Pontifícia Universidade Católica do Rio de Janeiro — PUC-Rio

TRANSPARÊNCIAS:

TRANSFORMAÇÕES GEOMÉTRICAS PARA VISUALIZAÇÃO EM 3D

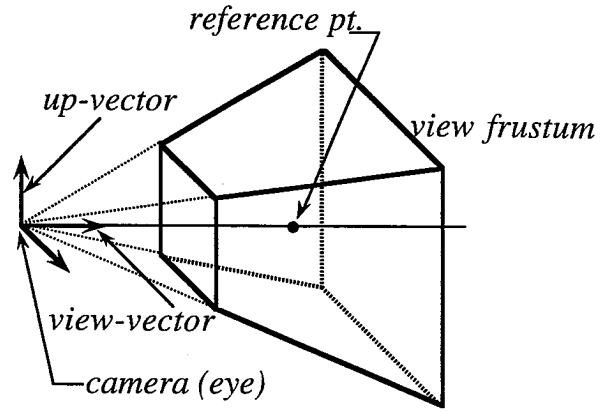
Luiz Fernando Martha

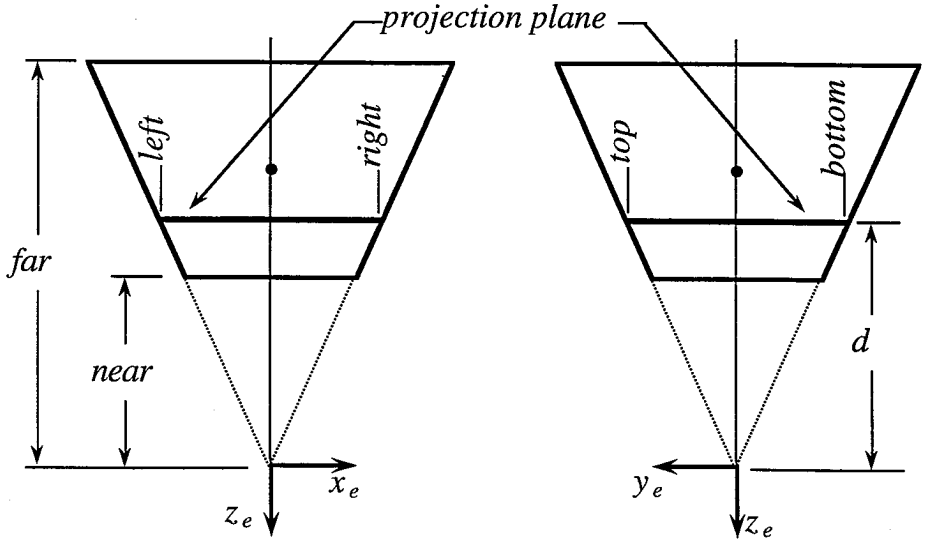
Referências

1. "Planar Geometric Projections and Viewing Transformations" – Carlbom, I.; Paciorek, J. – *ACM Computing Surveys*, Vol. 10, No. 4, Dec. 1978.
2. *OpenGL Programming Guide – The Official Guide to Learning OpenGL*, Release 1 – Neider, J.; Davis, T.; Woo, M. – Addison-Wesley Publishing Company, Reading, Massachusetts, 1993.
3. Manual IBM AIX Version 3 for RISC System/6000 – *Graphics Programming Concepts*, March 1990.

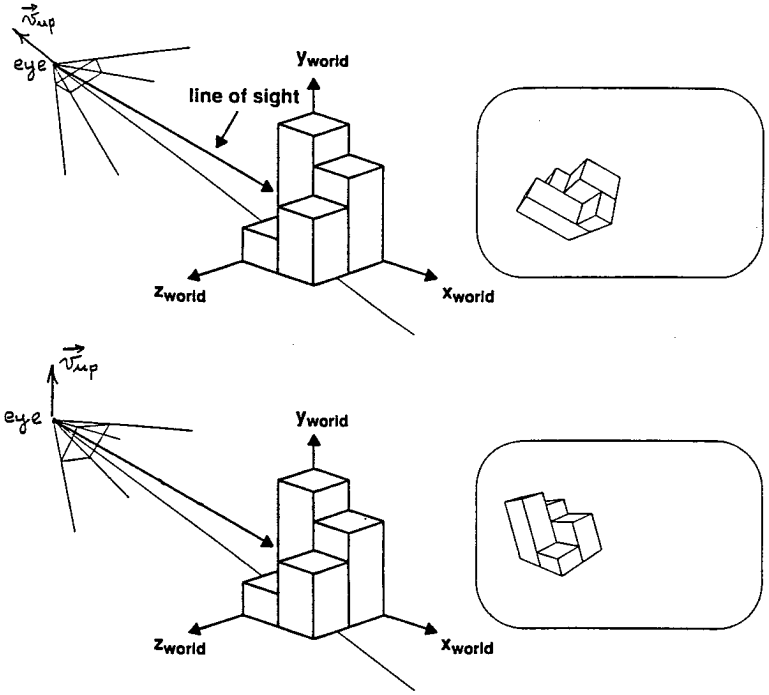
Rio de Janeiro, Novembro de 1994

Modelo de Câmera





Parâmetros de Visualização (Posicionamento da Câmera)



Parâmetros necessários:

- posição do olho → (eye_x, eye_y , eye_z) ⎫
- posição do ponto de referência → (ref_x, ref_y , ref_z) ⎬ Direção de visão
- orientação vertical da câmera → (vup_x, vup_y , vup_z) ⎭

Parâmetros de visualização em 3D

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)

Sistema de coordenadas do olho

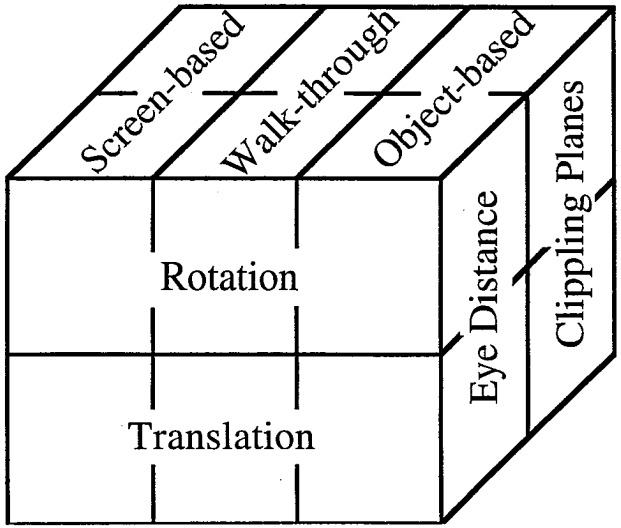
$\boldsymbol{view} = (ref_x, ref_y, ref_z) - (eye_x, eye_y, eye_z)$

