Transformações Geométricas 3D

SCC0250 - Computação Gráfica

Profa. Maria Cristina F. Oliveira

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

29 de agosto de 2023

Sumário

Sumário

Introdução

- As transformações geométricas 3D são extensões das formulações 2D, incluindo a coordenada z

- A translação e a escala são adaptações diretas, mas a rotação fica mais complexa
  - Em 2D somente são consideradas rotações em torno de um eixo perpendicular ao plano xy, em 3D o eixo de rotação é arbitrário

- Uma posição 3D expressa em coordenadas homogêneas é representada por meio de vetores coluna com 4 elementos, portanto as transformações 3D são matrizes 4×4

Sumário

Sumário

Translação 3D

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

$$x' = x + t_x$$

$$y' = y + t_y$$

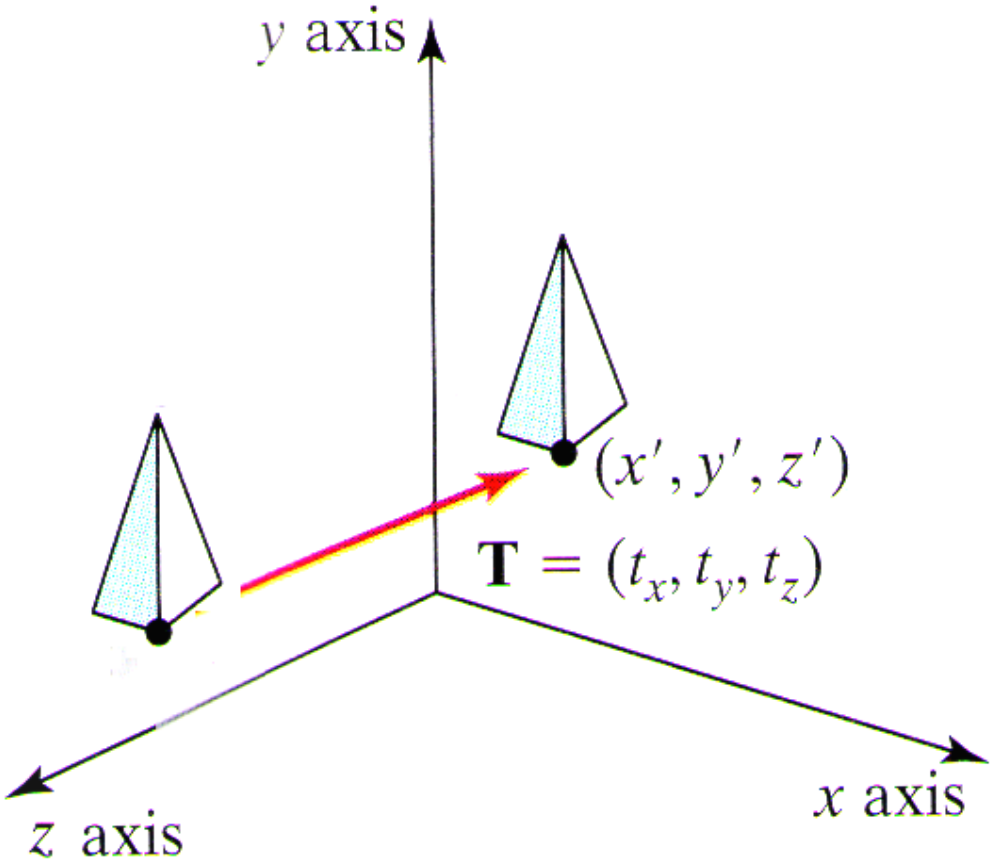
$$z' = z + t_z$$

- 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 descrita de forma semelhante a 2D, negando os *offsets* de translação

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

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

Sumário

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 reposiciona o objeto em relação a origem das coordenadas
  - Valores > 1 afastam da origem
  - Valores < 1 aproximam da origem

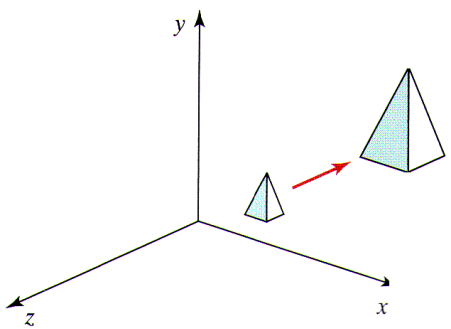


Figura: Dobrar o tamanho de um objeto também o afasta da origem

- Se $s_x = s_y = s_z$, então temos uma **escala uniforme**, caso contrário uma **escala não uniforme**

