Transformações Geométricas 3D

SCC0250 - Computação Gráfica

Prof. Fernando V. Paulovich
http://www.icmc.usp.br/~paulovic
paulovic@icmc.usp.br

Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)

15 de abril de 2010

Sumário

Sumário

Introdução

- Métodos para transformações geométricas 3D são extensões de métodos 2D, porém incluindo a coordenada z
- A translação e a escala são simples adaptações, mas a rotação é mais complexa
  - Em 2D somente são consideradas rotações em torno de um eixo perpendicular ao plano xy, em 3D pode-se pegar qualquer orientação espacial para o eixo de rotação
- Uma posição 3D expressa em coordenadas homogêneas é representada usando vetores coluna de 4 elementos, portanto as transformações 3D são matrizes 4×4

Sumário

Translação 3D

- Um objeto é movimentado adicionando-se *offsets* a cada uma das três direções Cartesianas

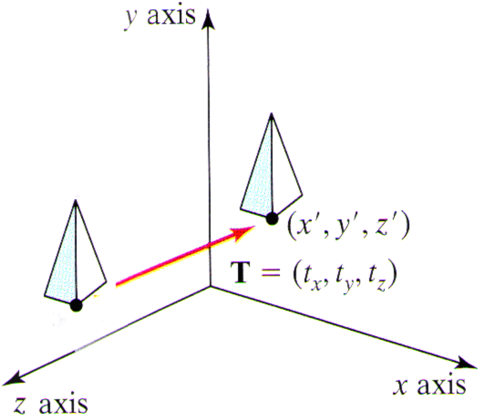
$$x' = x + t_x$$
$$y' = y + t_y$$
$$z' = z + t_z$$

- Representando matricialmente usando coordenadas homogêneas, temos

$$\mathbf{P}' = \mathbf{T} \cdot \mathbf{P}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Translação 3D



Translação 3D Inversa

- A translação inversa 3D é dada de forma semelhante a 2D, negando os *offsets* de translação

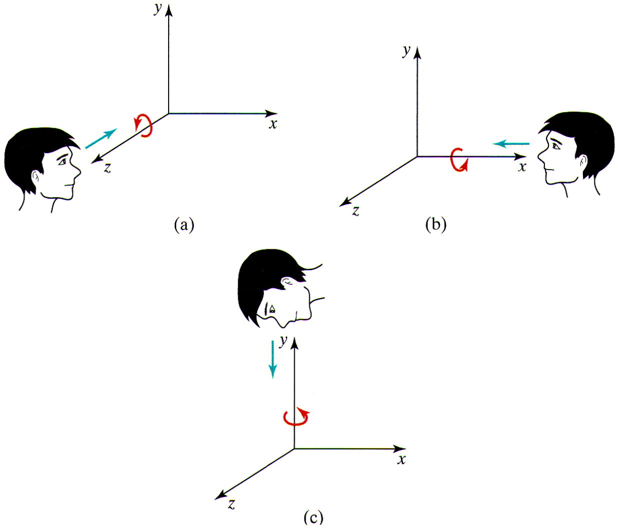
$$\mathbf{T}^{-1}(t_x, t_y, t_z) = \mathbf{T}(-t_x, -t_y, -t_z)$$

$$\mathbf{T}^{-1}(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotação 3D

- É possível rodar um objeto ao redor de qualquer eixo no espaço 3D, porém, as rotações mais fáceis são executadas ao redor dos eixos de coordenadas Cartesianas
  - É possível combinar rotações em tornos dos eixos Cartesianos para se obter rotações em torno de qualquer eixo no espaço
- Por convenção, ângulos positivos produzem rotações no sentido anti-horário

Rotação 3D



Rotação 3D nos Eixos Coordenados

- Uma rotação 2D é facilmente extendida parta uma rotação 3D ao redor do eixo z

$$x' = x\cos\theta - y\,\mathrm{sen}\,\theta$$

$$y' = x\,\mathrm{sen}\,\theta + y\cos\theta$$

$$z' = z$$

- Na forma matricial usando coordenadas homogêneas

$$\mathbf{P'} = \mathbf{R_z}(\theta)\cdot\mathbf{P}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\mathrm{sen}\,\theta & 0 & 0 \\ \mathrm{sen}\,\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotação 3D nos Eixos Coordenados