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

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

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

Uma taxonomia para o controle de visualização em 3D



Controle da visualização através de movimentos do *mouse*

Movimento do *mouse* em *pixels*

$\boldsymbol{m} = (m_x , m_y)$

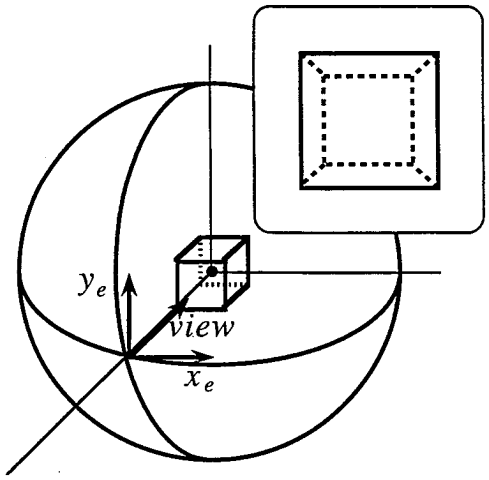
Movimento do *mouse* normalizado para controle de rotações

$\delta_x = m_x \,/\, hsize$
$\delta_y = m_y \,/\, vsize$
$\delta_{\boldsymbol{m}} = (\delta_x , \delta_y)$

Movimento do *mouse* normalizado para controle de translações

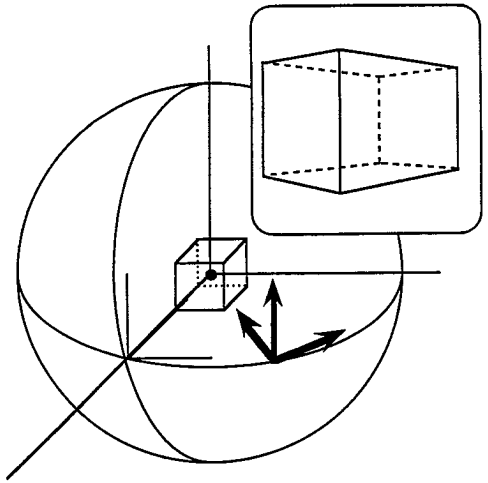
$\Delta_x = \delta_x \;\; (right - left)$
$\Delta_y = \delta_y \;\; (top - bottom)$
$\Delta_{\boldsymbol{m}} = (\Delta_x , \Delta_y)$

Posição inicial da câmera e imagem resultante

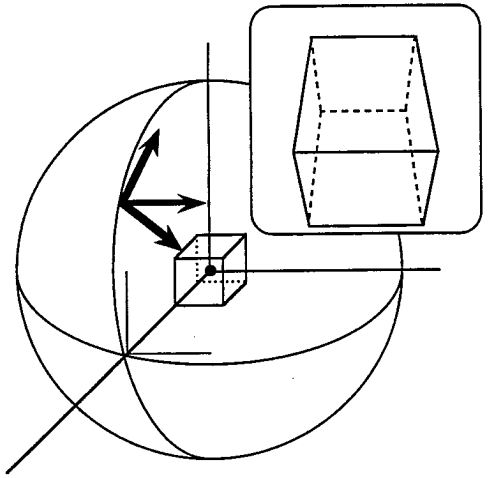


Controle de rotação tipo *Screen-based*

Movimento horizontal do *mouse*



Movimento vertical do *mouse*



Controle de rotação tipo *Screen-based* (cont.)

Eixo de rotação na tela

$r_m = (-\delta_y , \delta_x)$

Eixo de rotação no espaço do objeto

$r = \delta_x y_e - \delta_y x_e$

Ângulo de rotação
(movimento do *mouse* de um lado ao outro da janela $\Rightarrow 180°$)

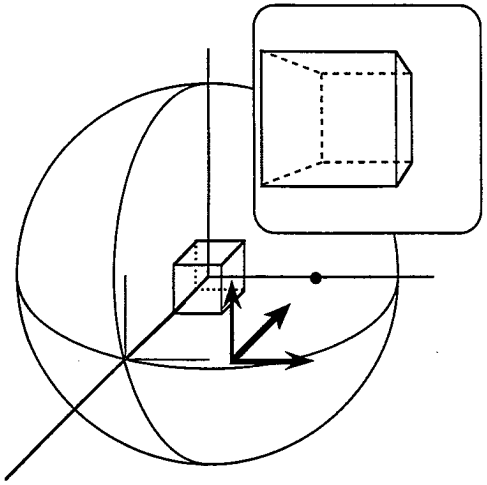
$angle = -180° \sqrt{\delta_x^2 + \delta_y^2}$

Atualização dos parâmetros de visualização

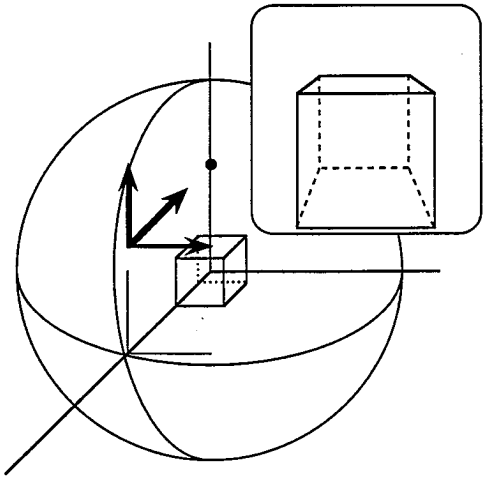
RotateAroundAxisAboutPoint($angle$, r_x, r_y, r_z, ref_x, ref_y, ref_z, eye_x, eye_y, eye_z)
RotateAroundAxisAboutOrigin($angle$, r_x, r_y, r_z, vup_x, vup_y, vup_z)

Controle de translação do tipo *Screen-based*

Movimento horizontal do *mouse*



Movimento vertical do *mouse*



Controle de translação tipo *Screen-based* (cont.)

