A = \begin{bmatrix} A_{00} & A_{01} & A_{02} & A_{03} \\ A_{10} & A_{11} & A_{12} & A_{13} \\ A_{20} & A_{21} & A_{22} & A_{23} \\ A_{30} & A_{31} & A_{32} & A_{33} \end{bmatrix}. \tag{4.18}$$

Ou seja,

$$u = (x_1 + |\vec{D}|m_x)A_{00} + (y_1 + |\vec{D}|m_y)A_{10} + (z_1 + |\vec{D}|m_z)A_{20} + A_{30} \quad , \tag{4.19}$$

$$v = (x_1 + |\vec{D}|m_x)A_{01} + (y_1 + |\vec{D}|m_y)A_{11} + (z_1 + |\vec{D}|m_z)A_{21} + A_{31} \quad , \tag{4.20}$$

$$n = (x_1 + |\vec{D}|m_x)A_{02} + (y_1 + |\vec{D}|m_y)A_{12} + (z_1 + |\vec{D}|m_z)A_{22} + A_{32} \quad . \tag{4.21}$$

Da equação (4.19) conclui-se que,

$$|\vec{D}| = \frac{u - x_1A_{00} - y_1A_{10} - z_1A_{20} - A_{30}}{m_xA_{00} + m_yA_{10} + m_zA_{20}}. \tag{4.22}$$

Expressando u em função das coordenadas do dispositivo (equação 4.15), tem-se:

$$|\vec{D}| = \frac{x'C'_{00} + y'C'_{10} + C'_{20} - x_1A_{00} - y_1A_{10} - z_1A_{20} - A_{30}}{m_xA_{00} + m_yA_{10} + m_zA_{20} + A_{30}}. \tag{4.23}$$

As coordenadas (x_2, y_2, z_2) correspondentes à nova posição (x', y') do *mouse* são,

$$\begin{array}{l} x_2 = x_1 + |\vec{D}|m_x \\ y_2 = y_1 + |\vec{D}|m_y \\ z_2 = z_1 + |\vec{D}|m_z \end{array} \tag{4.24}$$

Conforme definido anteriormente $m_x = \frac{\cos\alpha'}{|\vec{d}|}$, $m_y = \frac{\cos\beta'}{|\vec{d}|}$ e e $m_z = \frac{\cos\gamma'}{|\vec{d}|}$ são informações obtidas diretamente do movimento do *mouse*. Onde α', β' e γ' são ângulos obtidos a partir do espaço DC (conforme figura 4.8) e o módulo ($|\vec{d}|$) do deslocamento do *mouse* é obtido a partir da equação (4.13).

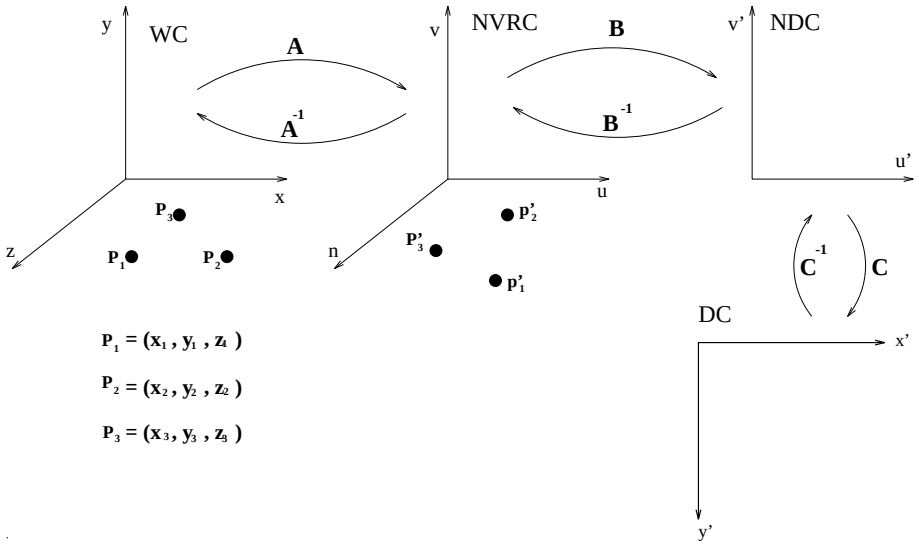

Figura 4.9: Técnica do plano de trabalho.

4.4.5 Técnica do plano de trabalho

O plano de trabalho é uma ferramenta que permite o mapeamento preciso de uma posição (x', y') do *mouse* na tela para uma posição (x, y, z) recuperada a partir de um plano de trabalho descrito no espaço NVRC.

Conforme explicado na seção 3.1, através da transformação de visualização a coordenada "n" no espaço NVRC é perdida. A definição de um plano de trabalho no espaço NVRC permite recuperar de maneira precisa esta informação.

Sejam definidos três pontos em coordenadas WC para definição do plano (x_1, y_1, z_1), (x_2, y_2, z_2) e (x_3, y_3, z_3). Através da transformação de focalização A, obtém-se os três pontos em coordenadas NVRC (figura 4.9).

$$\begin{bmatrix} u_1 & v_1 & n_1 & 1 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \end{bmatrix} * A$$

$$\begin{bmatrix} u_2 & v_2 & n_2 & 1 \end{bmatrix} = \begin{bmatrix} x_2 & y_2 & z_2 & 1 \end{bmatrix} * A$$

$$\begin{bmatrix} u_3 & v_3 & n_3 & 1 \end{bmatrix} = \begin{bmatrix} x_3 & y_3 & z_3 & 1 \end{bmatrix} * A$$

o plano de trabalho obtido no espaço NVRC a partir destes pontos é representado pela

equação:

$$Au + Bv + Cn + D = 0 \tag{4.25}$$

Para mapear uma posição (x', y') de entrada do *mouse* (DC) para uma posição (x, y, z) única no espaço WC, a técnica faz uso deste plano. Assim a partir da equação (4.25), a coordenada "n" do espaço NVRC pode ser obtida,

$$n = \frac{-D - Au - Bv}{C} \tag{4.26}$$

Assim também, para que uma posição (x', y') seja interpretada em coordenadas NDC, pode-se utilizar equações (4.3) e (4.4).

Continuando a considerar o tipo de projeção paralela, um ponto (u, v, n) do espaço NVRC pode ser dado a partir das equações (4.26), (4.3) e (4.4). Assim,

$$u = x'C'_{00} + y'C'_{10} + C'_{20} \quad , \tag{4.27}$$

$$v = x'C'_{01} + y'C'_{11} + C'_{21} \quad e \tag{4.28}$$

$$n = \frac{-D - Au - Bv}{C} \quad . \tag{4.29}$$

que em WC é dado por:

$$\left[\begin{array}{cccc} x & y & z & 1 \end{array} \right] = \left[\begin{array}{cccc} u & v & n & 1 \end{array} \right] * A^{-1} \tag{4.30}$$

ou seja,

$$\left[\begin{array}{cccc} x & y & z & 1 \end{array} \right] = \left[\begin{array}{cccc} x'C'_{00} + y'C'_{10} + C'_{20} & x'C'_{01} + y'C'_{11} + C'_{21} & \frac{-D-Au-Bv}{C} & 1 \end{array} \right] * A^{-1} \tag{4.31}$$

sendo que,

$$A^{-1} = \left[\begin{array}{cccc} A'_{00} & A'_{01} & A'_{02} & A'_{03} \\ A'_{10} & A'_{11} & A'_{12} & A'_{13} \\ A'_{20} & A'_{21} & A'_{22} & A'_{23} \\ A'_{30} & A'_{31} & A'_{32} & A'_{33} \end{array} \right] \tag{4.32}$$

Logo,

$$x = A'_{00}(x'C'_{00}+y'C'_{10}+C'_{20})+A'_{10}(x'C'_{01}+y'C'_{11}+C'_{21})+A'_{20}(\frac{-D-Au-Bv}{C})+A'_{30} \tag{4.33}$$

$$y = A'_{01}(x'C'_{00}+y'C'_{10}+C'_{20})+A'_{11}(x'C'_{01}+y'C'_{11}+C'_{21})+A'_{21}(\frac{-D-Au-Bv}{C})+A'_{31} \quad (4.34)$$

$$z = A'_{02}(x'C'_{00}+y'C'_{10}+C'_{20})+A'_{12}(x'C'_{01}+y'C'_{11}+C'_{21})+A'_{22}(\frac{-D-Au-Bv}{C})+A'_{32} \quad (4.35)$$

Assim, com esta técnica as coordenadas x, y e z de um ponto qualquer no mundo (WC) são obtidas a partir das coordenadas do dispositivo (DC) e da equação do plano.

A idéia de plano de trabalho pode ser generalizada para superfície de trabalho. Como exemplo, poderíamos trabalhar com superfícies cilíndricas ou esféricas. A escolha de um plano no contexto deste trabalho deve-se à facilidade de cálculo e ao tipo de aplicações que se deseja fazer.

4.5 Técnicas auxiliares de visualização

Com o intuito de favorecer as tarefas de interação 3D, foram sugeridas:

- estratégias e ferramentas visuais para realce da profundidade do espaço ou ambiente 3D; e
- uma representação gráfica para cada uma das técnicas de interações 3D, através do uso de um cursor tridimensional específico por cada técnica.

4.5.1 Realce do ambiente 3D

Entre as ferramentas[2] usadas para realce do ambiente 3D, tem-se:

- grades;
- eixos 3D.

As grades (figura 4.10) servem como pautas de referência sobre o plano de visualização para ter relações exatas de distâncias entre dois pontos quaisquer. O uso das grades pode ser otimizado através de uma parametrização dos espaçamentos da malha – como

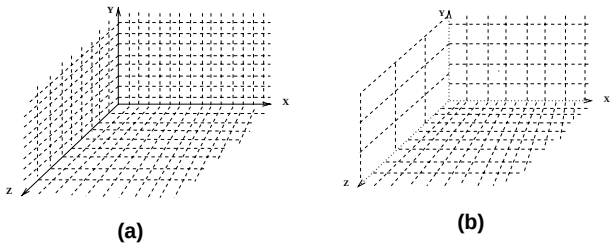

Figura 4.10: Grades: (a) com pautas iguais; (b) com pautas diferentes.

