Uma Taxonomia para Manipulação Interativa e Visualização de Objetos 3D

BEATRIZ CASTIER
LUIZ FERNANDO MARTHA
MARCELO GATTASS

Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Tecgraf – Grupo de Tecnologia em Computação Gráfica
Rua Marquês de São Vicente 225, Gávea
22453-900 Rio de Janeiro, RJ, Brasil
{bia,lfm,gattass}@tecgraf.puc-rio.br

Artigo apresentado e publicado no Sibgrapi'94 - VII Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens, Curitiba, PR, Nov. 1994, pp. 149-156.

Abstract. The paper describes a taxonomy for interactive manipulation of 3D objects using a 2D input device such as a mouse. The usual set of visualization parameters, eye position, reference point, and view up-vector, is adopted. The proposed classification divides the control models in three groups: Screen-based, Walk-through, and Object-based models. For each one of these models, translation and rotation geometric transformations may be performed. Independently, one may control other viewing parameters such clipping plane positions with respect to the eye point, and distance from eye to reference point. Existing manipulation schemes are analyzed according to the proposed classification. Basic screen-based and walk-through models are presented. In addition, it is proposed a model for object-based manipulation which is ideal for objects which present a natural bounding box, such as geological units. A companion paper summarizes the geometric transformations which are necessary for displaying a 3D environment on a screen, and which are based on the adopted visualization parameters.

1 Introdução

A visualização de blocos no maciço tectônico representados como subdivisões espaciais através de estruturas de dados topológicas *non-manifold* [Cavalcanti et al. (1992)] é uma etapa importante do entendimento de formações geológicas para atividades na indústria do petróleo. Este tipo de representação propicia uma flexibilidade muito grande tanto da modelagem quanto da visualização da formação geológica através de planos de corte. Esta visualização, entretanto, requer uma manipulação eficiente de objetos tridimensionais através de dispositivos de entrada tais como botões, potenciômetros e *mouse*.

Trabalhos como os de Chen et al. (1988), Emmerik (1990) e Shoemake (1992) apresentam modelos de manipulação interativa para a visualização de objetos 3D. A escolha de modelos para manipulação, entretanto, é complexa pois envolve, além de procedimentos algébricos que controlam a projeção e o movimento, a interface com a visão 3D do usuário. Para analisar estes trabalhos é necessário que primeiramente se estabeleçam taxonomias. Neste artigo é proposta uma taxonomia para classificar os diversos modelos de manipulação e, assim, criar subsídios para o desenvolvimento de novos tipos de controle.

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

A partir da taxonomia proposta para os modelos de manipulação, procura-se analisar alguns algoritmos existentes. Também são propostos dois modelos simples de manipulação, que podem ser facilmente implementados, e que resultam em grande eficiência para rodar e mover um objeto com relação aos eixos de tela e para simular a movimentação da câmera dentro de um ambiente.

Finalmente, propõe-se um modelo de manipulação, baseado na orientação corrente do objeto na tela, que é ideal para objetos que, a exemplo de unidades geológicas tridimensionais, se apresentam como uma caixa que envolve o domínio a ser simulado. A manipulação de visualização utilizando este modelo para um objeto de forma genérica pode ser feita através de sua caixa envolvente.

Este trabalho fica complementado por um outro artigo dos mesmos autores [Martha (1994)], onde são resumidas as transformações geométricas que um ponto do modelo sofre para ser visualizado na tela. Estas

transformações estão baseadas nos mesmos parâmetros de visualização adotados aqui.

2 Parâmetros de visualização em 3D

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

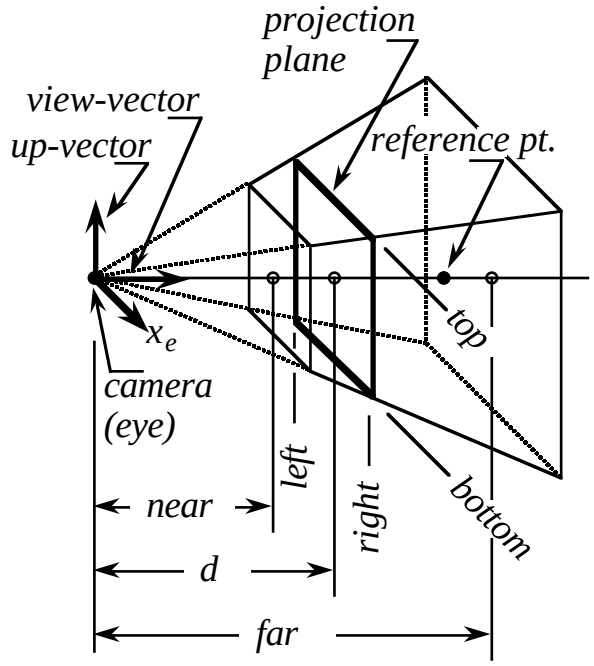

Figura 1: Modelo de câmera e frustum de visão.