Vetor de translação na tela

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

Vetor de translação no espaço do objeto

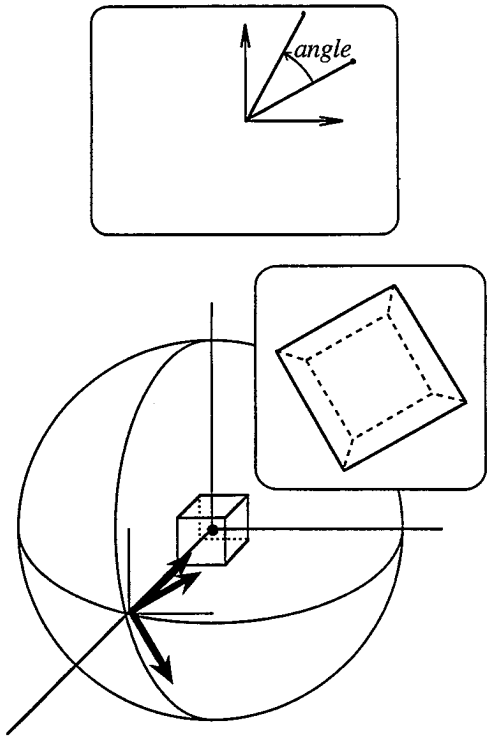
$t = \Delta_x x_e + \Delta_y y_e$

Atualização dos parâmetros de visualização

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 *)*

Controle de rotação axial (*spin*) tipo *Screen-based*

Ângulo de giro do *mouse* em relação ao centro da janela



Eixo de rotação no espaço do objeto

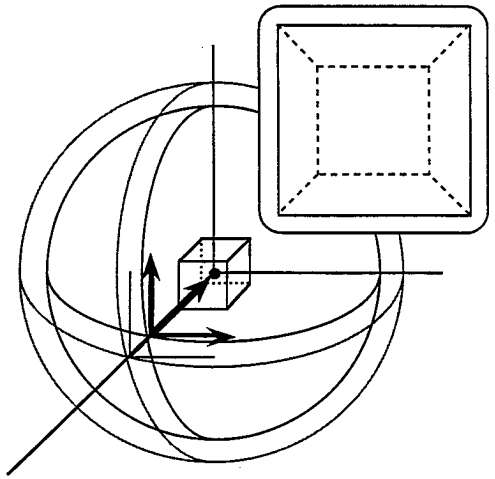
$r = z_e$

Atualização dos parâmetros de visualização

RotateAroundAxisAboutOrigin(−angle, r_x, r_y, r_z, vup_x, vup_y, vup_z)

Controle de translação radial tipo *Screen-based*

Movimento Δ do *mouse* na janela (usado como um potenciômetro)



Vetor de translação no espaço do objeto

$t = \Delta z_e$

Atualização dos parâmetros de visualização

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

Manipulação centralizada na câmera (*Walk-through*)

(É sempre o inverso da manipulação do tipo *Screen-based*)

Rotação horizontal e vertical

$r_m = (-\delta_y , \delta_x)$
$r = \delta_x y_e - \delta_y x_e$
$angle = -180° \sqrt{\delta_x^2 + \delta_y^2}$
RotateAroundAxisAboutPoint(angle, r_x, r_y, r_z, eye_x, eye_y, eye_z, ref_x, ref_y, ref_z)
RotateAroundAxisAboutOrigin(angle, r_x, r_y, r_z, vup_x, vup_y, vup_z)

Translação horizontal e vertical

$\Delta_m = (\Delta_x , \Delta_y)$
$t = \Delta_x x_e + \Delta_y y_e$
Translate(t_x, t_y, t_z, eye_x, eye_y, eye_z)
Translate(t_x, t_y, t_z, ref_x, ref_y, ref_z)

Rotação axial

(ângulo computado pelo giro do *mouse* em relação ao centro da janela)

$r = z_e$
RotateAroundAxisAboutOrigin(angle, r_x, r_y, r_z, vup_x, vup_y, vup_z)

Translação radial

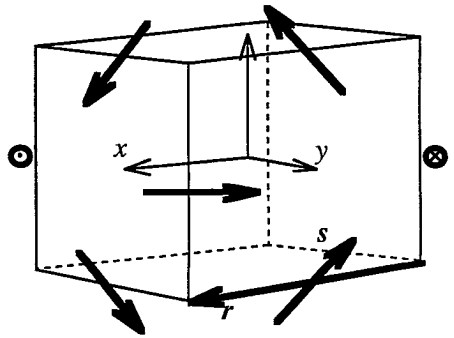
(movimento Δ do mouse na janela usado como um potenciômetro)

$t = \Delta z_e$
Translate(t_x, t_y, t_z, eye_x, eye_y, eye_z)
Translate(t_x, t_y, t_z, ref_x, ref_y, ref_z)
(neste caso o ponto de referência também é atualizado)

Manipulação centralizada no objeto proposta

(Ideal para objetos com caixa envolvente natural)

Rotação em torno de um eixo principal *r* da caixa envolvente



Vetor tangente *s* descrito no sistema de coordenadas do olho (*eye*)

$s_e = (s_{ex}, s_{ey}, s_{ez})$

Vetor *s* projetado na tela (plano de projeção) e normalizado

$v = (v_x, v_y)$
$v_x = s_{ex} / \sqrt{s_{ex}^2 + s_{ey}^2}$
$v_y = s_{ey} / \sqrt{s_{ex}^2 + s_{ey}^2}$

Ângulo de rotação em torno do centro *c* da caixa envolvente

(proporcional ao produto interno entre v e δ_m)

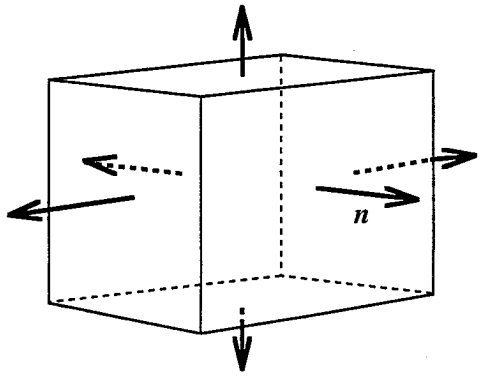
$angle = -180° \ (\delta_x v_x + \delta_y v_y)$

Atualização dos parâmetros de visualização

RotateAroundAxisAboutPoint(angle, r_x, r_y, r_z, c_x, c_y, c_z, eye_x, eye_y, eye_z)
RotateAroundAxisAboutPoint(angle, r_x, r_y, r_z, c_x, c_y, c_z, ref_x, ref_y, ref_z)
RotateAroundAxisAboutOrigin(angle, r_x, r_y, r_z, vup_x, vup_y, vup_z)

Manipulação centralizada no objeto proposta (cont.)