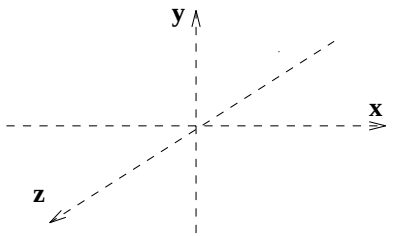

Figura 4.11: Eixos 3D.

apresentado na figura 4.10(b). Assim, pode-se ter diferentes tipos de pautas com unidades de medidas diversas.

Os eixos 3D (figura 4.11) são sistemas de referência que têm como função apresentar a direção e o sentido dos eixos coordenados x, y e z do espaço WC.

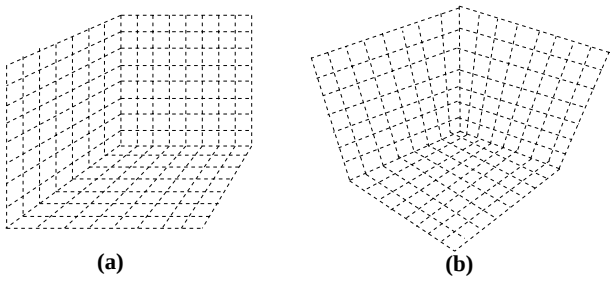

Figura 4.12: Projeções: (a) paralela e (b) perspectiva.

Entre as estratégias que podem ser usadas para facilitar a interpretação do ambiente 3D têm-se:

[2]importadas do IQL (seção 3.4).

- mudança do tipo de projeção da cena de paralela para perspectiva e vice-versa (figura 4.12);
- deslocamento de entidades gráficas na direção da linha de visão do observador (ou câmera);
- rotação de entidades gráficas em torno dos eixos que não sejam perpendiculares à tela de visualização;
- parametrização interativa dos dados da câmera, ou movimentação contínua dela através do *mouse*.

Estas estratégias e ferramentas podem também ser combinadas de forma a prover maior realidade tridimensional.

4.5.2 Representação gráfica das interações 3D

As técnicas de mapeamento descritas na seção 4.4 podem ter diferentes representações gráficas. Assim, um cursor 3D é uma forma de representação gráfica de uma técnica de interação 3D. Portanto, espera-se que a escolha destes cursores venha a caracterizar o tipo de interação que está sendo realizada. Dessa forma,

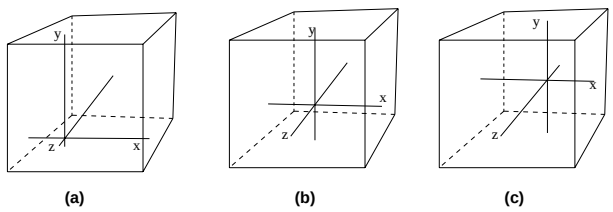

Figura 4.13: Fases do cursor 3D em direção à profundidade da câmera na técnica do particionamento.

- para a técnica de particionamento do espaço imagem, apropriada para movimentos livres no espaço WC, definiu-se um cursor 3D que permitisse uma rápida orientação espacial, assim como uma boa estimativa da sua posição no espaço WC (figura 4.13).
- para a técnica dos movimentos circulares, onde movimentos circulares do *mouse* (nos sentidos horário e anti-horário) são interpretados como movimentos de avanço

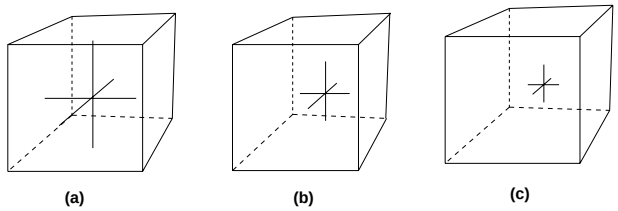

Figura 4.14: Interação do cursor 3D com a profundidade da câmera na técnica dos movimentos circulares.

e retrocesso do cursor no sentido da profundidade da câmera, propõe-se um cursor 3D que permita a variação interativa das suas proporções na tela (figura 4.14).

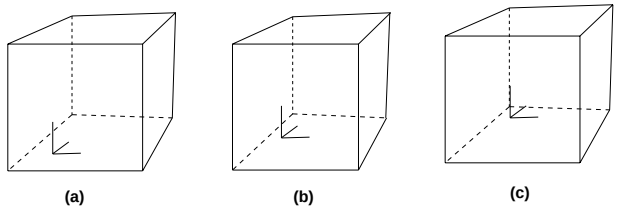

Figura 4.15: Interação do cursor 3D com a profundidade da câmera na técnica do plano de trabalho.

- para a técnica do plano de trabalho, em que o movimento do *mouse* ocorre somente sobre um plano, definiu-se um formado de cursor que permitisse o "assentamento" do cursor sobre o plano (figura 4.15).

A escolha das formas de um cursor em geral é intuitiva, porém espera-se que alguns critérios sejam obedecidos, tais como:

- suporte à profundidade;
- semântica compreensível;
- visualização rápida;
- orientação espacial.

Sugere-se também que cada uma das técnicas tenha um formato de cursor próprio, de modo que o usuário possa associar as formas do cursor ao tipo de técnica de interação que está sendo usada.

4.6 O IQLT: uma biblioteca de técnicas para interações 3D

Um dos recursos que uma biblioteca de componentes oferece é o controle das entradas do *mouse* em janelas de desenho (*canvas*). Uma janela de desenho é uma área de entrada na tela em que o usuário, através de um dispositivo como o *mouse*, manipula entidades gráficas 2D. A desvantagem de uma biblioteca de componentes ocorre quando, através de uma área de desenho como esta, deseja-se interagir com entidades gráficas 3D, uma vez que a maioria de bibliotecas de componentes de interface não previram a necessidade de interações 3D através de *mouse*.

Considerando esta necessidade, deseja-se que o uso de técnicas de mapeamento nestas bibliotecas permita a interação com as entidades 3D em janelas de desenho com recursos tridimensionais (*canvas 3D*) [Vela93]. Com este objetivo, diferentes técnicas 3D foram agrupadas numa biblioteca, incorporando-as e usando-as de forma análoga às componentes de interface 2D. Para isto, foram reunidas as técnicas de interação e visualização 3D descritas neste capítulo (seção 4.4) e os recursos tridimensionais do módulo IQL descritos no capítulo anterior (seção 3.4). Esta biblioteca de suporte à interação e visualização 2D e 3D será chamada de IQL**T** (IQL + **Techniques 3D**).

O IQL**T** foi desenvolvido com o auxílio de sistemas de suporte do ambiente *X* (seção 3.3). Para organização da biblioteca foi necessário a divisão conceitual das funções do IQL e das técnicas de mapeamento em módulos, de modo a alcançar integrabilidade em dois níveis:

- integração entre componentes de interface e
- integração da biblioteca IQL**T** com outros sistemas de suporte (tais como bibliotecas *XView*, PRODIA e *XMotif*).

Para garantir a integrabilidade, modularidade e portabilidade do IQL**T** as seguintes estratégias foram adotadas:

- modularização das técnicas de interação 2D e 3D;
- interface funcional clara;

- máxima utilização de convenções padronizadas e funções de domínio público (tal como o *Graphics Gems*, por exemplo) e
- uso de interfaces de comunicação chamada *drivers* (apêndice D)[3] (figura 4.16).

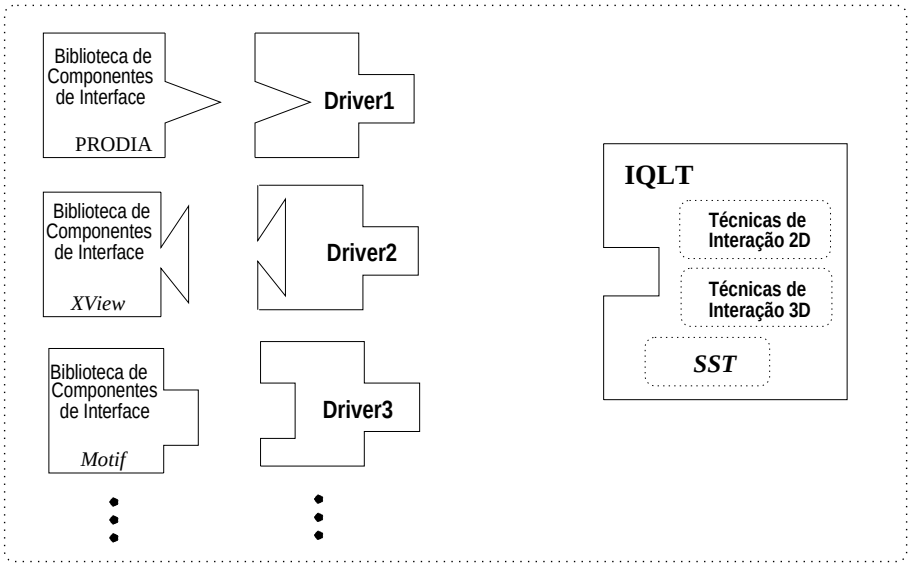

Figura 4.16: Integração do IQL**T** em bibliotecas através de *drivers*.

Esta organização dos módulos implicou também o desacoplamento das estruturas de dados do IQL no PRODIA e a reorganização dos arquivos de inclusão.

A biblioteca IQL**T** foi dividida em dois módulos:

(1) módulo de suporte à visualização e

(2) módulo de suporte à interação.

4.6.1 Módulo de suporte à visualização

O módulo de suporte à visualização (figura 4.17) procura simular um ambiente 3D. Ele reúne recursos que permitem a realização dessas funcionalidades. Este módulo foi dividido da seguinte forma:

[3]Com a criação dos *drivers*, o IQL**T** pôde efetivamente ser acoplado a outras bibliotecas de componentes, estendendo as suas capacidades.

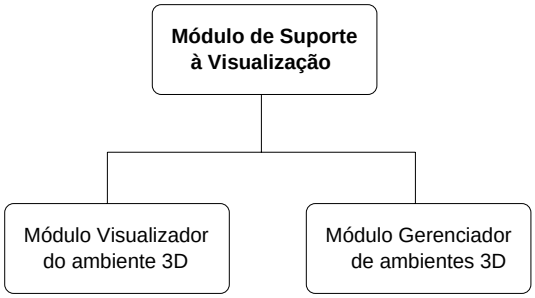

Figura 4.17: Módulo de suporte à visualização.