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

- posição da câmera (olho): (eye_X, eye_Y, eye_Z)
- posição do ponto de referência (um ponto no espaço de modelagem para onde a câmera mira): (ref_X, ref_Y, ref_Z)
- vetor de orientação vertical da câmera (*view up-vector* – ***vup***): (vup_X, vup_Y, vup_Z).

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

$\boldsymbol{view} = (ref_X, ref_Y, ref_Z) - (eye_X, eye_Y, eye_Z)$

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

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

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

O frustum de visão (Figura 1) pode ser definido de diversas maneiras, e neste trabalho ele é definido pelos limites da janela de visão (*left*, *right*, *bottom*, *top*) em um plano de projeção no sistema de coordenadas do olho e pelas distâncias dos planos de cerceamento anterior (*near*) e cerceamento posterior (*far*) ao olho, no sentido do vetor de visão. O **plano de projeção** corresponde à janela na tela do computador e é definido pela sua distância (*d*) ao olho, também no sentido do vetor de visão. Os planos de cerceamento anterior e posterior e o plano de projeção são perpendiculares ao eixo z_e e situam-se do seu lado negativo. Deve-se observar que, enquanto os parâmetros da janela de visão são abscissas com valores reais negativos ou positivos, os parâmetros *far*, *near* e *d* são por definição distâncias positivas.

3 Movimentos básicos da câmera

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

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

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

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

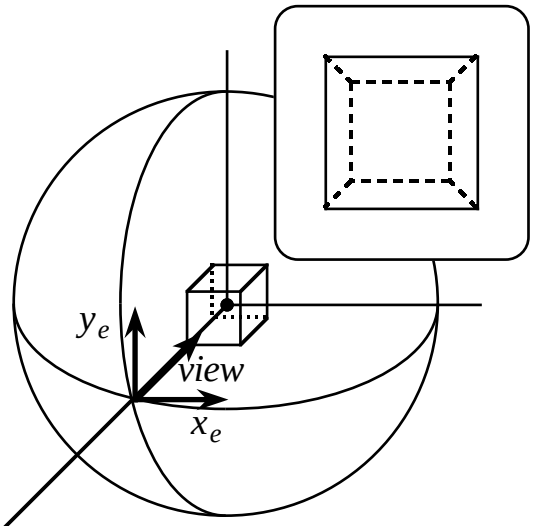

Figura 2: Um objeto simples visto pela posição inicial da câmera.

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

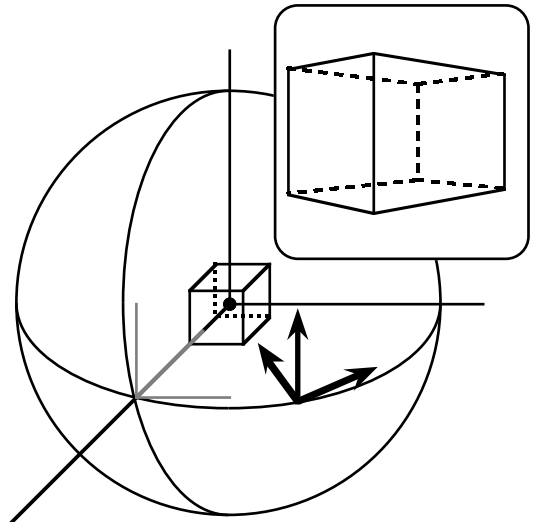

Figura 3: Rotação da câmera em torno do eixo vertical.

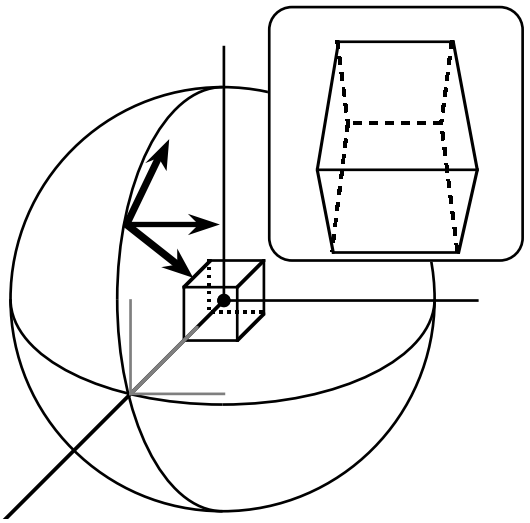

Figura 4: Rotação em torno do eixo horizontal.