Translação na direção de uma normal n da caixa envolvente



Vetor normal n descrito no sistema de coordenadas do olho (*eye*)

$n_e = (n_{ex}, n_{ey}, n_{ez})$

Vetor n projetado na tela (plano de projeção) e normalizado

$u = (u_x, u_y)$

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

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

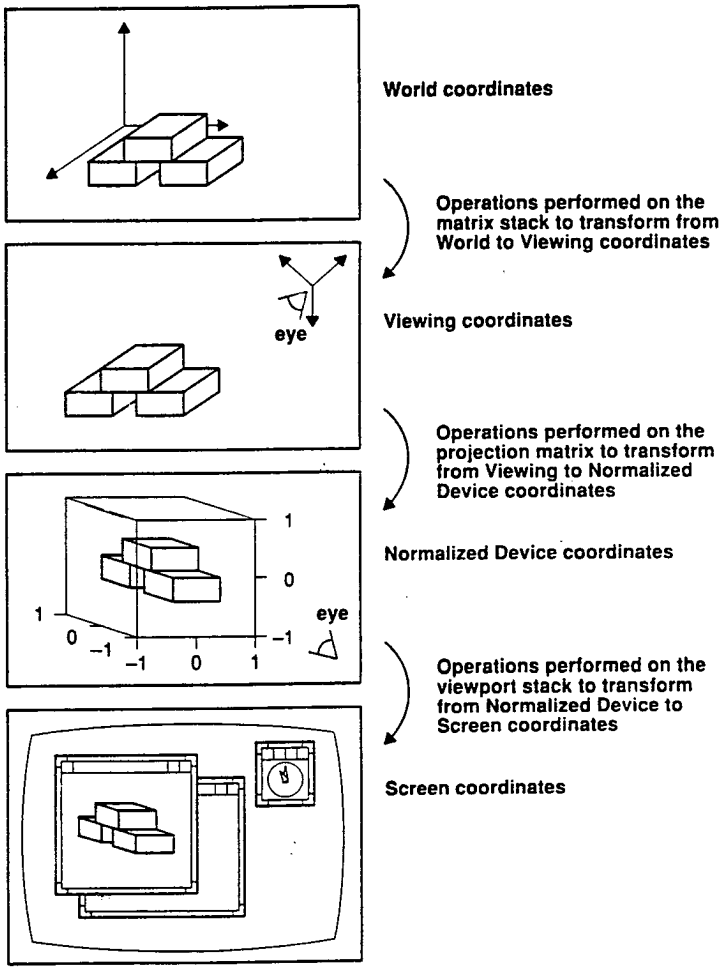
Vetor de translação no espaço do objeto
(proporcional ao produto interno entre u e Δ_m, na direção de n)

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

Atualização dos parâmetros de visualização

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

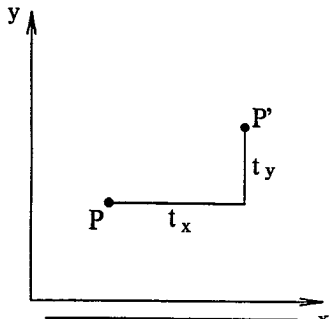
Transformações Geométricas para Visualização em 3D



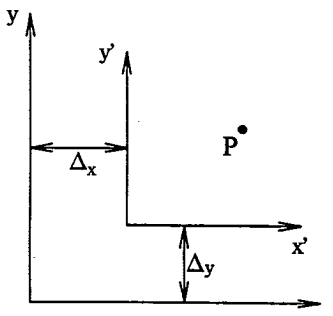
Coordinate Transformations

Transformações vs. Mudança de Base

Translação



Transformação de Translação



Mudança de Base por Translação

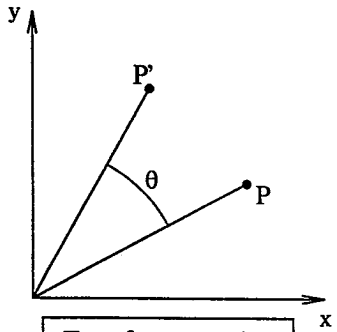
$$\begin{Bmatrix} x' \\ y' \end{Bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \end{Bmatrix} + \begin{Bmatrix} t_x \\ t_y \end{Bmatrix}$$

$$\begin{Bmatrix} x' \\ y' \end{Bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \end{Bmatrix} - \begin{Bmatrix} \Delta_x \\ \Delta_y \end{Bmatrix}$$

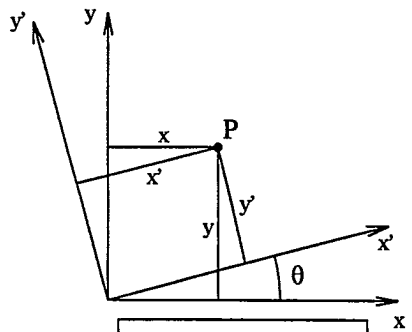
Conclusão

A mudança de base por translação pode ser vista como uma transformação de translação do ponto P de $(-\Delta_x,-\Delta_y)$.

Rotação



Transformação de Rotação



Mudança de Base por Rotação

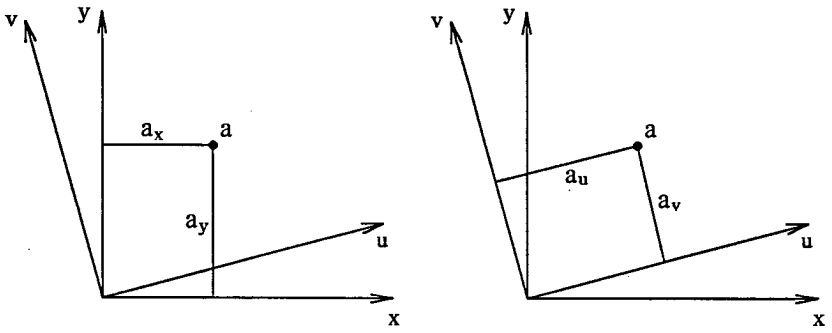
$$\begin{Bmatrix} x' \\ y' \end{Bmatrix} = \begin{bmatrix} \cos\theta & -\text{sen}\theta \\ \text{sen}\theta & \cos\theta \end{bmatrix} \begin{Bmatrix} x \\ y \end{Bmatrix}$$

$$\begin{Bmatrix} x' \\ y' \end{Bmatrix} = \begin{bmatrix} \cos\theta & \text{sen}\theta \\ -\text{sen}\theta & \cos\theta \end{bmatrix} \begin{Bmatrix} x \\ y \end{Bmatrix}$$

Conclusão

A mudança de base por rotação pode ser vista como uma transformação de rotação do ponto P de $(-\theta)$.