- módulo visualizador do ambiente e
- módulo gerenciador do ambiente.

O módulo visualizador do ambiente é responsável pelas transformações direta e inversa dos modelos 2D e 3D para as entidades gráficas visualizáveis (seções 3.4 e 3.1) e suporta funções para realce do ambiente: grades e eixos 2D e 3D (seções 3.4 e 4.5).

O módulo gerenciador do ambiente é responsável pela inicialização dos recursos associados a ele e possui o controle da edição dos ambientes. Entre os recursos associados a ele têm-se: câmara, grades, eixos e entidades gráficas.

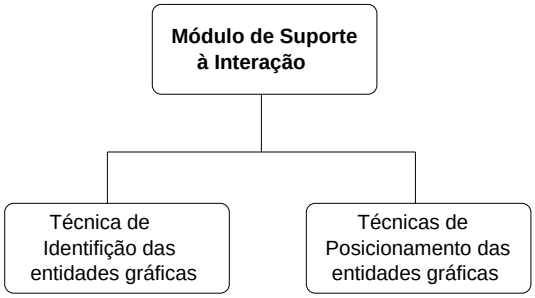

Figura 4.18: Módulo de suporte à interação.

4.6.2 Módulo de suporte à interação

O módulo de suporte à interação (figura 4.18) inclui os recursos para suporte à estruturação das entidades gráficas e possibilita a manipulação das mesmas. Existem especificamente dois módulos para interação com as entidades:

- módulo para identificação das entidades,
- módulo para posicionamento das entidades.

O primeiro módulo contém a técnica de identificação ordenada e o segundo contém as técnicas de mapeamento 3D desenvolvidas como componentes de interface (seção 4.4) (figura 4.19).

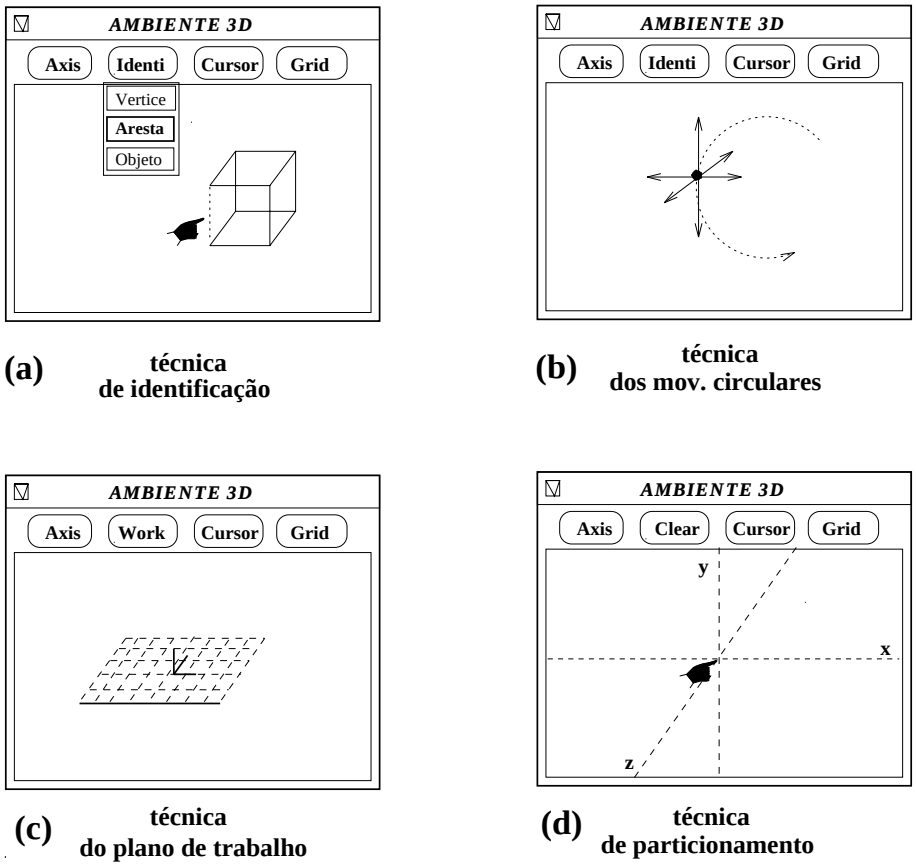

Figura 4.19: Técnicas de interação 3D.

Capítulo 5

Implementação e resultados

Em vista da nossa proposta, apresentam-se neste capítulo os detalhes concernentes à implementação do conjunto portátil de técnicas básicas – descritas no capítulo anterior – para interação e suporte à percepção de objetos 3D, usando como dispositivo de entrada o *mouse* 2D.

5.1 Componentes de interface

Nesta seção será dada uma breve explicação sobre cada uma das técnicas de interação implementadas como componentes de interface 3D.

No apêndice B, apresenta-se o manual de referência e, no apêndice C, o manual de programação destes componentes no contexto do *XView*.

5.1.1 O componente da técnica dos movimentos circulares

A técnica dos movimentos circulares é uma ferramenta que permite interagir com a profundidade da câmera através de movimentos circulares do *mouse*. Um ponto (x, y, z) no espaço WC pode ser localizado através desta técnica tornando correspondentes as coordenadas $(x'y')$ do *mouse* com as coordenadas u e v do ponto em NVRC. A terceira coordenada – a coordenada n – é obtida a partir de três pontos sucessivos e não colineares captados pelo *mouse*.

Figura 5.1: Cursor 3D na janela de visualização.

De acordo com a formalização dada na seção 4.4.3, a coordenada de profundidade da câmera – coordenada "n" do espaço NVRC – é obtida através do produto vetorial de três pontos coplanares descritos no plano de entrada do *mouse*. Primeiro, estes três pontos são mapeados de coordenadas DC para coordenadas NDC. Segundo, uma vez obtidos estes pontos em coordenadas NDC, são calculados dois vetores. Terceiro, o produto vetorial destes vetores resulta na coordenada "n" de profundidade da câmera.

O movimento na direção "n" é relativo; isto é, uma nova posição no espaço NVRC depende de uma posição anterior. A estratégia utilizada para estimar a primeira posição n_o foi fazê-la coincidente com o plano frontal de visualização. As outras coordenadas do ponto, isto é, as coordenadas "u" e "v", são obtidas diretamente de um ponto do *mouse*, considerando um tipo de projeção paralela (equações 4.6 e 4.7).

Na figura 5.1, tem-se a representação gráfica dada à tecnica conforme proposto na seção 4.5. O cursor 3D é projetado a partir do ponto (x, y, z) encontrado no espaço WC e de acordo ao ângulo de visualização. Em cada interação, o cursor 3D é projetado maior, ou menor, de acordo com a distância do cursor em relação ao observador.

Uma aplicação desta técnica será vista na seção 5.4.

Figura 5.2: Cursor 3D na janela de visualização.

5.1.2 O componente da técnica de particionamento

A técnica de particionamento é uma ferramenta para interações 3D que permite o mapeamento das coordenadas do dispositivo de entrada (x', y') para um ponto do espaço 3D (x, y, z).

Como apresentado na figura 5.2, a realimentação visual desta técnica de mapeamento é a de um cursor 3D. O cursor 3D é projetado na tela conforme o ângulo de visualização da câmera. Esta projeção varia cada vez que um novo ângulo de visualização é definido. Os movimentos horizontais, verticais e diagonais do *mouse* são interpretados, respectivamente, como movimentos nas coordenadas x, y e z do espaço WC.

Esta técnica é baseada em movimentos relativos; isto é, uma nova posição no espaço WC depende de uma posição anterior. Assim, o primeiro ponto para inicialização dos movimentos do cursor 3D no espaço WC precisa ser estimado. Adotou-se a seguinte estratégia: o ponto (x_o, y_o, z_o) corresponde ao ponto $(u_o, v_o, 0)$ que, projetado na tela, identifica-se com a posição corrente do cursor.

Esta técnica apresenta uma dificuldade. Conforme o seu desenvolvimento teórico, o deslocamento do cursor 2D do *mouse* no espaço DC deve corresponder *a priori* ao deslocamento do cursor 3D no espaço WC, mas isto nem sempre acontece. O problema ocorre porque os cosenos diretores são estimados e, em decorrência disso, o módulo de deslocamento

do cursor 3D no espaço WC (equação 4.23) também é um valor estimado e não exato[1].

Viu-se que, a partir das equações (4.19) e (4.20) no capítulo 4, podem ser calculados valores distintos de deslocamentos[2].

Atributos fixos de cor e estilo de linha podem facilmente ser mudados, dependendo da escolha do usuário.

5.1.3 O componente da técnica do plano de trabalho

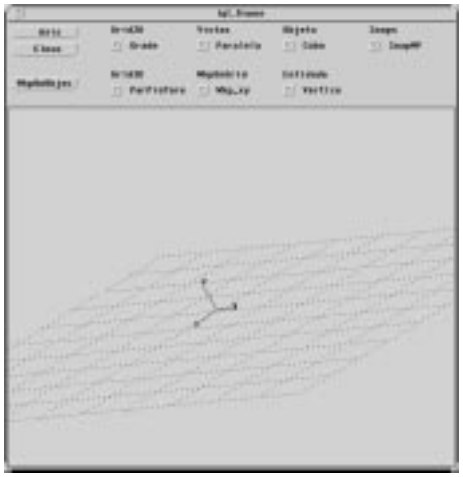

Figura 5.3: O plano de trabalho no espaço tridimensional.

A técnica do plano de trabalho é uma ferramenta para interação tridimensional que permite movimentos do *mouse* sobre um plano no espaço 3D (figura 5.3). Cada ponto (x', y') sobre a tela é mapeado para uma posição tridimensional (x, y, z) no plano. Esta ferramenta de interação possui duas formas de movimento sobre o plano:

- movimento discreto e
- movimento contínuo.

[1] Como foi visto na seção 4.4.4, para estimar o módulo de deslocamento do cursor 3D ($|\vec{D}|$), utilizam-se as equações (4.19) ou (4.20). O cálculo de $|\vec{D}|$ a partir da equação (4.19) não necessariamente coincidirá com o cálculo estimado a partir da equação (4.20).