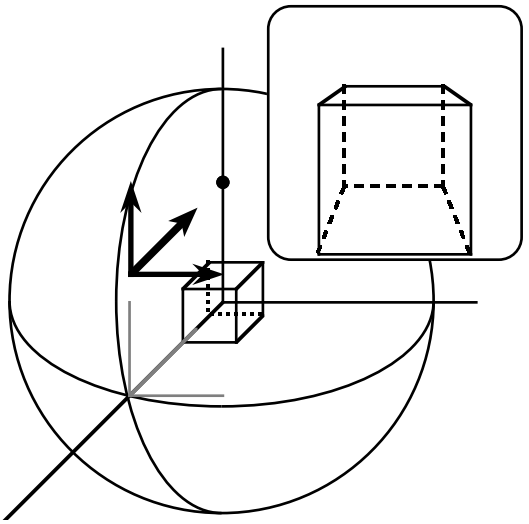

Figura 5: Translação vertical da câmera.

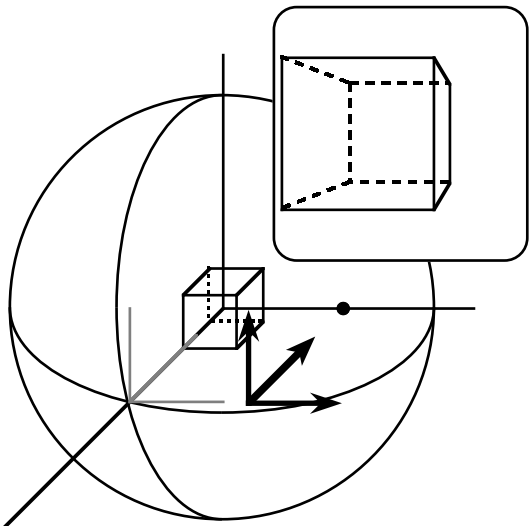

Figura 6: Translação horizontal da câmera.

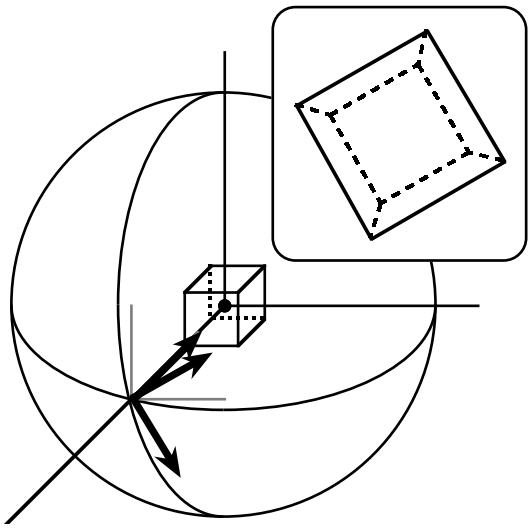

Figura 7: Rotação axial da câmera.

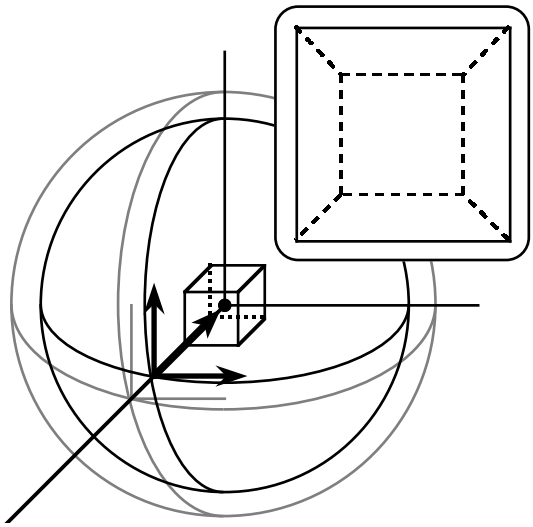

Figura 8: Translação radial da câmera.

4 Uma taxonomia para o controle de visualização em 3D

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

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

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

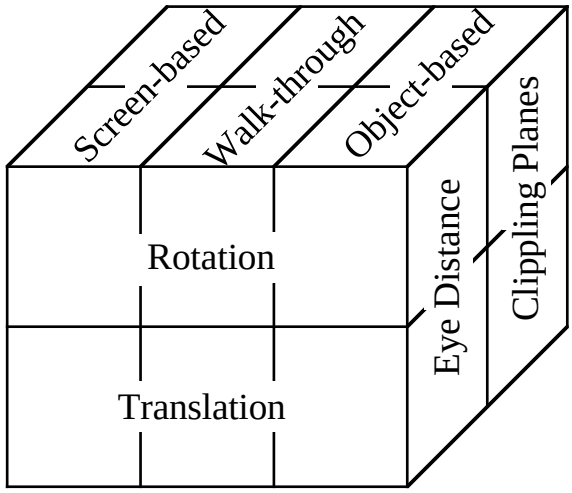

Figura 9: Uma taxonomia para o controle dos parâmetros de visualização.