Mais sobre mudança de base por rotação



Transformação de Coordenadas por Mudança de Base

$$M:\ R^2 \rightarrow R^2$$

$$\begin{Bmatrix} a_x \\ a_y \end{Bmatrix} \rightarrow \begin{Bmatrix} a_u \\ a_v \end{Bmatrix} = [M] \begin{Bmatrix} a_x \\ a_y \end{Bmatrix} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{Bmatrix} a_x \\ a_y \end{Bmatrix}$$

Objetivo

Tentar escrever os vetores que inicialmente se referem à base "xy" na base "uv". Ou seja, deseja-se encontrar a matriz [M].

São conhecidos

$$\begin{Bmatrix} a_x \\ a_y \end{Bmatrix} \quad \text{e} \quad \begin{Bmatrix} \hat{u} \\ \hat{v} \end{Bmatrix} = \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix} \begin{Bmatrix} \hat{i} \\ \hat{j} \end{Bmatrix}$$

Vamos usar o fato

que [M] é ortogonal, isto é, $[M]^{-1} = [M]^T$

"É mais fácil achar $[M]^{-1}$ do que [M] porque nós conhecemos $\hat{u}$ e $\hat{v}$ em função de $\hat{i}$ e $\hat{j}$ e não o inverso"

Assim

$$\begin{Bmatrix} a_u \\ a_v \end{Bmatrix} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{Bmatrix} a_x \\ a_y \end{Bmatrix} \quad \text{e} \quad \begin{Bmatrix} a_x \\ a_y \end{Bmatrix} = \begin{bmatrix} m_{11} & m_{21} \\ m_{12} & m_{22} \end{bmatrix} \begin{Bmatrix} a_u \\ a_v \end{Bmatrix}$$

Se for considerado que

$$\vec{a} = a_x \hat{i} + a_y \hat{j} = \{ a_x \ \ a_y \} \begin{Bmatrix} \hat{i} \\ \hat{j} \end{Bmatrix} \quad \text{e} \quad \vec{a} = a_u \hat{u} + a_v \hat{v} = \{ a_u \ \ a_v \} \begin{Bmatrix} \hat{u} \\ \hat{v} \end{Bmatrix}$$

Então

$$\vec{a} = \{ a_u \ \ a_v \} \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix} \begin{Bmatrix} \hat{i} \\ \hat{j} \end{Bmatrix}$$

Portanto

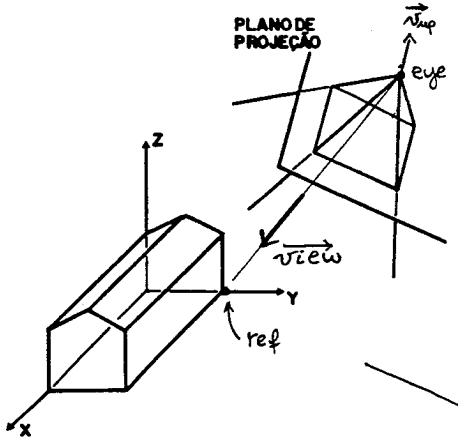
$$\{ a_x \ \ a_y \} = \{ a_u \ \ a_v \} \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix} \quad \text{ou melhor} \quad \begin{Bmatrix} a_x \\ a_y \end{Bmatrix} = \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix} \begin{Bmatrix} a_u \\ a_v \end{Bmatrix}$$

Vê-se que $[M]^{-1} = \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix}$

Finalmente $[M] = \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix}$

"A matriz que faz a transformação (por rotação) da mudança de uma base para outra é construída por linhas, sendo que em cada linha coloca-se os cossenos diretores dos versores da nova base descritos na base antiga"

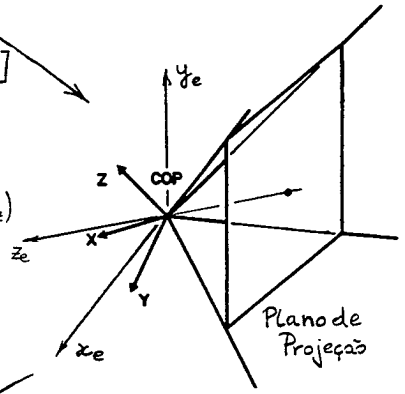
Mudança de Base XYZ → $X_eY_eZ_e$



Translação

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

[T]



Rotação da Base

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

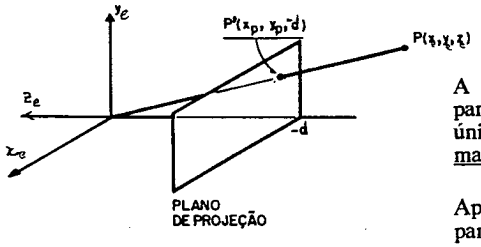
$$\hat{z}_e = \frac{-\vec{view}}{\|\vec{view}\|}$$

$$\hat{x}_e = \frac{\vec{v}_{up} \times \hat{z}_e}{\|\vec{v}_{up} \times \hat{z}_e\|}$$

$$\hat{y}_e = \hat{z}_e \times \hat{x}_e$$

[R]

$$[R] = \begin{bmatrix} xe_x & xe_y & xe_z & 0 \\ ye_x & ye_y & ye_z & 0 \\ ze_x & ze_y & ze_z & 0 \\ \hline 0 & 0 & 0 & 1 \end{bmatrix}$$

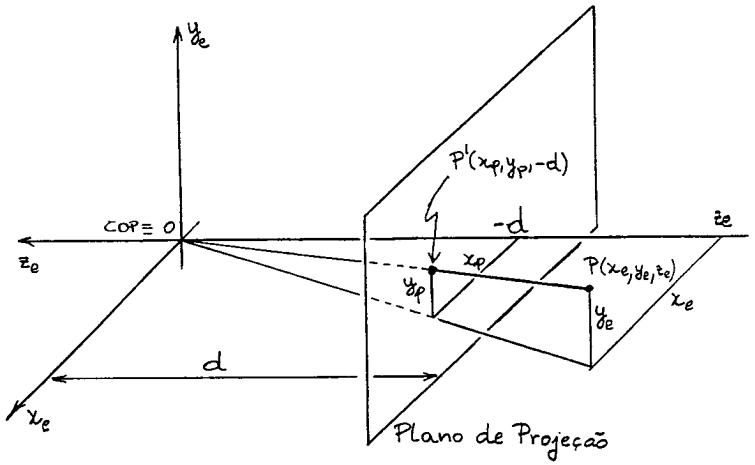


A mudança de base do espaço de modelagem para o espaço do olho pode ser grupada em uma única matriz de transformação, definida como matriz de posicionamento de câmera:

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

Após a translação [T] e a rotação [R], pronto para projetar no plano de projeção, fazendo uma projeção cônica (perspectiva) ou ortográfica.