[2] Em geral tem-se percebido que quando o valor do denominador do $|\vec{D}|$ obtido na equação 4.23, está no intervalo de $[-5x10^{-3}, +5x10^{-3}]$, o valor $|\vec{D}|$ não é uma boa estimativa para deslocamento. Quando isto ocorre, a outra equação é utilizada.

O movimento discreto do *mouse* sobre o plano é resultado da presença de pontos atratores nos nós da malha do plano de trabalho. Assim, todo valor (x', y') do *mouse* próximo a um nó da malha é atraído para estes pontos do plano.

O movimento contínuo do *mouse* sobre o plano é resultado de um mapeamento contínuo sobre qualquer ponto da malha. Assim, todo valor (x', y') do *mouse* é mapeado para um valor (x, y, z) diferente no plano.

Como sugerido na seção 4.5, a representação gráfica dos movimentos do *mouse* sobre o plano é um cursor tridimensional com três semi-eixos ortogonais, disposto conforme a inclinação do plano no espaço.

O plano pode ser visível ou invisível, dependendo da escolha do usuário. A técnica foi implementada de modo que somente possam ser localizados os pontos contidos no interior do plano. A técnica poderia ser flexibilizada de modo que o plano de trabalho venha a ser infinito. Isto permitiria o mapeamento de qualquer ponto da tela para o espaço WC.

Um detalhe de implementação importante é que existe um único caso em que o valor do ponto x, y, z do espaço WC não pode ser definido. Quando a coordenada n do espaço NVRC é infinita (observe a equação 4.26 da seção 4.4.5). Para contornar esta situação, a coordenada n, que se refere à profundidade da câmera, foi substituída pela profundidade do plano frontal do volume de visualização[3] – definida numa projeção paralela como $z = 0$.

5.2 Ferramentas visuais

Nesta seção serão referidas algumas funcionalidades importadas do módulo IQL (seção 3.4) que foram adaptadas para serem integradas na interface gráfica *ProSIm*.

A interface *ProSIm* permite a definição de diferentes tamanhos do mundo real (WC). Os eixos e grades 3D implementados no IQL não previram a necessidade de se ter um mundo real parametrizável[4]. Para inclusão destas ferramentas nesta interface foi necessário o uso de funções intermediárias para acesso ao tamanho corrente do mundo real. Com isso, a projeção das ferramentas será também proporcional ao tamanho do mundo real.

[3] seção 3.1.
[4] Entenda-se variável.

A adaptação destas ferramentas ao *ProSIm* requereu também a inclusão de *drivers* para uso das grades e eixos, tanto na plataforma gráfica *XView* – sobre a qual foi construído o *ProSIm* – como na plataforma original PRODIA.

A seguir será dada uma breve explicação sobre os detalhes concernentes à implementação de cada ferramenta.

5.2.1 Eixos 3D

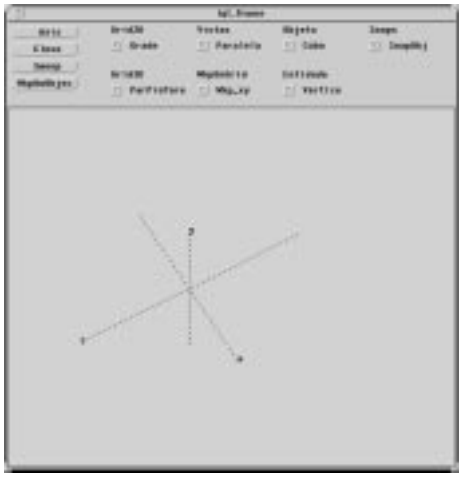

Figura 5.4: Eixos tridimensionais

Os eixos 3D são um componente visual usado para orientação dos usuários no espaço 3D. Incluem referenciais tridimensionais e dividem a tela em seis regiões que correspondem aos seis semi-eixos $(+x, -x, +y, -y, +z, -z)$ (figura 5.4). Os eixos 3D foram reconfigurados de modo que a origem dos eixos nem sempre coincida com a origem do sistema de coordenadas WC. Os eixos são projetados na tela de visualização de acordo com a posição do observador.

Os eixos 3D possuem atributos de visibilidade, cor, estilo de linha e tamanho. Estes atributos podem facilmente ser mudados de acordo com a escolha do usuário.

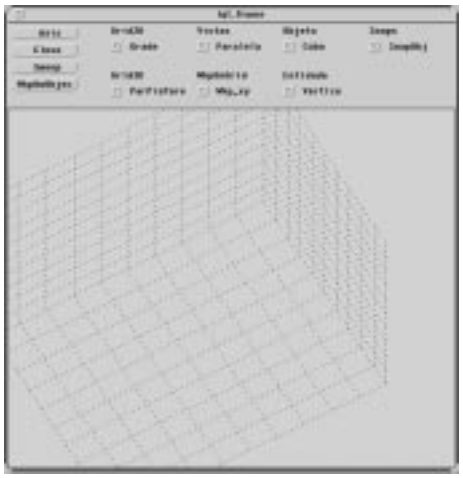

Figura 5.5: Grade tridimensional

5.2.2 Grades 3D

As grades 3D são um componente visual para realce do ambiente 3D. Sobrepostas à área de trabalho, ajudam no alinhamento de entidades gráficas ou de objetos 3D.

Cada plano das grades 3D possui marcações formando uma malha. As grades 3D podem ser combinadas por dois ou mais planos ortogonais. Na figura 5.5, apresenta-se uma forma padrão para uso do componente. É possível posicionar as grades em qualquer parte do espaço e alterar a resolução da malha.

As grades 3D possuem atributos de visibilidade, cor, estilo de linha e tamanho. Estes atributos poderiam ser mudados dependendo da escolha do usuário.

5.3 Testes de portabilidade

Foram experimentadas duas estratégias para integrar o módulo IQLT no *XView*:

1. criar um novo componente de interface, utilizando os recursos oferecidos pelo próprio *XView*, de tal forma que as funções implementadas no IQLT fossem encapsuladas neste novo componente;

2. estender os campos de atributos da área de desenho do *XView* (*Canvas*) através do uso de um campo extensível, de tal forma que todos os dados necessários ao módulo IQL**T** pudessem ser armazenados e acessados por ele.

XView dispõe de um conjunto de métodos que permitem criar novos componentes de interface (*XView Package*)[Nye90]. Estes novos componentes podem modificar a aparência ou estender as funcionalidades dos já existentes. O uso destes métodos exige a existência de uma classe-pai, da qual o novo componente herdará as características predominantes. No caso de um novo componente que requer o uso de janelas, só o componente *Window* pode ser utilizado como sua classe-pai. Isso implica que todas as facilidades oferecidas pelo componente *Canvas*[5] deverão ser adicionalmente implementadas. Para evitar este esforço computacional adicional, optamos pela segunda estratégia.

Cada componente de interface no *XView* é provido internamente de campos extensíveis (*Key Data*)[Nye90] que permitem a associação de um objeto a outro. A vantagem no uso deste segundo artifício – para a integração do IQL**T** – está no uso de todas as funcionalidades disponíveis no *Canvas*. Sob o ponto de vista organizacional do *XView*, o IQL**T** pode ser integrado a um *Canvas* através de uma extensão dos seus campos, contendo informações necessárias para a execução correta de rotinas do módulo IQL**T**. Logicamente, o IQL**T** é um novo componente de interface do *XView*[Vela93] que:

- apresenta todas as propriedades do *Canvas* e
- suporta a execução das funções do módulo IQL**T**.

As definições necessárias para o uso da biblioteca de componentes IQL**T** em qualquer sistema de construção de interfaces são encontradas nos arquivos: `"Iql.h"` e `"IqlDrv.h"`. O primeiro arquivo – `"Iql.h"` – reúne os tipos e estruturas de dados para o uso dos componentes: eixos 3D, grades 3D, as técnicas de identificação ordenada, do plano de trabalho, do particionamento e dos movimentos circulares. O segundo arquivo – `"IqlDrv.h"` – contêm os protótipos das funções de interface de comunicação (*drivers*) entre o conjunto de técnicas do IQL**T** e a biblioteca de componentes *XView* e PRODIA.

A interface de comunicação entre os componentes do IQL**T** e as bibliotecas *XView* e PRODIA foram descritas em linguagem *C*.

[5] *Canvas* é uma sub-classe de *Window*.

A integração foi concebida de tal forma que, para os programas aplicativos, a inicialização do IQLT é similar a um *Canvas* do *XView*, como aparece nos códigos seguintes (note-se que, só por questão de clareza, optamos por distinguir todas as funções relacionadas com IQLT pelas iniciais `iql_`):

```
/*-----------------------------------------------*/
/*      Inclusao do componente Canvas do XView       */
/*-----------------------------------------------*/
#include <xview/canvas.h>
{   Frame          frame;
    Canvas         canvas;

    /* inicializacao do xview */
    xv_init(XV_INIT_ARGC_PTR_ARGV,&argc,argv,NULL);
    frame = (Frame)xv_create(NULL, FRAME, NULL);

    /* cria um Canvas */
    canvas = (Canvas)xv_create(frame, CANVAS, NULL);

    xv_main_loop(frame);
}

/*-----------------------------------------------*/
/*      Inclusao do componente Canvas do IQLT        */
/*-----------------------------------------------*/
#include <xview/canvas.h>
#include "Iql.h"
#include "IqlDrv.h"
{   Frame       frame;
    Canvas      frame_id;
    char        *desc;

    /* inicializacao do xview */
    xv_init(XV_INIT_ARGC_PTR_ARGV, &argc, argv, NULL);
    frame = (Frame)  xv_create(NULL, FRAME, NULL);

    /* cria um Canvas do IQLT */
    iql_FrameCreate (frame, frame_id, &desc);

    xv_main_loop(frame_call);
}
```

A única diferença entre os dois trechos de código está na chamada das funções: `xv_create()` e `iql_FrameCreate()`. Entretanto, sob o ponto de vista de recursos, a cha-

mada de `iql_FrameCreate()` inicializa todos os recursos: tais como grades, eixos, câmera e entidades gráficas tridimensionais, tornando-os disponíveis aos programas aplicativos.