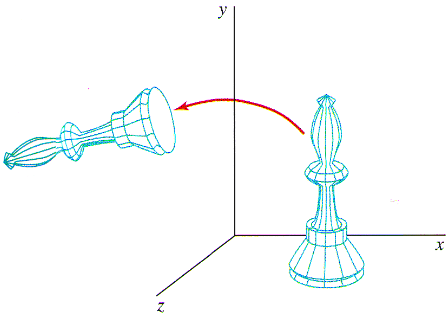
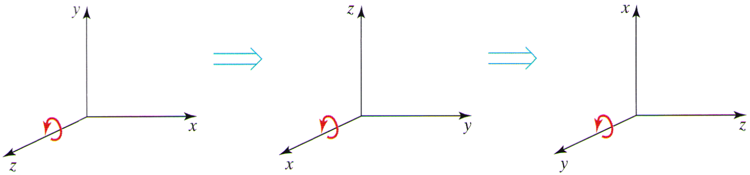


Figura: Rotação de um objeto em torno do eixo-z

Rotação 3D nos Eixos Coordenados

- As transformação de rotação para os outros eixos de coordenadas podem ser obtidas por meio de uma permutação cíclica das coordenadas x, y e z

$$x \rightarrow y \rightarrow z \rightarrow x$$



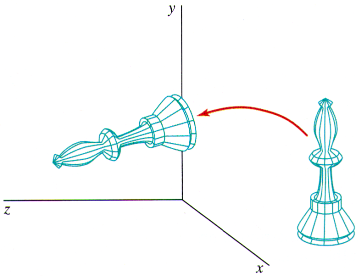
Rotação 3D nos Eixos Coordenados

- Considerando essa permutação e substituindo na equação da rotação 3D, compomos a rotação em torno do eixo-x

$$y' = y\cos\theta - z\,\mathrm{sen}\,\theta$$

$$z' = y\,\mathrm{sen}\,\theta + z\cos\theta$$

$$x' = x$$



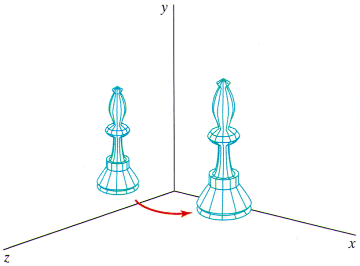
Rotação 3D nos Eixos Coordenados

- O mesmo ocorrendo para se obter as equações para rotação em torno do eixo-y

$$z' = z\cos\theta - x\,\mathrm{sen}\,\theta$$

$$x' = z\,\mathrm{sen}\,\theta + x\cos\theta$$

$$y' = y$$



Rotação 3D nos Eixos Coordenados

- Portanto as matrizes de rotação em torno dos eixos x e y são, respectivamente

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\operatorname{sen}\theta & 0 \\ 0 & \operatorname{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \operatorname{sen}\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\operatorname{sen}\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

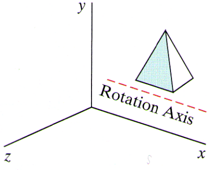
Rotação 3D Inversa

- A inversa de uma rotação é obtida trocando θ por $-\theta$
- Como somente o sinal do seno é alterado, a inversa pode ser obtida trocando as linhas pelas colunas, isto é $\mathbf{R}^{-1} = \mathbf{R}^T$

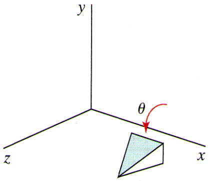
Rotação 3D Geral

- A rotação em torno de qualquer eixo pode ser obtida como a combinação de rotações e translações
- No caso especial quando o eixo de rotação é paralelo a algum eixo de coordenadas, obtemos a rotação desejada fazendo
    1. Translado o objeto de forma que o eixo de rotação coincida com o eixo paralelo de coordenadas
    2. Executo a rotação
    3. Translado o objeto de forma que o eixo de rotação é movido de volta a posição original