Perspectiva Canônica



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

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

$$z_p = -d$$

[P]

$$\begin{Bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{Bmatrix} \triangleq \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ \hline 0 & 0 & -1 & 0 \end{bmatrix} \begin{Bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{Bmatrix} = \begin{Bmatrix} dx_e \\ dy_e \\ dz_e \\ (-z_e) \end{Bmatrix} \triangleq \begin{Bmatrix} (d/-z_e) \cdot x_e \\ (d/-z_e) \cdot y_e \\ -d \\ 1 \end{Bmatrix}$$

w $\div w$

Sistema de Coordenadas da Tela ($X_S Y_S Z_S$)

As coordenadas $(x_p, y_p, -d)$ são as coordenadas de um ponto (x,y,z) do objeto transformadas e projetadas no plano de projeção da tela. No entanto, não existe nenhuma informação sobre a profundidade do ponto para o interior da tela. Esta profundidade é uma informação importante pois possibilita a correta identificação dos objetos que estão mais próximos do plano de projeção e que, por isso, são os objetos visíveis. É, então, criado um espaço de coordenadas (x_s, y_s, z_s), chamado de:

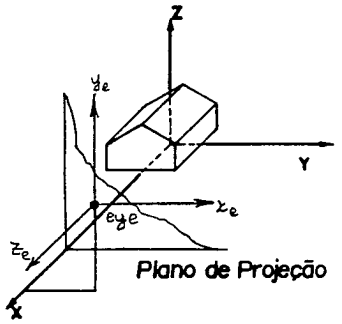
Sistema de coordenadas da tela:

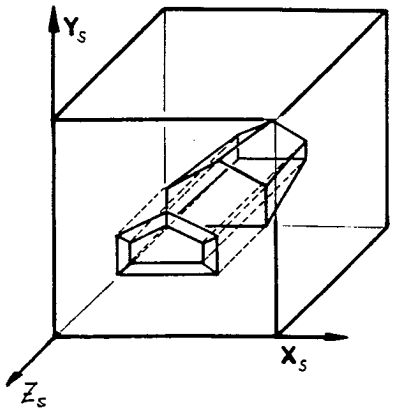
$$x_s = x_p = \frac{d}{-z_e} x_e$$

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

z_s = profundidade

Na verdade, para desenhar na tela um ponto (x_s, y_s, z_s), nós precisamos apenas usar as coordenadas x_s e y_s e ignorar a coordenada z_s. O desenho é uma projeção ortográfica do objeto transformado para o sistema de coordenadas da tela. Isto é, a transformação do objeto para o sistema de coordenadas da tela o distorce de tal maneira que o resultado da projeção ortográfica no plano $x_s y_s$ é igual à transformação de projeção de perspectiva [P] sobre o objeto.





Determinação da Profundidade Z_s

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

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

- Linhas retas no sistema de coordenadas do olho sejam transformadas para linhas retas no sistema de coordenadas da tela.
- Planos no sistema de coordenadas do olho sejam transformados para planos no sistema de coordenadas da tela.

Assim:

$$a\,x_e + b\,y_e + c\,z_e + q = 0 \qquad (1)$$

$$a'\,x_s + b'\,y_s + c'\,z_s + q' = 0 \qquad (2)$$

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

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

Substituindo x_e e y_e na equação (1) e dividindo por $-z_e$, tem-se:

$$\frac{a}{d} x_s + \frac{b}{d} y_s - c - \frac{q}{z_e} = 0 \qquad (3)$$

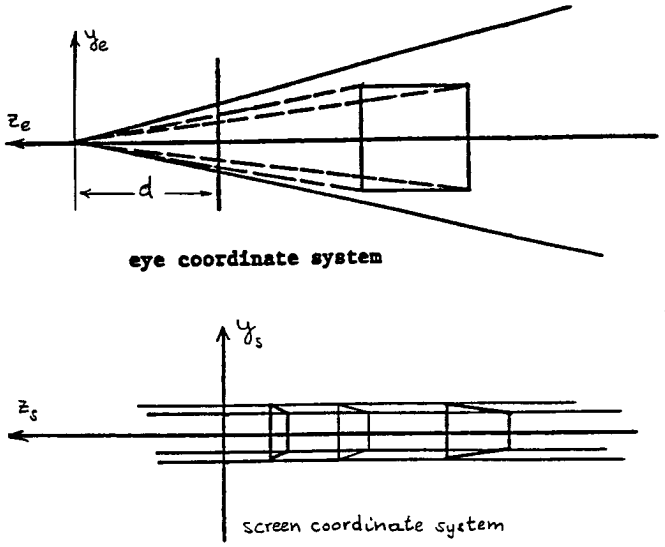
Comparando (3) com (2), vê-se que:

$$c'\,z_s + q' = -c - \frac{q}{z_e} \quad \rightarrow \quad \boxed{z_s = \alpha + \frac{\beta}{z_e}}$$

Escolha Conveniente dos Parâmetros da Profundidade Z_s

Existem infinitos valores de α e β que satifazem a imposição de planaridade $z_s = \alpha + \beta / z_e$. Isto é, dados dois valores quaisquer para α e β, tem-se que qualquer reta ou plano no sistema de coordenadas do olho vai se transformar em um plano ou reta no sistema de coordenadas da tela.

De fato, o coeficiente α pode ser considerado uma translação de corpo rígido e β um coeficiente influenciando a quantidade de distorção por perspectiva.



A figura mostra o espaço distorcido da imagem (sistema de coordenadas da tela). Todos os objetos distorcidos mostrados satisfazem as condições de planaridade e resultam em soluções idênticas para valores de projeção ortográfica x_s e y_s. A transformação de perspectiva canônica fica assim:

$$\begin{Bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{Bmatrix} \triangleq \left[\begin{array}{ccc|c} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & -\alpha & -\beta \\ \hline 0 & 0 & -1 & 0 \end{array}\right] \begin{Bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{Bmatrix} = \begin{Bmatrix} d.x_e \\ d.y_e \\ -\alpha.z_e - \beta \\ -z_e \end{Bmatrix} \triangleq \begin{Bmatrix} (d/-z_e)\,x_e \\ (d/-z_e)\,y_e \\ \alpha + (\beta/z_e) \\ 1 \end{Bmatrix}$$

[P] ÷ $(-z_e)$

Escolha Conveniente dos Parâmetros da Profundidade Z_s (cont.)