Escala 3D

- Para evitar o reposicionamento do objeto pode-se definir a escala em 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 padrão
  3. Translado o ponto fixo de volta à 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

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

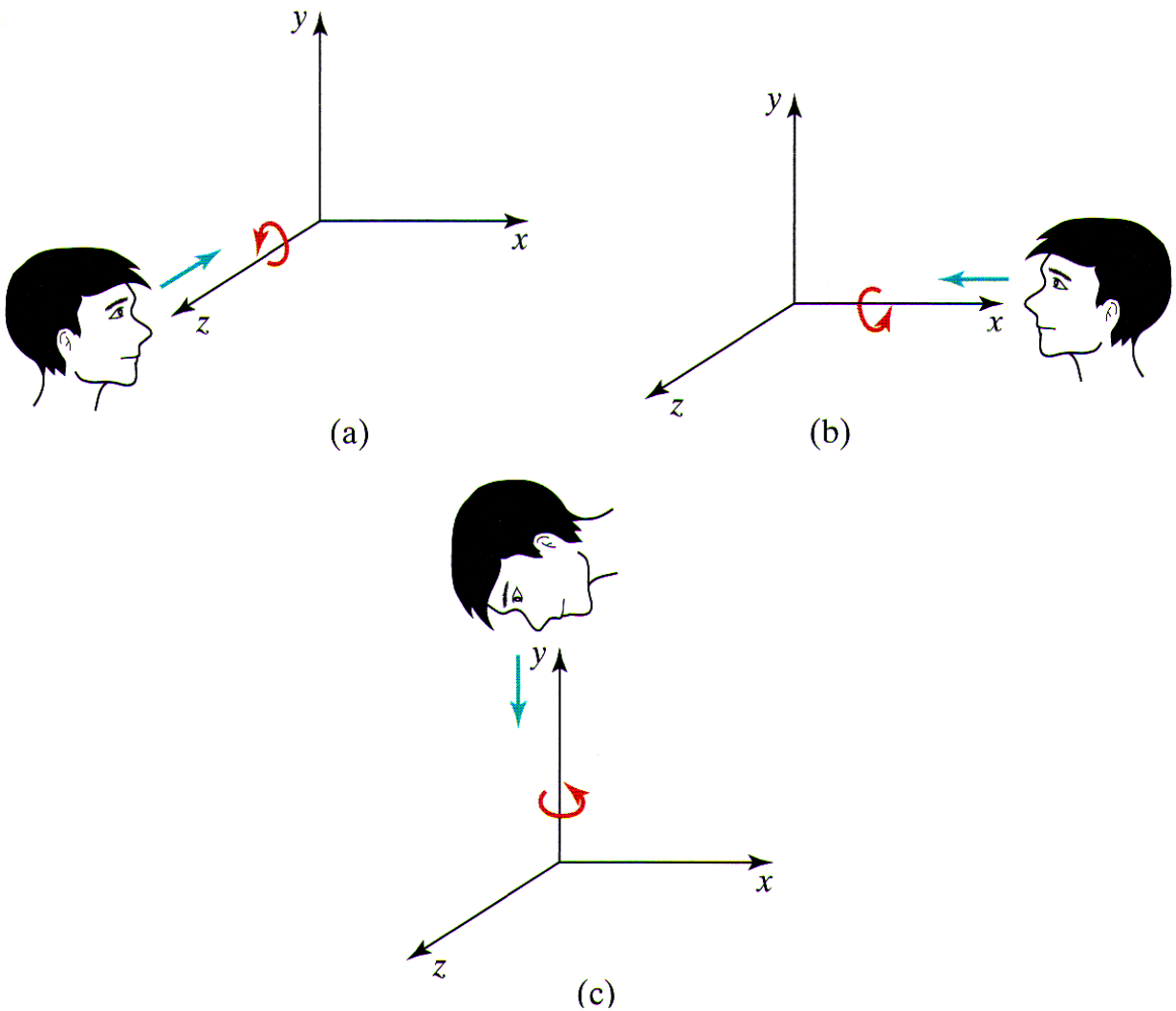
Sumário

Rotação 3D

- É possível rotacionar um objeto ao redor de qualquer eixo no espaço 3D. As rotações mais triviais são executadas em relação aos eixos principais do sistema de coordenadas Cartesianas
  - É possível combinar rotações em tornos dos eixos Cartesianos para definir rotações em torno de um eixo qualquer no espaço