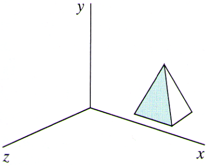
Rotação 3D Geral



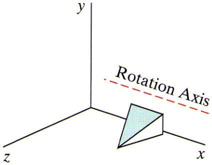
(a)
Original Position of Object



(c)
Rotate Object Through Angle θ



(b)
Translate Rotation Axis onto x Axis



(d)
Translate Rotation
Axis to Original Position

Rotação 3D Geral

- Essa sequencia de transformação sobre um ponto P é
$$\mathbf{P}' = \mathbf{T}^{-1} \cdot \mathbf{R_x}(\theta) \cdot \mathbf{T} \cdot \mathbf{P}$$

- Ou seja, a matriz composta de rotação é
$$\mathbf{R}(\theta) = \mathbf{T}^{-1} \cdot \mathbf{R_x}(\theta) \cdot \mathbf{T}$$

- Que é a mesma forma da matriz de rotação 2D quando o eixo de rotação (ortogonal ao plano xy) não coincide com a origem

Rotação 3D Geral

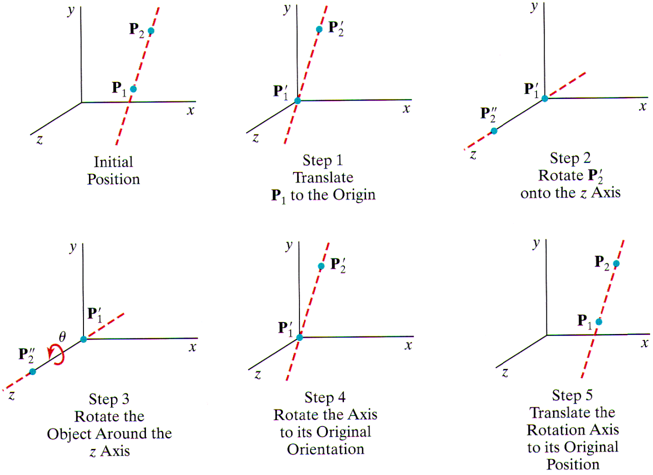
- Quando o eixo de rotação não é paralelo aos eixos de coordenadas, algumas transformações adicionais são necessárias
  - Também são necessárias rotações para alinhar o eixo de rotação com o eixo de coordenadas escolhido e para trazer de volta o eixo de rotação para a posição original

Rotação 3D Geral

- Dado o eixo de rotação e o ângulo de rotação, isso pode ser feito como
  1. Transladar o objeto de forma que o eixo de rotação passe pela origem do sistema de coordenadas
  2. Rotacionar o objeto para que o eixo de rotação coincida com um dos eixos de coordenadas
  3. Realizar a rotação sobre o eixo de coordenadas escolhido
  4. Aplicar a rotação inversa para trazer o eixo de rotação para sua orientação original
  5. Aplicar a translação inversa para trazer o eixo de rotação para sua posição espacial original

- Por conveniência, o eixo de coordenadas escolhido para o alinhamento normalmente é o eixo-z

Rotação 3D Geral



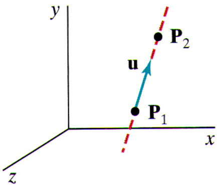
Rotação 3D Geral

- Assumindo que o eixo de rotação é definido por dois pontos ($\mathbf{P_2}$ para $\mathbf{P_1}$) e que a rotação se dá em sentido anti-horário em relação a esse eixo, podemos calcular suas componentes como

$$\mathbf{V} = \mathbf{P_2} - \mathbf{P_1} = (x_2 - x_1, y_2 - y_1, z_2 - z_1)$$

- E o vetor unitário do eixo de rotação é

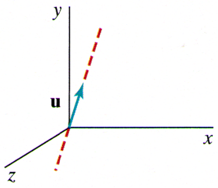
$$\mathbf{u} = \frac{\mathbf{V}}{|\mathbf{V}|} = (a, b, c)$$