Os valores de α e β podem ser escolhidos de tal forma que todos os valores z_e que estão dentro do frustum de visão no sistema de coordenadas do olho mapeiem para valores convenientes de z_s no sistema de coordenadas da tela.

Uma possível solução é manter os valores da coordenada z_e para os planos frontal e posterior de cerceamento no espaço da tela. Isto é, associa-se uma coordenada $z_s = -near$ para pontos com $z_e = -near$ e associa-se $z_s = -far$ para pontos com $z_e = -far$. O resultado disso é:

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

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

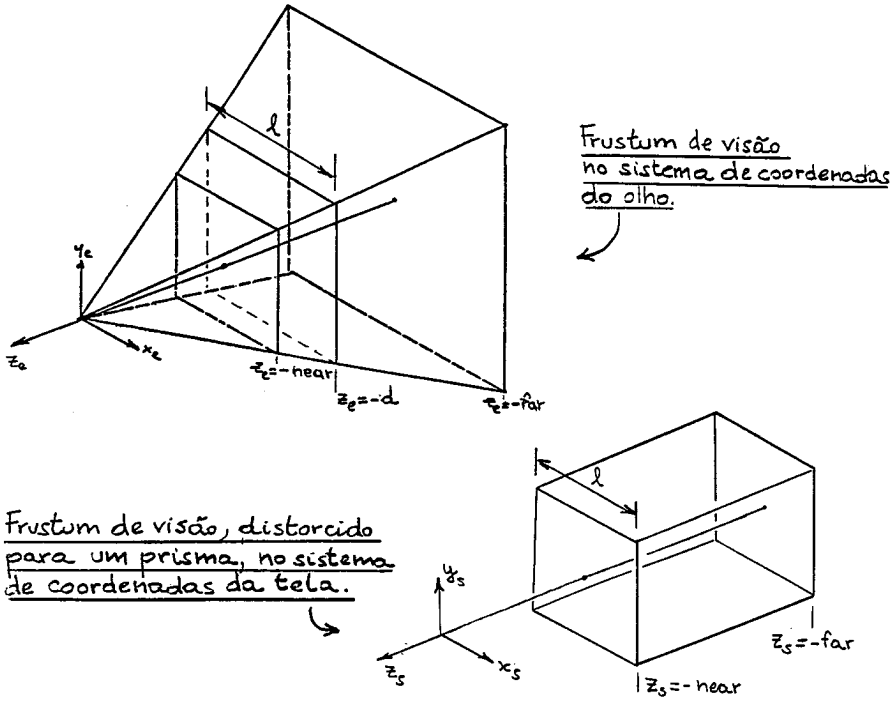
A matriz [P] com estes valores para α e β fica assim:

$$[P] = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & (far + near) & (far\ near) \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Distorção do Frustum de Visão para o Espaço da Tela

A transformação [P] distorce o frustum de visão de um tronco de pirâmide para um prisma, onde as coordenadas máximas e mínimas z_e nos planos de cerceamento anterior e posterior são preservadas.

As dimensões laterais deste prisma correspondem às dimensões da janela de visão que é definida pela interseção do plano de projeção com o frustum de visão no espaço do olho.

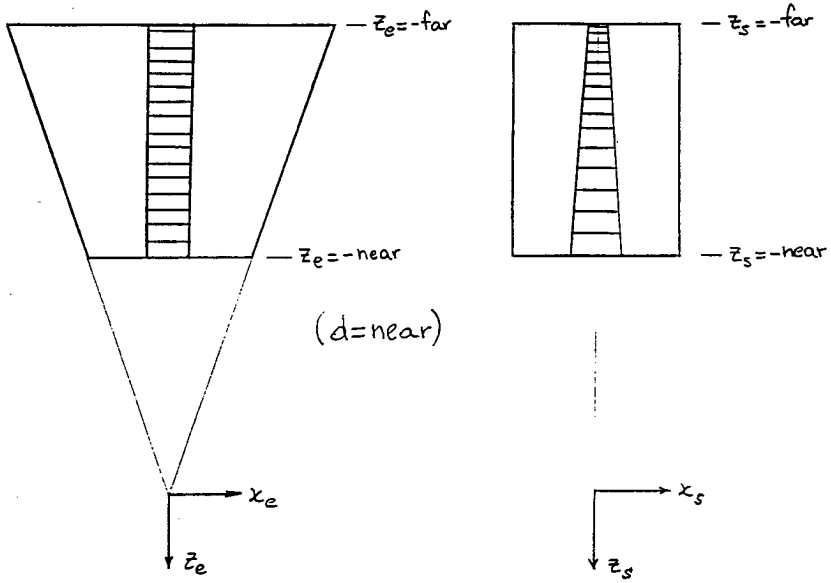


Não-linearidade na Distorção para o Espaço da Tela

A condição de planaridade $z_s = \alpha + \beta / z_e$, associada à projeção de perspectiva canônica, garante que linhas retas no sistema de coordenadas do olho sejam transformadas para linhas retas no sistema de coordenadas da tela.

Entretanto, a distorção introduzida por esta condição não preserva a métrica das linhas retas, isto é, pontos igualmente espaçados ao longo de uma reta no espaço do olho, quando transformados para o espaço da tela, não são igualmente espaçados, embora pertençam a uma mesma reta. Pode-se mostrar que eles só se mantêm igualmente espaçados quando a reta é paralela ao plano de projeção.

Esta não linearidade da distorção do espaço do olho para o espaço da tela pode ser entendida através do exemplo do trilho do trem mostrado abaixo.