- Por convenção, ângulos positivos descrevem rotações no sentido anti-horário

Rotação 3D



Rotação 3D nos Eixos Coordenados

- Uma rotação 3D ao redor do eixo z nada mais é do que uma rotação 2D estendida para o espaço tridimensional

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

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

$$z' = z$$

- 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

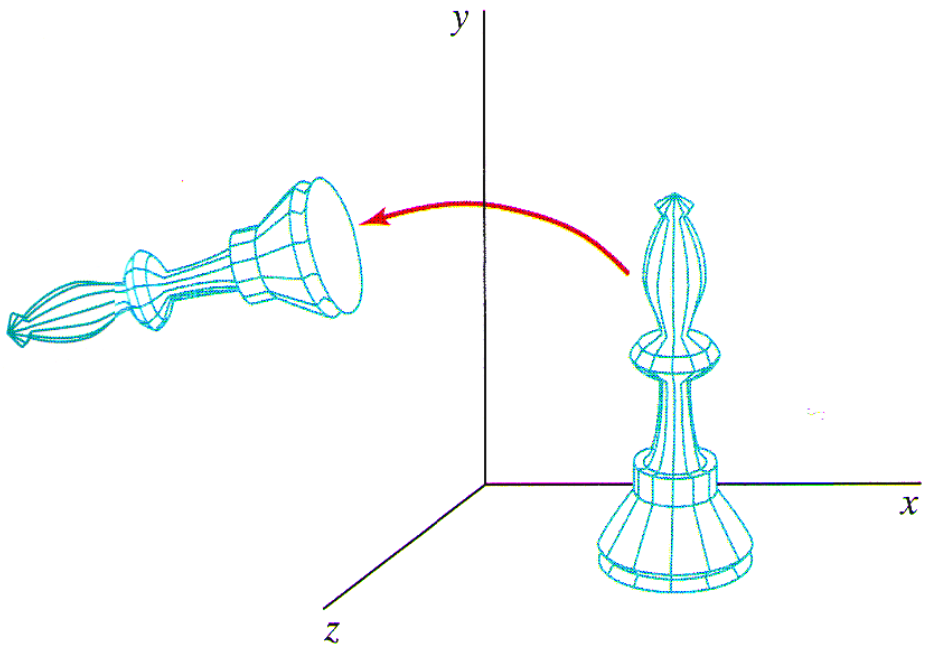
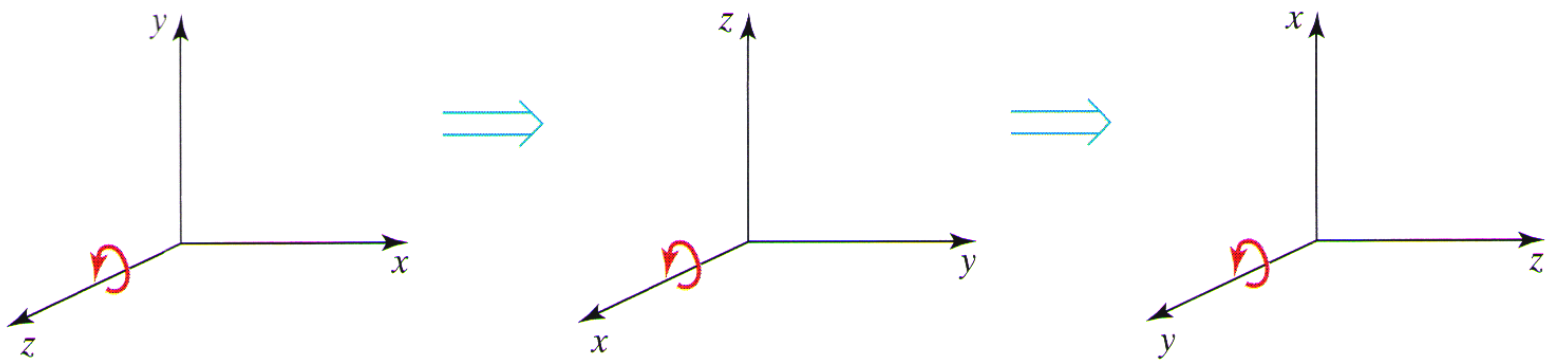