Na próxima seção, apresenta-se um aplicativo para um dos componentes, incorporado atualmente na interface interativa *ProSIm*[6].

5.4 Uma aplicação

No modelo de câmera implementado na interface *ProSIm* os movimentos do observador são em torno de uma esfera. O observador realiza movimentos horizontais (longitude da esfera – ângulo ϕ), verticais (latitude da esfera – ângulo θ) e de aproximação e afastamento em direção ao centro de atenção (raio da esfera – ρ) (figura 5.6).

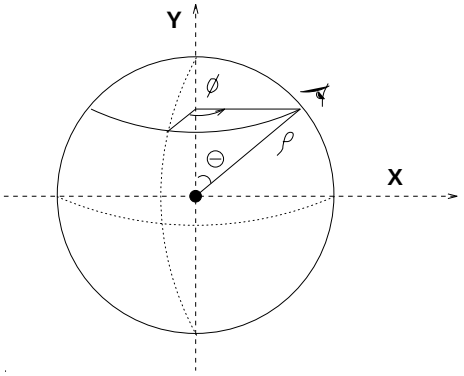

Figura 5.6: Modelo de câmera na interface *ProSIm*.

Inicialmente, estes parâmetros eram controlados a partir de deslizadores gráficos (*sliders*). Mostra-se aqui que, com o uso da técnica dos movimentos circulares, estes parâmetros (ϕ, θ, ρ) podem ser controlados de maneira bastante intuitiva a partir de movimentos do *mouse*.

Viu-se na seção 4.4.3 que a técnica de movimentos circulares fornece a posição corrente do *mouse* em coordenadas NVRC, (u_i, v_i, n_i). Se for armazenada a posição anterior $(u_{i-1}, v_{i-1}, n_{i-1})$, pode-se considerar que a diferença $u_i - u_{i-1}$ seja o deslocamento do observador em torno da longitude da esfera; a diferença $v_i - v_{i-1}$, o deslocamento em torno

[6] *Sistema de Prototipação e Síntese de Imagens Fotorealísticas* em desenvolvimento no Departamento de Engenharia de Computação e Automação da Engenharia Elétrica (DCA) da UNICAMP.

da latitude e a diferença $n_i - n_{i-1}$, o deslocamento ao longo do eixo do observador – centro de atenção.

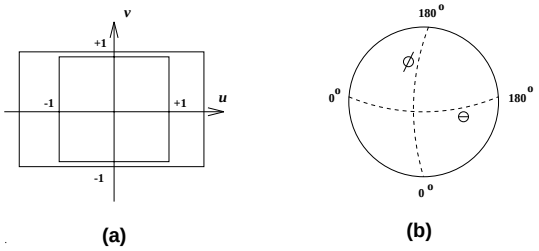

Figura 5.7: (a) Espaço NVRC (b) Movimento da câmera.

De acordo com a figura 5.7, os deslocamentos do *mouse* podem ser relacionados com os deslocamentos da câmera em coordenadas esféricas da seguinte forma: uma variação dos movimentos do *mouse* em u (Δu) corresponde a uma variação do observador na longitude ($\Delta\theta$) e uma variação dos movimentos do *mouse* em v (Δv) corresponde, analogamente, a uma variação do observador na latitude ($\Delta\phi$). Como o domínio de u e v é $[-1, +1]$ e deseja-se simular os movimentos do observador no intervalo de $[0^o, 180^o]$ em longitude[7] e de $[0^o, 180^o]$ em latitude, tem-se as seguintes relações:

$$\frac{\Delta\theta}{\Delta u} = \frac{180^o}{2} \quad e \quad \frac{\Delta\phi}{\Delta v} = \frac{180^o}{2}, \tag{5.1}$$

isto é,

$$\boxed{\Delta\theta = 90^o(u_i - u_{i-1})} \quad \text{e} \quad \boxed{\Delta\phi = 90^o(v_i - v_{i-1})}$$

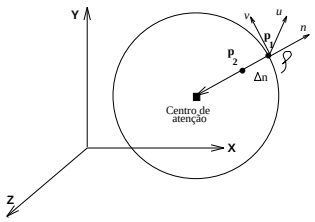

Figura 5.8: Deslocamentos relativos da câmera em ρ.

O deslocamento do observador em direção ao centro de atenção ($\Delta\rho$) corresponde ao distanciamento deste em relação ao plano de projeção ($n = 0$). Fixando um ponto qualquer

[7]Sem perda de generalidade, os movimentos do observador foram restritos a $[0^o, 180^o]$.

sobre o plano, $p_1 = (u_p, v_p, 0)$, a nova posição do observador seria

$$p_2 = (u_p, v_p, \Delta n). \tag{5.2}$$

Por simplicidade, escolhe-se $u_p = v_p = 0$. Assim, o deslocamento $\Delta\rho$ em WC é obtido calculando o módulo entre os pontos p_1 e p_2 transformados para WC através da matriz de focalização inversa (A^{-1}). Como resultado desta transformação, obtém-se os pontos:

$p_{1_{wc}} = (x_1, y_1, z_1)$ e

$p_{2_{wc}} = (x_2, y_2, z_2)$.

e o deslocamento $\Delta\rho$ é,

$$\boxed{\Delta\rho = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}}$$

Assim, $\Delta\theta$, $\Delta\phi$ e $\Delta\rho$ são os deslocamentos do observador em WC controlados a partir do *mouse*.

Como na técnica dos movimentos circulares a coordenada n é obtida em função das coordenadas x' e y' do *mouse* (equações 4.3, 4.4 e 4.5), os movimentos horizontais (longitude da esfera) e verticais (latitude da esfera) do *mouse* provocam deslocamentos em direção ao centro de interesse. Esta situação não é necessariamente um problema, mas, se o que se quer é que movimentos na longitude e latitude da esfera não afetem a distância entre observador e centro de atenção, podem-se controlar os deslocamentos do observador na latitude e longitude através de um dos botões do *mouse* e os deslocamentos em direção ao centro de interesse (profundidade da câmera) através de outro botão. No trabalho foram implementadas e incorporadas no *ProSIm* estas duas situações de controle da câmera e deixadas disponíveis para escolha do usuário.

As figuras 5.9, 5.10 e 5.11, ilustram uma sequência de movimentos rotacionais no sentido horário, permitindo deslocamentos da câmera em direção ao centro de atenção.

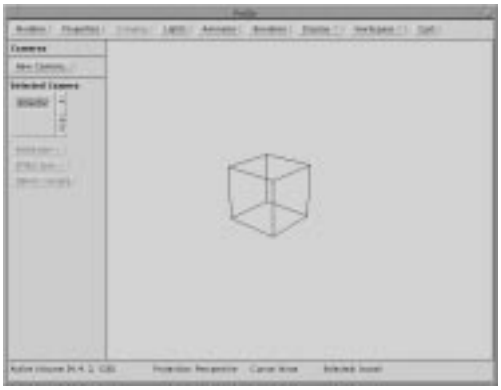

Figura 5.9: Primeira etapa da câmera móvel.

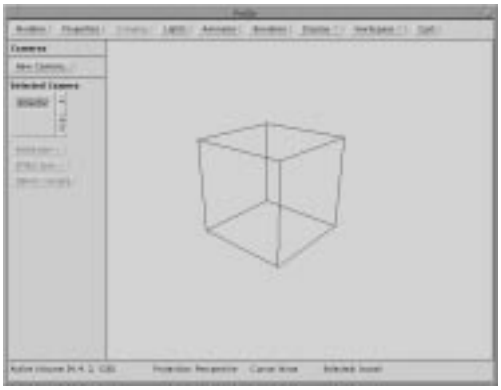

Figura 5.10: Segunda etapa da câmera móvel.

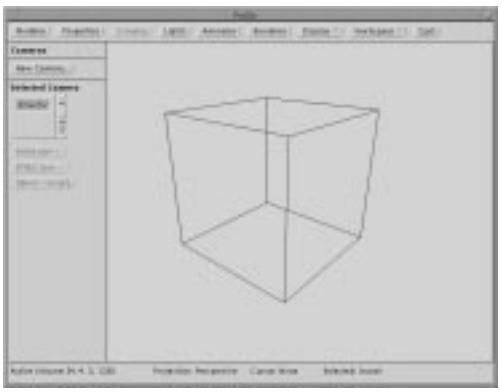

Figura 5.11: Terceira etapa da câmera móvel.

Capítulo 6

Conclusões

"Tudo têm seu tempo determinado, e há tempo
para todo o propósito debaixo do céu:
Há tempo de nascer, e tempo de morrer;
tempo de derrubar, e tempo de edificar;
tempo de chorar, e tempo de rir
tempo de espalhar pedras, e tempo de ajuntar pedras;
tempo de estar calado, e tempo de falar;".
tempo de plantar, e tempo de arrancar o que se plantou;
tempo de guerra e tempo de paz".
Eclesiastes 3:1-5.
"Hoje é tempo de arrancar o que se plantou..."

Neste trabalho foram analisadas e implementadas um conjunto de técnicas que permitem interagir diretamente com o espaço 3D através de *mouse*. Esta análise não tem como interesse final julgar uma técnica de interação melhor do que a outra, nem fazer uso de uma delas com alguma finalidade específica; mas, proporcionar um estudo que permita otimizá-las para reutilização em diferentes aplicações de manipulação direta.

As propostas definidas no capítulo 4 foram atingidas. Para o problema de recuperação da coordenada da profundidade foram implementadas algumas técnicas de interação 3D: a técnica de identificação ordenada, dos movimentos circulares, de particionamento do espaço imagem e do plano de trabalho. A partir do estudo detalhado destas técnicas, verifica-se que de fato muitas das técnicas de interação 3D foram limitadas às tarefas de interação para as quais foram construídas. O estudo das técnicas nas suas diferentes coor-

denadas de visualização: do mundo real (WC), da câmera (NVRC), do dispositivo normalizado (NDC) e do dispostivo de entrada (DC), permite a aplicação de uma mesma técnica a diferentes contextos da manipulação 3D. Assim, por exemplo, a técnica dos movimentos circulares pode ser aplicada para movimentos da câmera (seção 5.4), posicionamento de um ponto no espaço 3D (seção 5.1.1) e movimentos rotacionais de um objeto [Evan81].