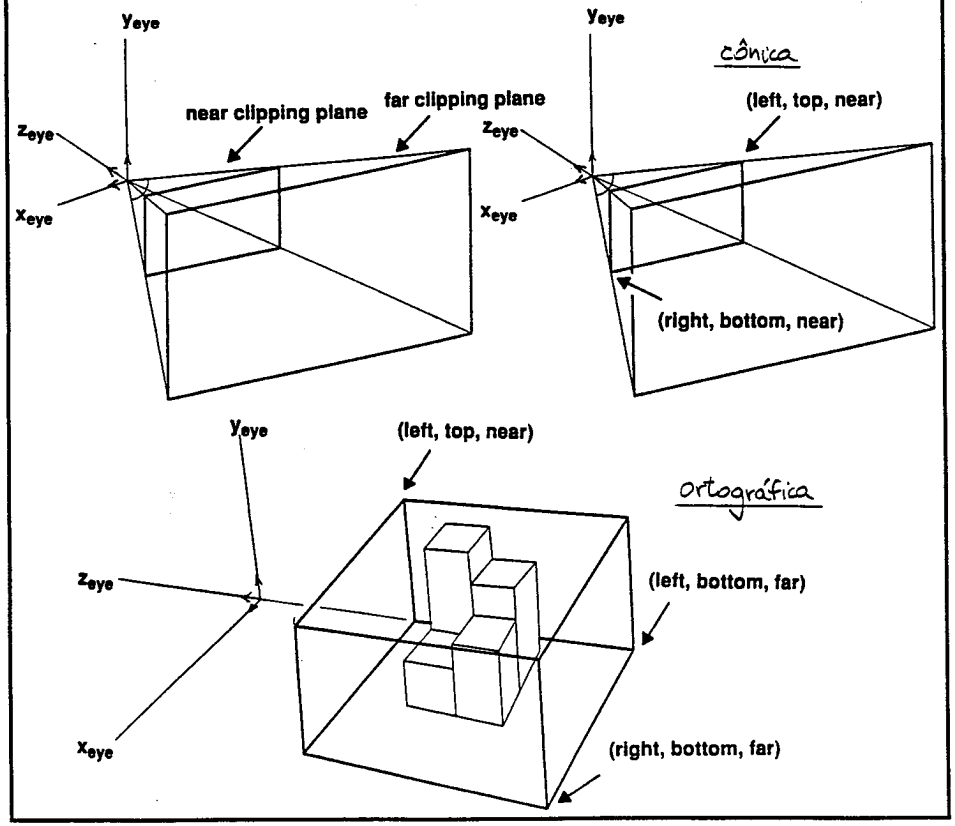
Volume de Visão

Além das transformações geométricas para visualização em 3D, é preciso definir o volume de visão, que é o volume correspondente ao frustum de visão. As porções dos objetos fora deste volume são cerceadas ("clipped"), isto é, não são desenhadas.

O volume de visão é definido, no sistema de coordenadas do olho, pelos dois planos de cerceamento anterior (*near*) e posterior (*far*), e pelos limites da janela de visão (*left*, *right*, *bottom*, *top*) definida no plano de projeção. É comum definir o plano de cerceamento anterior como sendo o plano de projeção. Isto é,

$\boxed{d = near}$

A figura mostra os parâmetros que definem o volume de visão para projeções cônicas no caso de d = *near*. Observe que os mesmos parâmetros servem para definir o volume de visão no caso de projeções ortográficas.



Normalização de Coordenadas

O cerceamento ("clipping") é feito, no caso de projeções cônicas (perspectiva), depois da transformação para o sistema de coordenadas da tela. Alguns sistemas gráficos executam o cerceamento, para projeções cônicas e ortográficas, após uma normalização das coordenadas do volume de visão para um cubo com coordenadas variando de –1 a +1.

Nesta normalização de coordenadas, em geral, a menor coordenada $z_s = -1$ corresponde ao plano $z_e = -near$ e a maior coordenada $z_s = +1$ corresponde ao plano $z_e = -far$. Nota-se que é feito um espelhamento em relação ao plano $x_s y_s$. Isto é feito para que pontos no plano anterior de cerceamento tenham a menor profundidade z_s, e pontos no plano posterior de cerceamento tenham a maior profundidade z_s.

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

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

onde $\begin{cases} S_x = \dfrac{2}{right - left} \\ S_y = \dfrac{2}{top - bottom} \\ S_z = \dfrac{2}{far - near} \end{cases}$ e $\begin{cases} \Delta_x = \dfrac{right + left}{2} \\ \Delta_y = \dfrac{top + bottom}{2} \\ \Delta_z = -\dfrac{far + near}{2} \end{cases}$

Normalização de Coordenadas (cont.)

A matriz de normalização para projeções ortográficas que resulta é:

$$[N] = \begin{bmatrix} \frac{2}{right - left} & 0 & 0 & -\frac{right + left}{right - left} \\ 0 & \frac{2}{top - bottom} & 0 & -\frac{top + bottom}{top - bottom} \\ 0 & 0 & \frac{-2}{far - near} & -\frac{far + near}{far - near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

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

- A primeira transformação [P] a ser aplicada distorce o frustum de visão de um tronco de pirâmide para um prisma, onde as coordenadas máximas e mínimas z_e nos planos de cerceamento anterior e posterior são preservadas. As outras coordenadas máximas e minimas deste prisma correspondem à janela de visão (*left, right, bottom, top*) definida no plano de projeção.
- A transformação [P] distorceu o frustum de visão para um prisma e fez o problema recair em uma normalização de projeções ortográficas. Portanto, basta aplicar a transformação [N] para normalizar as coordenadas.

A matriz [F] resultante é mostrada abaixo:

$$[F] = \begin{bmatrix} \frac{2\,d}{right - left} & 0 & \frac{right + left}{right - left} & 0 \\ 0 & \frac{2\,d}{top - bottom} & \frac{top + bottom}{top - bottom} & 0 \\ 0 & 0 & -\frac{far + near}{far - near} & \frac{-2\,far\,near}{far - near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Resumo das Transformações Geométricas para Visualização em 3D

1º Mudança de base por translação da origem do sistema XYZ do espaço de modelagem para a posição do olho (origem do sistema de coordenadas do olho) — [T].

2º Mudança de base por rotação do sistema XYZ do espaço de modelagem para o sistema $X_eY_eZ_e$ do olho — [R].

3º Se for projeção cônica, transformação do espaço $X_eY_eZ_e$ para o sistema de coordenadas da tela $X_sY_sZ_s$ (isto não é necessário para o caso de projeção ortográfica) — [P].

4º Normalização das coordenadas do volume de visão para um cubo com coordenadas variando de -1 a +1 — [N].

5º Cerceamento de pontos fora do volume de visão — < clip >. A divisão pela quarta coordenada homogênea é feita somente após o "clip" pois esta divisão projeta pontos atrás do olho no plano de projeção.

6º Remoção de pontos de linhas e superfícies ocultos por outros pontos mais próximos do plano de projeção — < hidden >.

7º Projeção ortogonal para o plano de projeção — [O]. Esta operação se resume na simples eliminação da coordenada z.

8º Transformação bidimensional afim dos pontos projetados na janela de visão ("window") para a janela do dispositivo ("viewport") — [V].

$$\begin{Bmatrix} x_v \\ y_v \end{Bmatrix} = [V]\,[O]\,<\text{hidden}>\,<\text{clip}>\,[N]\,[P]\,[R]\,[T] \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

$$\begin{Bmatrix} x_v \\ y_v \end{Bmatrix} = [V]\,[O]\,<\text{hidden}>\,<\text{clip}>\,[F]\,[C] \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$