Figura: Rotação de um objeto em torno do eixo-z

Rotação 3D nos Eixos Coordenados

- As transformações de rotação em relação aos outros eixos de coordenadas podem ser obtidas por meio de uma permutação cíclica das coordenadas x, y e z

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



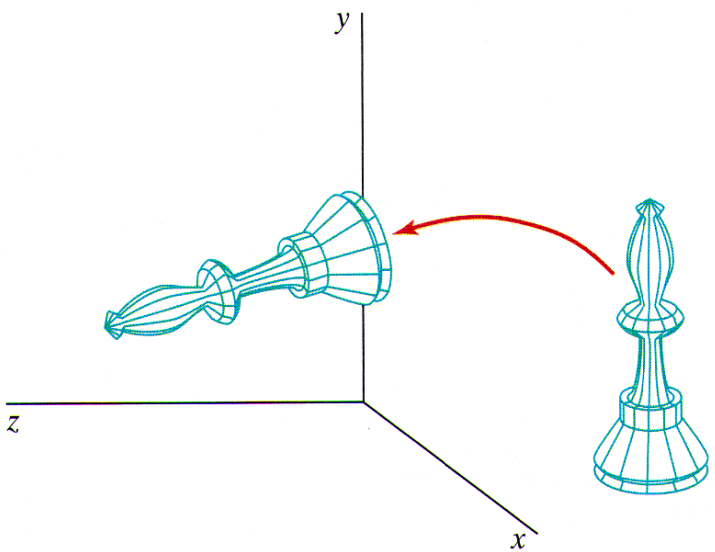
Rotação 3D nos Eixos Coordenados

- Considerando essa permutação, e substituindo na equação da rotação 3D, tem-se a matriz de rotação em torno do eixo-x

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

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

$$x' = x$$



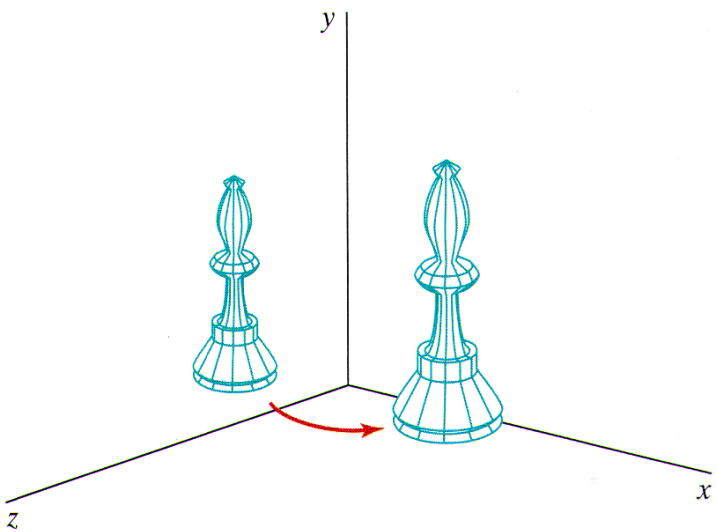
Rotação 3D nos Eixos Coordenados

- E equivalentemente para obter as equações de 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

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

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

Rotação 3D Inversa

- A inversa de uma rotação é obtida trocando θ por $-\theta$

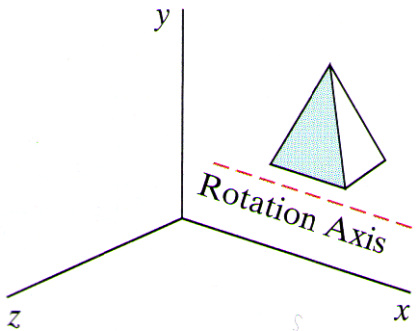
- Como somente o sinal do seno é alterado, pode-se obter a inversa simlesmente transpondo a matriz original, isto é $\mathbf{R}^{-1} = \mathbf{R}^{T}$