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

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

A **esfera virtual** [Chen et al. (1988)] simula, com o uso de *mouse*, o mecanismo de um *trackball* capaz de girar livremente em torno de qualquer eixo do espaço 3D. O objeto é mostrado, na tela, dentro de um círculo

que representa a esfera virtual. Ao se rolar a esfera, é girado também o objeto. Para definir a rotação, o usuário deve fornecer dois pontos. Um movimento dentro do círculo define uma direção tangencial à rotação. Um movimento fora do círculo define uma rotação em torno do eixo z_e. A esfera virtual é um exemplo de manipulação *Screen-based*, pois é feita em relação aos eixos da tela e adota o centro da esfera como ponto de referência e centro de rotação.

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

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

5 Modelos básicos de manipulação

Nesta seção são propostos modelos simples do tipo *Screen-based* e *Walk-through*, baseados nos movimentos básicos de câmera apresentados na seção 3. A manipulação de distância da câmera ao ponto de referência (*Eye Distance*) e a manipulação de distância dos planos de cerceamento anterior e posterior (*Clipping Planes*) e de projeção ao olho são comuns a todos estes tipos de controle e, por serem simples manipulações unidimensionais de distâncias, não são abordadas neste trabalho.

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

Os parâmetros de visualização que são atualizados pelas transformações geométricas são a posição do olho (câmera), a posição do ponto de referência e o vetor de orientação vertical da câmera. Conforme foi mostrado anteriormente, estes parâmetros definem os vetores unitários da base de coordenadas do olho em relação ao sistema de coordenadas de modelagem: $\boldsymbol{x_e}$, $\boldsymbol{y_e}$, $\boldsymbol{z_e}$.

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

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

$\delta_x = m_x$ / *hsize*

$\delta_y = m_y$ / *vsize*,

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

$\delta_{\boldsymbol{m}} = (\delta_x, \delta_y)$.

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

$\Delta_x = \delta_x$ (*right* – *left*)

$\Delta_y = \delta_y$ (*top* – *bottom*).

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

$\Delta_{\boldsymbol{m}} = (\Delta_x, \Delta_y)$.

5.1 Modelo *Screen-based* básico

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$\boldsymbol{t} = \Delta_x \ \boldsymbol{x_e} + \Delta_y \ \boldsymbol{y_e}$.

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

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

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

Nota-se que com este este procedimento o objeto acompanha o movimento do *mouse* na tela.

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

$\boldsymbol{t} = \Delta \boldsymbol{z_e}$

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

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

5.2 Modelo *Walk-through* básico

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

6 Manipulação natural de objetos com caixa envolvente

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

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

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

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

Para implementar esta metáfora de manipulação, imaginou-se que cada aresta contém uma direção $\boldsymbol{s}$ tangente de giro que está contida em um plano perpendicular à aresta. Isto é ilustrado na Figura 10. Estas direções são tais que, nos seus sentidos positivos, fazem um giro que segue a regra da mão direita em torno do eixo principal paralelo à aresta correspondente. Neste trabalho optou-se por direções tangentes que fazem angulos de 45° com os planos adjacentes das arestas.

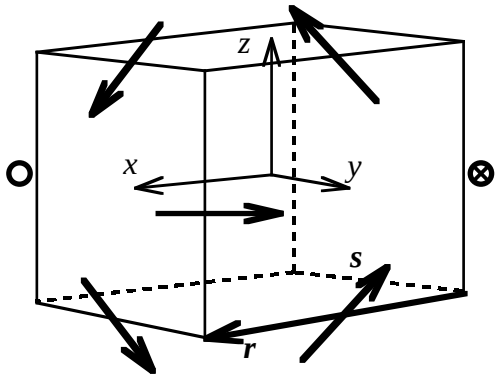

Figura 10: Arestas da caixa envolvente e a suas direções tangentes.

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

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

$\boldsymbol{v} = (v_x, v_y)$.

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

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

E o ângulo de rotação pode ser calculado em função do produto interno entre $\delta_{\boldsymbol{m}}$ e $\boldsymbol{v}$ através da expressão:

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

Nota-se que a rotação é ajustada para que um movimento do mouse de um extremo ao seu oposto na tela resulte em uma rotação total de 180°, a exemplo do que foi feito na seção 5.1.