Rotação 3D Geral

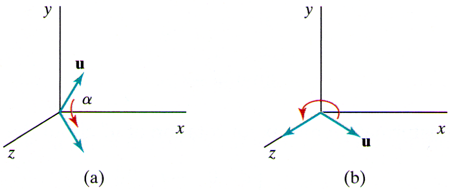
- O primeiro passo da sequencia de rotação é definir uma matriz de translação para reposicionar o eixo de rotação de forma que esse passe pela origem
  - Como a rotação se dá no sentido anti-horário, movemos o ponto $\mathbf{P_1}$ para a origem, ou seja

$$\begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



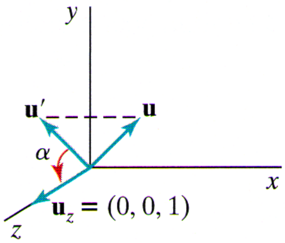
Rotação 3D Geral

- Após isso, encontramos a transformação que coloca o eixo de rotação sobre o eixo z
  - Existem várias maneiras de se realizar esse alinhamento, por exemplo, primeiro rotacionamos sobre o eixo x, depois sobre o eixo y
  - A rotação sobre o eixo x define o vetor $\mathbf{u}$ no plano xz, e a rotação no eixo y rotaciona $\mathbf{u}$ até sobrepor o eixo z



Rotação 3D Geral

- A rotação em torno do eixo x pode ser definida determinando os senos e cossenos do ângulo de rotação necessário para projetar $\mathbf{u}$ no plano xz
- Esse ângulo de rotação (α) é o ângulo entre a projeção de $\mathbf{u}$ no plano yz com o eixo z positivo



Rotação 3D Geral

- Se a projeção de $\mathbf{u}$ no plano yz for $\mathbf{u}' = (0, b, c)$, então o cosseno do ângulo de rotação α pode ser determinado a partir do produto escalar de $\mathbf{u}'$ com o vetor unitário $\mathbf{u_z}$ ao longo do eixo z

$$\cos\alpha = \frac{\mathbf{u}' \cdot \mathbf{u_z}}{|\mathbf{u}'||\mathbf{u_z}|} = \frac{c}{d}$$

- Onde d é a magnitude de $\mathbf{u}'$, isto é

$$d = \sqrt{b^2 + c^2}$$

Rotação 3D Geral

- Similarmente é possível determinar o seno de α igualando a forma independente de coordenadas do produto vetorial

$$\mathbf{u}' \times \mathbf{u_z} = \mathbf{u_x}|\mathbf{u}'||\mathbf{u_z}| \operatorname{sen} \alpha$$

- Com a sua forma Cartesiana

$$\mathbf{u}' \times \mathbf{u_z} = \mathbf{u_x} \cdot b$$

$$\mathbf{u}' \times \mathbf{u_z} = \mathbf{u_x}|\mathbf{u}'||\mathbf{u_z}| \operatorname{sen} \alpha = \mathbf{u_x} \cdot b$$

- Como $|\mathbf{u_z}| = 1$ e $|\mathbf{u}'| = d$, então

$$\operatorname{sen} \alpha = \frac{b}{d}$$

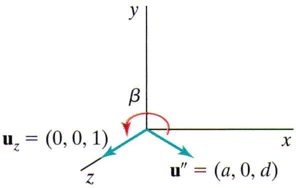
Rotação 3D Geral

- Com os senos de cossenos de α determinados, podemos definir a matriz para a rotação $\mathbf{u}$ sobre o eixo x no plano xz

$$\mathbf{R_x}(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{c}{d} & -\frac{b}{d} & 0 \\ 0 & \frac{b}{d} & \frac{c}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotação 3D Geral

- O próximo passo é determinar a matriz de rotação que vai rotacionar o vetor unitário $\mathbf{u}''$ (resultante da rotação anterior) no plano xz em torno do eixo y até sobrepor o eixo z
  - Como $\mathbf{u} = (a, b, c)$, então $\mathbf{u}'' = (a, 0, d)$ pois a rotação em torno do eixo x não altera a coordenada x, a coordenada y é zerada pela projeção no plano xz e a coordenada $z = d$ porque $|\mathbf{u}''| = |\mathbf{u}|$