Para o problema de modelamento das relações entre as entidades gráficas 3D, optou-se pelo uso da biblioteca de funções SST.

Com o intuito de permitir o realce do ambiente 3D em que as interações acontecem, foram readaptados os códigos da biblioteca IQL que implementam grades e eixos. Tais utilitários podem ser usados como recursos referenciais para posicionamento e estimativas de distâncias entre as entidades gráficas contidas no ambiente 3D. Mostra-se também que o ambiente pode ser realçado por uma estratégia baseada no movimento interativo da câmera, fazendo uso da técnica dos movimentos circulares.

Para o problema de ausência de recursos tridimensionais nas bibliotecas de componentes, foi adotada a estratégia do uso de interfaces de comunicação (*drivers*). Neste trabalho, deixa-se à disposição um conjunto de funções de interface de comunicação, que permitem a incorporação dos componentes de interface do IQL**T** à biblioteca *XView*, estendendo as suas funcionalidades.

Entre as contribuições deixadas no trabalho ressaltam-se ainda

- a reorganização dos códigos do núcleo gráfico IQL e sua definição como biblioteca de funções extensível e configurável.
- foram especificadas as funções de uso destas componentes da biblioteca. E um exemplo de integração das mesmas no contexto do *XView*.

Este trabalho abre o espaço para explorar alguns aspectos interessantes em relação às interações 3D. Como sugestões para trabalhos futuros, gostaria de destacar os seguintes pontos:

- Aplicação da técnica de particionamento do espaço imagem (seção 4.4.4) para criação de uma outra câmera móvel, assim como na técnica dos movimentos circulares. Para

determinação de uma nova posição do observador, devem ser consideradas as coordenadas de entrada do *mouse* transformadas para o espaço NVRC.

- Aplicação da técnica dos movimentos circulares (seção 4.4.3) para movimentos rotacionais de objetos 3D: os movimentos circulares do cursor indicariam rotações do objeto em torno do eixo n. Movimentos na horizontal e vertical indicariam rotações em torno dos eixos u e v.
- Criação de um conjunto adicional de cursores 3D para serem usados pelas diferentes técnicas de interação 3D: em nosso trabalho foi criado um único cursor 3D para cada uma das técnicas de interação. Estas representações gráficas definidas para cada uma das técnicas podem ser consideradas como representações *default*. O que sugere-se como trabalho complementar é deixar à disposição do usuário um conjunto adicional de representações do cursor.
- Desenvolvimento de técnicas *rubberbanding* como ferramentas básicas para construção de objetos 3D[1]. Assim, por exemplo, podem ser desenvolvidas técnicas *rubber-line*, *rubber-rectangle*, *rubber-circle* e *rubber-ellipse*.
- Através da composição de tarefas, as técnicas de interação podem ser combinadas para modelagem de objetos 3D. Assim, por exemplo, para construir objetos a partir da técnica varredura (*sweeping*) [Fole90], podem ser definidos interativamente o perfil e a trajetória em *sweeping* translacional.
- Desenvolvimento de novas interfaces de comunicação (*drivers*) para uso integrado das técnicas de interação do IQLT com a biblioteca de componentes *XMotif*.

[1] Foley pg. 382-386 [Fole90].

Apêndice A

Dispositivos de entrada 2D

Como parte do processo de escolha do dispositivo de entrada, foi realizado um estudo dos diferentes dispositivos de entrada. Este apêndice A apresenta uma classificação de dispositivos de entrada, das tarefas mais comuns de usuário e uma descrição dos vários dispositivos de posicionamento 2D [Merk91] [Fole90].

A.1 Mouse

Figura A.1: O *mouse*

Dois princípios são normalmente utilizados na construção de *mouses* : os mecânicos e os óticos. Os mecânicos captam os movimentos por meio de esferas ou rodas. Os óticos tem como princípio de funcionamento a reflexão de luz.

O movimento de controle de um *mouse*, tanto nos mecânicos como nos óticos, consiste na translação do mesmo sobre um plano de apoio e comumente do acionamento de botões. O usuário o translada através de movimentos da mão, do braço e ou do antebraço, de forma que o mesmo deslize sobre o plano (figura A.1). O plano não necessita ser especial, exceto no caso da tecnologia ser ótica. Entretanto, como o *mouse* não pode ser utilizado

como dispositivo para posicionamento absoluto que permite digitalizações, pois não possui um registro de todos os pontos, é comum a utilização de um plano simples de superfície de atuação do *mouse*.

O *mouse* é um dispositivo de posicionamento relativo. Sua precisão só é razoável quando o usuário o movimenta vertical ou horizontalmente e não quando o rotaciona. Pela própria construção do *mouse* não se captam de modo real os movimentos discretos verticais e horizontais relativos. Para que movimentos rotacionais sejam captados é necessário um ponto de referência ou origem e um registro de posições com respeito a essa origem, isto é, é necessário um dispositivo de posicionamento absoluto.

No *mouse* a relação entre a distância medida (do *mouse*) e a distância reportada (do cursor) pode ser controlada. Esta distância medida permite um controle do "passo" do cursor através do *software* (como realizado neste trabalho com o cursor 3D). O usuário pode ajustar os movimentos do cursor de acordo com as suas necessidades. Por exemplo, o "passo" do cursor para seleção de objetos não necessita ser tão preciso quanto para desenho à mão livre.

A.2 Mesa digitalizadora

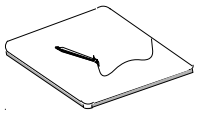

Figura A.2: A *mesa digitalizadora*

A mesa digitalizadora (*tablet*) é uma superfície plana que pode detetar a posição de uma caneta móvel que desliza sobre ela (figura A.2). A maior parte das mesas digitalizadoras faz uso de um mecanismo sensor elétrico para a determinação da posição da caneta. Uma grade de fios é disposta na superfície da mesa e pulsos eletrônicos são aplicados sequencialmente nas linhas e colunas da grade. Estes pulsos geram sinais eletromagnéticos que induzem fluxo de corrente num pequeno solenóide que fica dentro da caneta. Deste modo, a força da corrente induzida por cada pulso é usada para localização da caneta e também para estimar a distância entre a caneta e a mesa.

A.3 Trackball

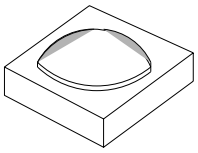

Figura A.3: O *trackball*

O *trackball* (figura A.3) é uma esfera apoiada numa superfície côncava. A ação do usuário resume-se à rolagem da esfera com a palma da mão. Como o *mouse*, o *trackball* é um dispositivo relativo, onde a variação das suas coordenadas são registradas baseadas em posições relativas, isto é, uma nova posição é calculada a partir de uma posição anterior. A esfera é conetada a potenciômetros que são responsáveis pela codificação dos movimentos.

O *trackball* é um dispositivo indireto, isto é, não é usado diretamente sobre superfície da tela. Pelo fato da esfera permitir dois ângulos de rotação e estar limitada a uma mesma área de movimento, estes dispositivos são indicados apenas para os casos de posicionamento. Para atividades como por exemplo, desenho, estes dispositivos oferecem pouca flexibilidade.

A.4 Joystick

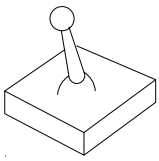

Figura A.4: O *joystick*

O *joystick* (figura A.4) é um dispositivo fixo e pode ser controlado com o movimento dos dedos, pulso ou antebraço. O que é uma vantagem quando se tem pouco espaço ou quando o ambiente se movimenta, como no interior de veículos. Para codificar os movimentos são usados potenciômetros. O *joystick* é fixo e são usadas molas para fazê-lo retornar do seu posicionamento central.

É difícil usar um *joystick* para controlar a posição de um cursor na tela porque o dispositivo é muito sensível. Por isso, o *joystick* é usado para controlar a velocidade de deslocamento.

Existem dois tipos de *joysticks*, os isotônicos e os isométricos. Os isotônicos são dispositivos móveis, onde as medidas são feitas em função do deslocamento angular de uma haste de controle. Nos isométricos, mede-se a força aplicada sobre uma haste rígida.

A.5 Canetas óticas e telas sensíveis ao toque

Canetas óticas (*light pens*) são canetas que detectam pulsos de luz na tela de visualização. Estão associadas a pacotes gráficos que registram os *pixels* correspondentes à posição da caneta. Como o *light pen* não registra coordenadas de pontos completamente pretos, há técnicas especiais para superar este problema: uma delas consiste em emitir cor azul escuro, por um tempo muito curto, na região de *pixels* pretos.

Tela sensível ao toque (*touch panel*) é uma nova tecnologia que permite ao usuário interagir diretamente com o dedo na tela de visualização. Há três tecnologias básicas na implementação desta técnica:

- LED's infravermelhos e fotosensores, dispostos nas extremidades da tela, formam uma grade sobre a tela. O toque sobre a tela interrompe raios de luz horizontais e verticais que permitem a localização.
- Ondas sonoras de alta frequência viajam horizontal e verticalmente, consecutivamente, nas extremidades da tela. O toque sobre a tela causa reflexão das ondas sonoras até a fonte. A distância entre o dedo e a extremidade da tela pode ser calculada pelo tempo de percurso da onda.
- Duas superfícies de materiais transparentes muito próximos formam a tela. Entre elas há uma camada de material condutor e uma camada de material resistivo. Quando o dedo pressiona a região da tela, a voltagem cai na camada resistiva e permite a localização.

Apêndice B

Manual de referência dos componentes de interface

A seguir apresenta-se neste apêndice o manual de referência para uso dos componentes do módulo IQLT. O arquivo "`iql_Triad`" contém o manual para uso da técnica de particionamento, o arquivo "`iql_Circ`" para uso da técnica dos movimentos circulares e o arquivo "`iql_Wkp`" para uso da técnica do plano de trabalho (*working plane*). Estes manuais foram gerados pelo utilitário *c2man* (versão 2.0).

O *XView* estendido executa no sistema operacional UNIX (SunOS 4.1.3) das estações *SPARCstation*. O ambiente de trabalho utilizado foi o *OpenWindows*.