Define-se o vetor $\boldsymbol{r}$ como o eixo de giro na direção da aresta selecionada (Figura 10) e $\boldsymbol{c}$ como o centro da caixa envolvente. Dessa forma, tem-se as transformações geométricas que as posições do olho e do ponto de referência e a direção vertical da câmera devem sofrer para implementar as rotações neste modelo:

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

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

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

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

do movimento desejado. Desta forma a direção de movimento está indicada pela normal $\boldsymbol{n}$ da face selecionada (Figura 11).

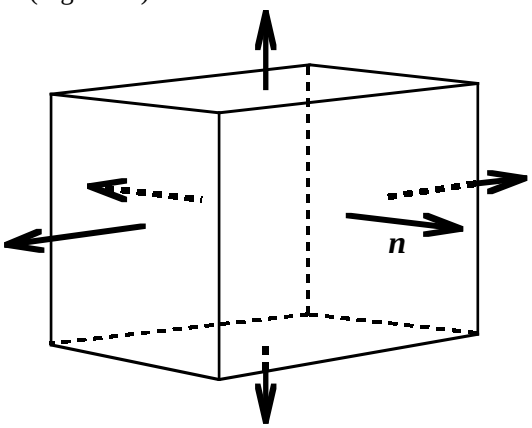

Figura 11: Normais das faces da caixa envolvente.

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

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

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

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

E o vetor de translação pode ser calculado em função do produto interno entre $\Delta_{\boldsymbol{m}}$ e $\boldsymbol{u}$ através da expressão:

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

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

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

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

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

7 Conclusões

Diversos modelos de manipulação foram implementados e testados em dois cursos de *Computação Gráfica Interativa* e um curso de *Interface com o Usuário* do Programa Interdisciplinar de Computação Gráfica da PUC-Rio. Destas experiências resultaram a taxonomia e a organização da álgebra das transformações propostas aqui. Nestes cursos foram feitas implementações com diversos ambientes: DOS para PCs, Xlib para *workstations* SUNs e GL para *worsktations* RISC 6000 da IBM. Para todas estas implementações, a metodologia proposta se mostrou simples e os alunos não tiveram maiores dificuldades em desenvolver seus trabalhos.

Sobre a taxonomia, pode-se afirmar que ela permitiu um melhor entendimento dos diversos métodos existentes na literatura e possibilitou o desenvolvimento dos modelos básicos de manipulação que são descritos neste trabalho. Nas avaliações preliminares feitas com manipulações de um objeto representado na tela, os modelos *Screen-based* e *Object-based* se mostraram mais eficientes quando a manipulação é feita através do *mouse*. O modelo *Screen-based* é melhor em uma movimentação livre e o *Object-based* em um ajuste mais preciso da posição do objeto na tela. Quando a manipulação é feita através do teclado, o modelo *Screen-based* se mostrou superior. O modelo *Walk-through* parece ser mais adequado quando vários objetos compõem o ambiente. Porém poucos testes foram feitos.

Para trabalhos futuros espera-se, com base na taxonomia e na metodologia proposta aqui, avaliar cada uma das manipulações apresentadas junto a grupos de usuários. O estudo de manipulações de objetos através de rotações e posicionamentos precisos necessários para a área de CAD também é um dos tópicos de interesse .

Agradecimentos

Este trabalho foi desenvolvido no ICAD – Laboratório de CAD Inteligente da PUC-Rio. O ICAD é suportado financeiramente pelo TeCGraf, Grupo em Tecnologia de Computação Gráfica da PUC-Rio, o que é feito através de projetos principalmente com o CENPES/Petrobrás e com o CEPEL/Eletrobrás. Os autores são gratos também a Paul (Wash) Wawrzynek pelas idéias e figuras sobre os controles básicos da câmera.

Referências

P.R. Cavalcanti, P.C.P. Carvalho, L.F., Martha, Criação e Manuteção de Subdivisões Espaciais, *Anais do SIBGRAPI V* (1992) 105-113.

M. Chen, S. Mountford, A. Sellen, A Study in Interactive 3-D Rotation Using 2-D Control Devices, *ACM Computer Graphics* **22**, (1988) 121-129.

M. van Emmerik, A Direct Manipulation Technique for Specifying 3D Object Transformations with a 2D Input Device, *Comp. Graph. Forum* **9** (1990) 355-361.

L.F. Martha, M. Gattass, Um Resumo das Transformações Geométricas para Visualização em 3D, *Anais do SIBGRAPI'94*.

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

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

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