Rotação 3D Geral

- Com isso podemos novamente encontrar os senos e cossenos do ângulo β fazendo

$$\cos\beta = \frac{\mathbf{u''} \cdot \mathbf{u_z}}{|\mathbf{u''}||\mathbf{u_z}|}$$

- Como $|\mathbf{u_z}| = |\mathbf{u''}| = 1$

$$\cos\beta = d$$

Rotação 3D Geral

- Igualando a forma independente de coordenadas do produto vetorial

$$\mathbf{u}'' \times \mathbf{u_z} = \mathbf{u_y}|\mathbf{u}''||\mathbf{u_z}| \operatorname{sen} \beta$$

- Com a forma Cartesiana

$$\mathbf{u}'' \times \mathbf{u_z} = \mathbf{u_y} \cdot (-a)$$

$$\mathbf{u}'' \times \mathbf{u_z} = \mathbf{u_y}|\mathbf{u}''||\mathbf{u_z}| \operatorname{sen} \beta = \mathbf{u_y} \cdot (-a)$$

- Temos

$$\operatorname{sen} \beta = -a$$

Rotação 3D Geral

- Portanto, a matriz de rotação de $\mathbf{u}''$ sobre o eixo y é

$$\mathbf{R_y}(\beta) = \begin{bmatrix} d & 0 & -a & 0 \\ 0 & 1 & 0 & 0 \\ a & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotação 3D Geral

- Com essas rotação em α e β nós alinhamos o eixo de rotação sobre o eixo z, então agora a rotação de um ângulo θ pode ser aplicada

$$\mathbf{R_z}(\theta) = \begin{bmatrix} \cos\theta & -\operatorname{sen}\theta & 0 & 0 \\ \operatorname{sen}\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotação 3D Geral

- Assim, a matriz de rotação completa sobre um eixo arbitrário fica

$$\mathbf{R}(\theta) = \mathbf{T}^{-1} \cdot \mathbf{R_x}^{-1}(\alpha) \cdot \mathbf{R_y}^{-1}(\beta) \cdot \mathbf{R_z}(\theta) \cdot \mathbf{R_y}(\beta) \cdot \mathbf{R_x}(\alpha) \cdot \mathbf{T}$$

Rotação 3D Geral

- Uma forma menos intuitiva de obter a matriz de rotação composta $\mathbf{R_y}(\beta)\mathbf{R_x}(\alpha)$ é lembrando que a matriz para qualquer sequencia de rotações 3D é da forma

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Onde a matriz 3×3 superior é ortonormal

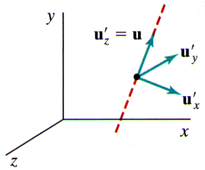
Rotação 3D Geral

- Portanto podemos definir um sistema de coordenadas locais com um eixo alinhado ao eixo de rotação, e os vetores unitários para os três eixos de coordenadas são usados para construir a matriz de rotação
- Assumindo que o eixo de rotação não é paralelo a qualquer eixo de coordenadas, esse vetores poderiam ser calculados como

$$\mathbf{u}'_{\mathbf{z}} = \mathbf{u} = (u'_{z1}, u'_{z2}, u'_{z3})$$

$$\mathbf{u}'_{\mathbf{y}} = \frac{\mathbf{u} \times \mathbf{u_x}}{|\mathbf{u} \times \mathbf{u_x}|} = (u'_{y1}, u'_{y2}, u'_{y3})$$

$$\mathbf{u}'_{\mathbf{x}} = \mathbf{u}'_{\mathbf{y}} \times \mathbf{u}'_{\mathbf{z}} = (u'_{x1}, u'_{x2}, u'_{x3})$$