Apêndice C

Manual de programação dos componentes de interface

C.1 Componente da técnica dos movimentos circulares no *XView*

```
/*-----------------------------------------------*/
/*    Uso do componente dos mov. circulares      */
/*-----------------------------------------------*/
#include "Iql.h"
#include "IqlDrv.h"
#include "IqlEvn.h"
void iql_MovCirc(window)
{
     IQL_3D_COORDINATES *cursor3D; /* Valor (x,y,z) no espaco */
     FRAME_POINT        *cursor2D; /* Valor (x,y) do mouse */
     IQL_FRAME          *frame;    /* Frame IQL */
     IQL_HANDLE         *handle;
     FRAME_ID           frame_id;
     IQL_CAMERA         *camera;

     /* Inicializacao do IQL */
     iql_init(handle, iql_handle);

     /* identificacao do frame */
     frame_id = (FRAME_ID)xv_get((Frame)window, XV_KEY_DATA, CANVAS_KEY);

     /* Funcao de criacao do frame IQL */
     frame = (IQL_FRAME *)iql_FrameCreate(handle, &frame_id, &desc);
```

```
    /* Inicializacao da camera */
    camera = iql_getCameraOfFrame(frame);

    /* Inicializa os dados de visualizacao da camera */
    iql_setEyePoint(camera, ABSOLUTE, 4.0, 60.0, 30.0);

    /* Dimensoes da "viewport" */
    iql_setViewPort(camera, ABSOLUTE, 50.0, 50.0, -1.0, 1.0);

    /* Angulo "tilt" da camera (em graus) */
    iql_setTiltAngle(camera, ABSOLUTE, 0.0);

    /* Foco da camera */
    iql_setFocus(camera, ABSOLUTE, 1.0);

    /* Tipo de visualizacao */
    iql_setViewType(camera, PARALLEL);

    /* Matriz de visualizacao */
    iql_UpdateViewMatrix(camera);

    while(1){
    case EnterNotify:
       break;
    case LeaveNotify:
       break;
    case MotionNotify:
       /* Cursor em movimento */
       cursor2D_new.x=event_x(event);
       cursor2D_new.y=event_y(event);
       /* Tecnica dos movimentos circulares */
       iql_Circ(handle,frame_id,frame, &cursor2D, &cursor3D);
       break;
  }
```

C.2 Componente da técnica de particionamento no *XView*

```
/*-----------------------------------------------*/
/*     Uso do componente particionamento         */
/*-----------------------------------------------*/
#include "Iql.h"
#include "IqlDrv.h"
```

```
#include "IqlEvn.h"
void iql_PartEspImg(window)
{
     IQL_3D_COORDINATES *cursor3D; /* Valor (x,y,z) no espaco */
     FRAME_POINT *cursor2D;        /* Valor (x,y) do mouse */
     IQL_FRAME          *frame;    /* Frame IQL */
     IQL_HANDLE         *handle;
     FRAME_ID           frame_id;
     IQL_CAMERA         *camera;

     /* Inicializacao do IQL */
     iql_init(handle, iql_handle);

     /* identificacao do frame */
     frame_id = (FRAME_ID)xv_get((Frame)window, XV_KEY_DATA, CANVAS_KEY);

     /* Funcao de criacao do frame IQL */
     frame = (IQL_FRAME *)iql_FrameCreate(handle, &frame_id, &desc);

     /* Inicializacao da camera */
     camera = iql_getCameraOfFrame(frame);

     /* Inicializa os dados de visualizacao da camera */
     iql_setEyePoint(camera, ABSOLUTE, 4.0, 60.0, 30.0);

     /* Dimensoes da "viewport" */
     iql_setViewPort(camera, ABSOLUTE, 50.0, 50.0, -1.0, 1.0);

     /* Angulo "tilt" da camera (em graus) */
     iql_setTiltAngle(camera, ABSOLUTE, 0.0);

     /* Foco da camera */
     iql_setFocus(camera, ABSOLUTE, 1.0);

     /* Tipo de visualizacao */
     iql_setViewType(camera, PARALLEL);

     /* Matriz de visualizacao */
     iql_UpdateViewMatrix(camera);



   while(1){
     case EnterNotify:
        break;
     case LeaveNotify:
```

```
        break;
     case MotionNotify:
        /* Cursor em movimento */
        cursor2D_new.x=event_x(event);
        cursor2D_new.y=event_y(event);
        /* Tecnica de particionamento do espaco imagem */
        iql_Triad(handle,frame_id,frame, &cursor2D, &cursor3D);
        break;
   }
}
```

C.3 Componente da técnica do plano de trabalho no *XView*

```
/*-----------------------------------------------*/
/*    Uso do componente plano de trabalho       */
/*-----------------------------------------------*/
#include "Iql.h"
#include "IqlDrv.h"
#include "IqlEvn.h"
void iql_PlanoTrab(window)
{
     IQL_WKP_CONTEXT    wkp_type;
     IQL_WKP_DRAWTYPES  draw_type;
     IQL_3D_COORDINATES Wkpoints[3];
     INT                num;
     IQL_FRAME          *frame;    /* Frame IQL */
     IQL_HANDLE         *handle;
     FRAME_ID           frame_id;
     IQL_CAMERA         *camera;

     /* Inicializacao do IQL */
     iql_init(handle, iql_handle);

     /* identificacao do frame */
     frame_id = (FRAME_ID)xv_get((Frame)window, XV_KEY_DATA, CANVAS_KEY);

     /* Funcao de criacao do frame IQL */
     frame = (IQL_FRAME *)iql_FrameCreate(handle, &frame_id, &desc);

     /* Inicializacao da camera */
     camera = iql_getCameraOfFrame(frame);

     /* Inicializa os dados de visualizacao da camera */
     iql_setEyePoint(camera, ABSOLUTE, 4.0, 60.0, 30.0);
```

```
    /* Dimensoes da "viewport" */
    iql_setViewPort(camera, ABSOLUTE, 50.0, 50.0, -1.0, 1.0);

    /* Angulo "tilt" da camera (em graus) */
    iql_setTiltAngle(camera, ABSOLUTE, 0.0);

    /* Foco da camera */
    iql_setFocus(camera, ABSOLUTE, 1.0);

    /* Tipo de visualizacao */
    iql_setViewType(camera, PARALLEL);

    /* Matriz de visualizacao */
    iql_UpdateViewMatrix(camera);

    /* Seta uso do working plane */
    dr_set_working(frame_id, 1);

    /* Tecnica do plano de trabalho */
    iql_Wkp(handle, frame_id, 10, WKP_ON_GRID, WKP_FILL, Wkpoints);
}
```

Apêndice D

Descrição dos *Drivers*

D.1 *Drivers* como interfaces entre IQLT e outros sistemas de construção de interfaces

Os *drivers* implementados para comunicação da caixa de ferramentas IQL**T** com os construtores de interface podem ser classificados de acordo às funções que realizam como:

1. *Drivers* inicializadores da gerenciação da caixa IQL**T**
   - `dr_init_iqlHandle()`: *Driver* inicializador da gerenciação da caixa IQL**T**.
   - `dr_get_iqlHandle()`: *Driver* que retorna o gerenciaor da caixa IQL**T**.
2. *Drivers* gerenciadores de eventos da caixa IQL**T**: Estes *drivers* controlam a informação a respeito das janelas IQL**T** (janelas de acesso tridimensional).
   - `dr_get_iqlList()`: *Driver* que obtém a primeira janela IQL**T** de uma lista.
   - `dr_succ_iqlList()`: *Driver* que obtém a janela IQL**T** sucessora da lista.
   - `dr_screen_num()`: *Driver* que retorna o número de janelas IQL**T**'s da lista.
   - `dr_get_dpy()`: *Driver* que retorna a janela IQL**T** que esta sendo exibida.
3. *Drivers* gerenciadores de eventos de cada janela IQL**T**: estes *drivers* controlam a manipulação individual de cada janela IQL**T**.

- `dr_first_create_frame()`: *Driver* que cria a primeira janela IQL**T** dando-lhe um identificador e a inclui numa lista de janelas.
- `dr_create_frame()`: *Driver* que cria uma janela IQL**T** dando-lhe um identificador e a inclui numa lista de janelas.
- `dr_close_frame()`: *Driver* que retira uma janela IQL**T** da lista, destruindo a estrutura de dados correspondentes a essa janela IQL**T**.
- `dr_getframe()`: *Driver* que retorna a janela IQL**T** dado o seu identificador.
- `dr_frm_repaint()`: *Driver* que reconfigura a janela IQL**T**.

4. *Drivers* que manipulam características específicas de cada janela IQL**T**: considerando que cada janela IQL**T** está composta de duas áreas específicas:

   (a) A base da janela IQL**T** e

   (b) A área de desenho da janela IQL**T**

   As características geométricas (ou aparência) da base e área de desenho de cada janela IQL**T** variará de acordo com construtor de interface utilizado. Sendo assim, os drivers listados a seguir terão como função comunicar as caraterísticas geométricas que cada janela haverá de tomar. Entre os *drivers* que comunicam as características geométricas da base da janela IQL**T** temos:

   - `dr_init_window()`: *Driver* que determina um identificador para a base da janela.
   - `dr_win_inqrefpoint()`: *Driver* que retorna os pontos iniciais da base da janela.
   - `dr_frame_p_win()`: *Driver* que retorna dimensões da base da janela e pontos iniciais da base.
   - `dr_win_getxid()`: *Driver* que devolve o identificador da base da janela.

   Entre os *drivers* que comunicam as características geométricas da área de desenho da janela IQL**T** temos:

   - `dr_set_desc()`: *Driver* que "seta" dimensões da área de desenho da janela IQL**T**.
   - `dr_get_desc()`: *Driver* que retorna as dimensões da área de desenho da janela IQL**T**.

- `dr_update_desc()`: *Driver* que atualiza as dimensões e a origem da área de desenho da janela IQL**T**.
- `dr_get_coord()`: *Driver* que retorna as coordenadas iniciais da área de desenho da janela IQL**T**.
- `dr_find_frame_elem()`: *Driver* que retorna numa estrutura todos os dados correspondentes à área de desenho da janela IQL**T**.

Referências Bibliográficas

[Bank92] **David Banks**. Interactive manipulation and display of two-dimensional surfaces in four-dimensional space. *Computer Graphics*, pp. 197–207, 1992. Proceedings of the ACM SIGGRAPH'92.