Rotação 3D Geral

- Então a matriz buscada $\mathbf{R_y}(\beta)\mathbf{R_x}(\alpha)$ fica

$$\mathbf{R} = \begin{bmatrix} u'_{x1} & u'_{x2} & u'_{x3} & 0 \\ u'_{y1} & u'_{y2} & u'_{y3} & 0 \\ u'_{z1} & u'_{z2} & u'_{z3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Que transforma os vetores unitários $\mathbf{u'_x}$, $\mathbf{u'_y}$ e $\mathbf{u'_z}$ nos eixos x, y e z, alinhando o eixo de rotação com o eixo z, porque $\mathbf{u'_z} = \mathbf{u}$

Escala 3D

- A matriz de escala 3D é uma simples extensão da 2D, incluindo a variável z
- Considerando os fatores de escala $s_x > 0$, $s_y > 0$ e $s_z > 0$, temos

$$x' = x \cdot s_x$$

$$y' = y \cdot s_y$$

$$z' = z \cdot s_z$$

Escala 3D

- Que definem a transformação

$$\mathbf{P}' = \mathbf{S} \cdot \mathbf{P}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Escala 3D

- Essa definição de escala muda a posição do objeto com relação a origem das coordenadas
  - Valores > 1 afastam da origem
  - Valores < 1 aproximam da origem

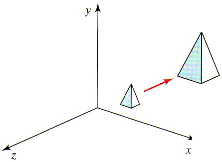


Figura: Dobrar o tamanho de um objeto também afasta o mesmo da origem

- Se $s_x = s_y = s_z$, então temos uma **escala uniforme**, caso contrário o objeto apresenta **escala diferencial**

Escala 3D

- Para se evitar esse problema podemos definir a escala com relação a uma posição fixa (x_f, y_f, z_f)
  1. Translado o ponto fixo para a origem
  2. Aplico a transformação de escala
  3. Translado o ponto fixo de volta a sua posição original

$$\mathbf{T}(x_f, y_f, z_f) \cdot \mathbf{S}(s_x, s_y, s_z) \cdot \mathbf{T}(-x_f, -y_f, -z_f)$$

$$\begin{bmatrix} s_x & 0 & 0 & (1 - s_x)x_f \\ 0 & s_y & 0 & (1 - s_y)y_f \\ 0 & 0 & s_z & (1 - s_z)z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Escala Inversa 3D

- A matriz de escala inversa 3D é obtida trocando os fatores de escala por seus opostos

$$\mathbf{T}^{-1}(s_x, s_y, s_z) = \begin{bmatrix} \frac{1}{s_x} & 0 & 0 & (1-\frac{1}{s_x})x_f \\ 0 & \frac{1}{s_y} & 0 & (1-\frac{1}{s_y})y_f \\ 0 & 0 & \frac{1}{s_z} & (1-\frac{1}{s_z})z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

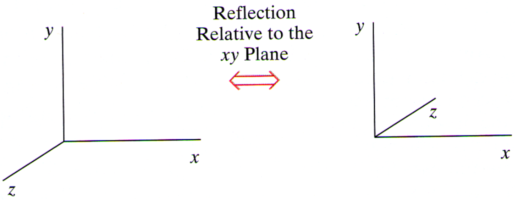
Compondo Transformações 3D

- Assim como nas transformações 2D, as transformações 3D são compostas multiplicando matrizes
- Novamente a tranformações mais a direita será a primeira a ser aplicada, e é necessário observar se a API gráfica utilizada é pós- ou pré-multiplicada

Sumário

Reflexão 3D

- É semelhante a reflexão 2D: rotação de 180^0 sobre um eixo (plano) de rotação
- Quando o plano de rotação é um plano coordenado (xy, xz ou yz), essa transformação pode ser vista como uma conversão entre um sistema orientado com a mão-esquerda e um orientado com a mão-direita (ou vice-versa)



Reflexão 3D

- Essa conversão entre um sistema orientado pela mão-direita, para um orientado pela mão-esquerda é obtido trocando o sinal da coordenada z, mantendo as coordenadas x e y (reflexão relativa ao plano xy)

$$\mathbf{M_{z_{reflect}}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- As reflexões relativas ao planos yz e xz são obtidas de forma semelhante

Cisalhamento 3D

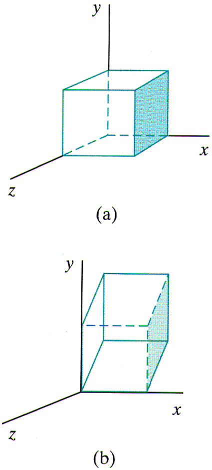
- Cisalhamento relativo aos eixos x e y é o mesmo que o já discutido em 2D, mas em 3D também é possível realizar o cisalhamento relativo ao eixo z
- O cisalhamento geral em torno do eixo-z, dado um ponto de referência z_{ref} é produzido pela seguinte matriz

$$\mathbf{M_{z_{shear}}} = \begin{bmatrix} 1 & 0 & sh_{zx} & -sh_{zx} \cdot z_{ref} \\ 0 & 1 & sh_{zy} & -sh_{zy} \cdot z_{ref} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- O efeito de sh_{zx} e sh_{zy} é alterar os valores das coordenadas x e y uma quantidade proporcional a distância de z_{ref}, enquanto mantém a coordenada z inalterada

Cisalhamento 3D

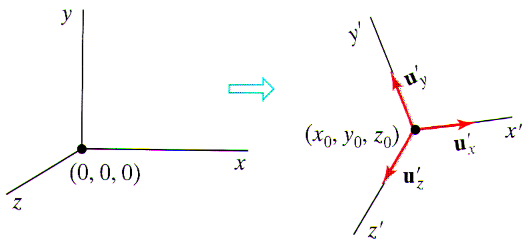
- Exemplo de matriz de cisalhamento com $sh_{zx} = sh_{zy} = 1$ e $z_{ref} = 0$ aplicada sobre um cubo unitário



Sumário

Transformações entre Sistemas de Coordenadas

- Assim como em 2D, a transformação entre sistemas de coordenadas 3D se dá sobrepondo sistemas de coordenadas diferentes
- Para transformar um sistema Cartesiano de coordenadas $x'y'z'$ em outro xyz, dado que $x'y'z'$ é definido com respeito a xyz fazemos
  1. Transladamos a origem de $x'y'z'$ para a origem de xyz
  2. Executamos uma sequencia de rotações para para alinhar os eixos $x'y'z'$ com xyz



Transformações entre Sistemas de Coordenadas

- Nesse exemplo, a origem de $x'y'z'$ é sobreposta a origem de xyz transladando de

$$\mathbf{T}(-x_0, -y_0, -z_0)$$

- E a matriz de rotação pode ser obtida usando-se os vetores unitários $\mathbf{u'}_x$, $\mathbf{u'}_y$ e $\mathbf{u'}_z$

$$\mathbf{R} = \begin{bmatrix} u'_{x1} & u'_{x2} & u'_{x3} & 0 \\ u'_{y1} & u'_{y2} & u'_{y3} & 0 \\ u'_{z1} & u'_{z2} & u'_{z3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Então a transformação completa é dada por $\mathbf{R} \cdot \mathbf{T}$

Sumário

Transformações Afim

- Uma transformação afim é dada pela forma

$$\begin{aligned} x' &= a_{xx}x + a_{xy}y + a_{xz}z + b_x \\ y' &= a_{yx}x + a_{yy}y + a_{yz}z + b_y \\ z' &= a_{zx}x + a_{zy}y + a_{zz}z + b_z \end{aligned}$$

- x', y' e z' são transformações lineares das coordenadas originais x, y e z

- Uma propriedade geral é que linhas paralelas são transformadas em linhas paralelas e pontos finitos são transformados em pontos finitos
- Translação, rotação, escala, reflexão e cisalhamento, ou suas combinações, são transformações afim

Sumário

Programação OpenGL

```
#include <stdlib.h>
#include <GL/glut.h>
#include <math.h>

float alpha = 0;
float beta = 0;
float delta = 1;

void init(void)
{
    glClearColor(0, 0, 0, 0); //define a cor de fundo
    glEnable(GL_DEPTH_TEST); //remoção de superfície oculta

    glMatrixMode(GL_PROJECTION); //define que a matrix é a de projeção
    glLoadIdentity(); //carrega a matrix de identidade
    glOrtho(-5, 5, -5, 5, -5, 5); //define uma projeção ortográfica
}
```