[Bier86] **Eric A. Bier**. Skitters and jacks: Interactive 3D positioning tools. In *Proceedings 1986 Workshop on Interactive 3D Graphics*, pp. 183–196, Chapel Hill, North Carolina, New York, outubro 1986.

[Bier89] **Eric A. Bier**. Snap-dragging: Interactive geometric design in two and three dimensions. Master's thesis, University of California, Berkeley & Xerox Corporation, setembro 1991.

[Bier90] **Eric A. Bier**. Snap-dragging in three dimensions. *Computer Graphics*, v. 24, n. 4, pp. 193–204, março 1990. Proceedings of the 1990 Symposium on Interactive 3D Graphics.

[Chen88] **Michael Chen e Joy Mountford**. A study in interactive 3D rotation using 2D control devices. *Computer Graphics*, v. 22, n. 4, pp. 121–129, agosto 1988. Proceedings of ACM SIGGRAPH'88.

[Conn92] **D. Brookshire Conner et al.** Three-dimensional widgets. *Computer Graphics*, v. 25, n. 2, pp. 183–188, março 1992. Proceedings of the 1992 Symposium on Interactive 3D Graphics, ACM SIGGRAPH'92. Brown University, *http://www.cs.brown.edu/research/graphics/publications.html*.

[DEBU90] **Sun Microsystems**. *Debugging Tools Manual*. Sun Microsystems, Inc., USA, 1990.

[Diam94] **Marcelo Diamand**. *Sistema para visualização holográfica de figuras geradas por computador*. Tese de Mestrado, Universidade Estadual de Campinas - UNICAMP, dezembro 1994.

[Doug94] **Sarah A. Douglas e Arant K. Mithal**. The effect of reducing homing time on the speed of a finger-controlled isometric pointing device. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, pp. 411–416, Massachussets, abril 1994. Addison Wesley.

[Emme90] **M. van Emmerik**. A direct manipulation technique for specifying 3D object transformations with a 2D input device. *Computer Graphics Forum 9*, pp. 355–361, 1990.

[Evan81] **Kenneth Evans B., Peter P. Tanner, e Marceli Wein**. Tablet-based valuators that provide one, two, or three degrees of freedom. *Computer Graphics*, v. 15, n. 3, pp. 91–97, agosto 1981. Proceedings of ACM SIGGRAPH'81.

[Fole90] **James D. Foley, Andries van Dam, e Steven Feiner**. *Computer Graphics: Principles and Practice*. Addison-Wesley, USA, 2. ed., 1990.

[Furu92] **C.A. Furuti e R. Drummond**. Stardust: Editor gráfico. *VI Simposio Brasileiro de Engenharia de Software*, 1992. Gramado - RS.

[Hage91] **Margaret A. Hagen**. How to make a visually realistic 3D display. *Computer Graphics*, v. 25, n. 2, pp. 77–81, abril 1991.

[Hans90] **Andrew J. Hanson**. The rolling ball: Context-free control of spatial orientation with two-dimensional input devices, novembro 1990. Computer Science Departament - Indiana University - Technical Report. *http://www.cs.indiana.edu/graphics/papers.html*.

[Hans94] **Andrew J. Hanson**. Rotations for N-dimensional graphics, junho 1994. Computer Science Departament - Indiana University - Technical Report. *http://www.cs.indiana.edu/graphics/papers.html*.

[Hell90] **D. Heller**. *XView Programming Manual*, volume 7. O'Reilly & Associates, Inc., Massachusetts Institute of Technology, 1990.

[Hern92] **K. P. Herndon et al.** Interactive shadows. In *Proceedings of the ACM UIST'92 Symposium on User Interface Software and Technology*, pp. 1–6, California, novembro 1992. *http://www.cs.brown.edu/research/graphics/publications.html*.

[Hix93] **Deborah Hix e H. Rex Hartson**. *Developing User Interfaces.* John Weley & Sons, Inc., USA, 1. ed., 1993.

[Houd92] **Stephanie Houde**. Interactive design of an interface for easy 3D direct manipulation. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, pp. 135–142, maio 1992.

[IQL90] **Fritz Loseries**. Functional interface of the IQL frame type of PRODIA/11, 1990. FhG-IGD – Relatório Interno.

[Jian94] **Jiandong Liang e Mark Green**. JDCAD: A highly interactive 3D modeling system. *Computer & Graphics*, v. 18, n. 4, pp. 499–505, julio/agosto 1994. Department of Computing Science, University of Alberta, Canada. *ftp://ftp.cs.ualberta.ca/pub/graphics/papers.*

[Kauf90] **A. Kaufman, R. Yagel, e R. Bakalash**. Direct interaction with a 3D volumetric environment. *Computer Graphics*, pp. 33–34, 1990.

[Merk91] **Luiz Ernesto Merkle**. *Guaiá: Um dispositivo sensor de três graus de liberdade para posicionamento no plano.* Tese de Mestrado, CEFET: Centro Federal de Educação Tecnológica do Paraná. Curso de pós graduação em Informática Industrial, dezembro 1991.

[Moti89] **Open Software Foundation**. *OSF/Motif Style Guide.* Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[Myer92] **Brad A. Myers e Mary B. Rosson**. Survey on user interface programming. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, pp. 195–202, Monterrey - California, maio 1992. Addison Wesley.

[Newm79] **William M. Newman**. *Principles of Interactive Computer Graphics.* McGraw-Hill, Xerox Corporation - USA, 2. ed., 1979.

[Niel86] **G. M. Nielson e D.R. Jr. Olsen**. Direct manipulation techniques of 3D objects using 2D locator devices. In *Proceedings 1986 Workshop on Interactive 3D Graphics*, pp. 175–182, Chapel Hill, North Carolina, New York, outubro 1986.

[Nye90] **Adrian Nye**. *Xlib Programming Manual.* O'Reilly & Associates, Inc., Massachusetts Institute of Technology, 1990.

[Open89] **Sun Microsystems**. *OPEN LOOK Graphical User Interface Application Style Guidelines*. Addison Wesley, Montain View, CA, 1989.

[Osbo92] **James R. Osborn e Alice M. Agogino**. An interface for interactive spatial reasoning and visualization. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, pp. 75–82, maio 1992.

[PHIG90] **Sun Microsystems**. *SunPHIGS 1.2 Programming Guide*. Sun Microsystems, Inc., USA, 1990.

[Phil92] **Cary B. Phillips, Norman I. Badler, e John Granieri**. Automatic viewing control for 3D direct manipulation. *Computer Graphics*, pp. 71–74, março 1992. Symposium on Interactive 3D Graphics ACM SIGGRAPH'92.

[PROD90] **D. Krömker, H. Steusloff, e H. Steubel**. PRODIA und PRODAT, dialog und datenbankschnittstellen für systemenentwurfswerkzeuge, 1990. Springer-Verlag, Heidelberg.

[Roge90] **David F. Rogers e Adams J. Alan**. *Mathematical Elements for Computer Graphics*. McGraw-Hill, USA, 2. ed., 1990.

[Rubi92] **Dean Rubine**. Combining gestures and direct manipulation. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, pp. 659–660, Monterrey - California, maio 1992. Addison Wesley.

[Shoe92] **K. Shoemake**. Arcball: A user interface for specifying three-dimensional orientation using a mouse. In *Proceedings of Graphicas Interface'92*, pp. 151–156, 1992.

[Silva95] **Elton José Silva**. *Representação em projeto de interfaces homem-computador: Estudo, aplicação e propostas de extensão do formalismo UAN*. Tese de Mestrado, Universidade Estadual de Campinas - UNICAMP, julho 1995.

[Sioc89] **Antonio C. Siochi e H. Rex Hartson**. Task-oriented representation of asynchronous user interfaces. In *Proceedings of ACM CHI'89 Conference on Human Factors in Computing Systems*, pp. 183–188. Addison Wesley, maio 1989.

[Slat92] **Mel Slater**. An algorithm to support 3D interaction on relatively low performance graphics systems. *Computer & Graphics*, v. 16, n. 3, pp. 311–315, 1992.

[Snib92] **S.S. Snibbe et al.** Using deformations to explore 3D widget design. *Computer Graphics*, v. 26, n. 2, pp. 351–352, julho 1992. Proceedings of the ACM SIGGRAPH'92. *http://www.cs.brown.edu/research/graphics/publications.html.*

[Stap93] **Loretta Staples**. Representation in virtual space: Visual convention in the graphical user interface. In *Proceedings of ACM CHI'93 Conference on Human Factors in Computing Systems*, pp. 348–355, Amsterdam, abril 1993. Addison Wesley.

[Vela93] **Perla Velásquez e Wu Shin Ting**. Canvas 3D: Um novo componente de interface no XView. *VI Sibgrapi - Comunicação*, novembro 1993.

[Veno93] **Dan Venolia**. Facile 3D direct manipulation. In *Proceedings of ACM CHI'93 Conference on Human Factors in Computing Systems*, pp. 348–355, Amsterdam, abril 1993. Addison Wesley.

[Ware88] **Colin Ware e Danny R. Jessome**. Using the bat: A six-dimensional mouse for object placement. *IEEE Computer Graphics & Applications*, v. 8, n. 6, pp. 65–70, novembro 1988.

[Ware90] **Colin Ware e Steven Osborn**. Exploration and virtual camera control in virtual three dimensional environments. *Computer Graphics*, v. 24, n. 4, pp. 175–183, março 1990.

[XWin86] **Robert W. Scheifler e Jim Gettys**. The x-window system. *ACM Transactions on Graphics*, v. 5, n. 2, pp. 79–109, abril 1986.

[Zele93] **Zeleznik Robert C. et al.** An interactive 3D toolkit for constructing 3D widgets. In *Computer Graphics, Anual Conference Series*, pp. 81–84, agosto 1993. Proceedings of the ACM SIGGRAPH'93. *http://www.cs.brown.edu/research/graphics/publications.html.*

[Zhai94] **Shumin Zhai, William Buxton, e Paul Milgram**. The "silk cursor" : Investigating transparency for 3D acquisition. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, pp. 459–464, Massachussets, abril 1994. Addison Wesley.