Programação OpenGL

```
void display(void)
{
    //limpa o buffer
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    //define que a matrix é a de modelo
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity(); //carrega matrix identidade

    //rotaciona e escala uma esfera 'aramado'
    glRotatef(beta, 0, 1, 0);
    glRotatef(alpha, 1, 0, 0);
    glScalef(delta, delta, delta);
    glColor3f(1,1,0);
    glutWireSphere(1.0f,20,20);

    //desenha um 'piso' sob a esfera
    glTranslatef(0,-1,0);
    glScalef(4, 0.1, 4);
    glColor3f(0,0,1);
    glutSolidCube(1.0f);

    //força o desenho das primitivas
    glutSwapBuffers();
}
```

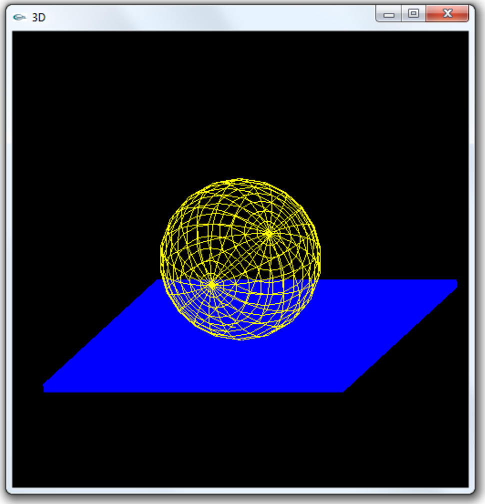
Programação OpenGL

```
void special_keyboard(int key, int x, int y)
{
    switch (key)
    {
    case GLUT_KEY_PAGE_UP: //faz zoom-in
        delta = delta * 1.1;
        break;
    case GLUT_KEY_PAGE_DOWN: //faz zoom-out
        delta = delta * 0.809;
        break;
    case GLUT_KEY_UP: //gira sobre o eixo-x
        alpha = alpha - 1;
        break;
    case GLUT_KEY_DOWN: //gira sobre o eixo-x
        alpha = alpha + 1;
        break;
    case GLUT_KEY_LEFT: //gira sobre o eixo-y
        beta = beta + 1;
        break;
    case GLUT_KEY_RIGHT: //gira sobre o eixo-y
        beta = beta - 1;
        break;
    }

    //forçar o redesenho da tela usando double-buffering
    glutPostRedisplay();
}
```

Programação OpenGL

```
void keyboard(unsigned char key, int x, int y)
{
    //'q' ou 'Q' ou ESC para sair do sistema
    if ('q' == key || 'Q' == key || 27 == key)
    {
        exit(0);
    }
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutCreateWindow("3D");

    init();

    //funções de callback
    glutDisplayFunc(&display); //registra função de desenho
    glutKeyboardFunc(&keyboard); //registra teclado
    glutSpecialFunc(&special_keyboard); //registra teclado especial

    glutMainLoop();

    return 0;
}
```

Programação OpenGL



Programação OpenGL

- Armazenando e restaurando transformações

```
void display(void)
{
    //limpa o buffer
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    //define que a matriz é a de modelo
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    glScalef(delta, delta, delta); //faça a escala de todos objetos

    glPushMatrix(); //armazena a matriz corrente
      glTranslatef(-3,0,0);
      glRotatef(beta,0,1,0);
      glRotatef(alpha,1,0,0);
      glColor3f(1,1,0);
      glutWireSphere(1,20,20);
    glPopMatrix(); //restaura a matriz anterior

    glPushMatrix(); //armazena a matriz corrente
      glTranslatef(3,0,0);
      glRotatef(beta,0,1,0);
      glRotatef(alpha,1,0,0);
      glColor3f(1,0,0);
      glutWireSphere(1,20,20);
    glPopMatrix(); //restaura a matriz anterior

    //força o desenho das primitivas
    glFlush();
    glutSwapBuffers();
}
```

Programação OpenGL