Rotação 3D Geral

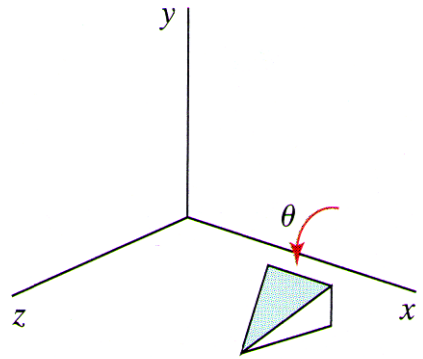
- A rotação em torno de um eixo arbitrário pode ser obtida como uma combinação de rotações e translações

- No caso especial em que o eixo de rotação é paralelo a um dos eixos de coordenadas principais, é simples:
  1. Translado o objeto de modo a alinhar o eixo de rotação com o eixo de coordenadas paralelo
  2. Executo a rotação
  3. Translado o objeto de modo a mover de volta o eixo de rotação para a posição original

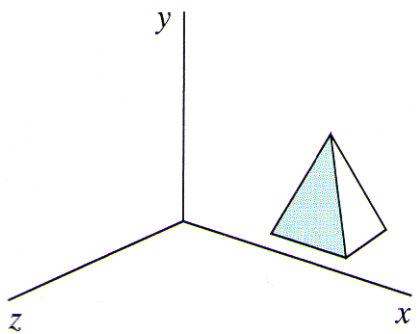
Rotação 3D Geral



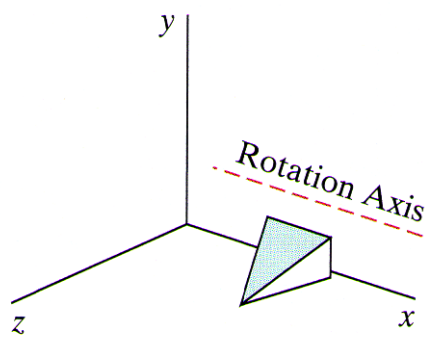
(a)
Original Position of Object



(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 transformações sobre um ponto P, para um eixo que passa por $P_1 = (x_1, y_1, z_1)$ é dada por

$$\mathbf{P'} = \mathbf{T}(x_1, y_1, z_1) \cdot \mathbf{R_x}(\theta) \cdot \mathbf{T^{-1}}(\mathbf{x_1}, \mathbf{y_1}, \mathbf{z_1}) \cdot \mathbf{P}$$

- Ou seja, a matriz composta de rotação é

$$\mathbf{R}(\theta) = \mathbf{T}(\mathbf{x_1}, \mathbf{y_1}, \mathbf{z_1}) \cdot \mathbf{R_x}(\theta) \cdot \mathbf{T}^{-1}(x_1, y_1, z_1)$$

- Note que essa matriz é similar à matriz de rotação 2D (eixo ortogonal ao plano xy) definida em relação a um pivô de rotação diferente da origem

Rotação 3D Geral

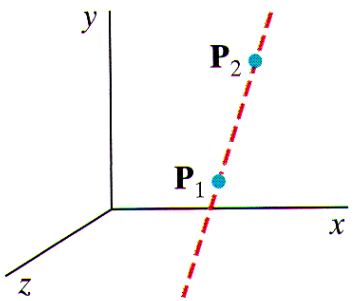
- Quando o eixo de rotação não é paralelo aos eixos principais, é necessário aplicar uma sequencia de transformações que alinhem o eixo arbitrário de rotação com um dos eixos principais

Rotação 3D Geral

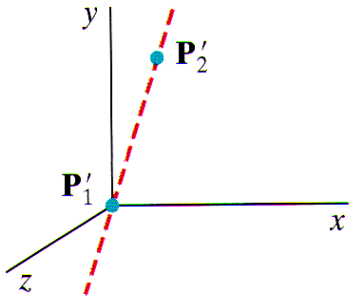
- Dados o eixo de rotação e o ângulo de rotação, isso pode ser feito da seguinte forma:
  1. Translado o objeto de forma que o eixo de rotação passe pela origem do sistema de coordenadas
  2. Rotaciono o objeto para alinhar o eixo de rotação com um dos eixos principais
  3. Realizo a rotação em relação ao eixo principal escolhido
  4. Aplico a rotação inversa para retornar o eixo de rotação à sua orientação original
  5. Aplico a translação inversa para trazer o eixo de rotação de volta à sua posição espacial original

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

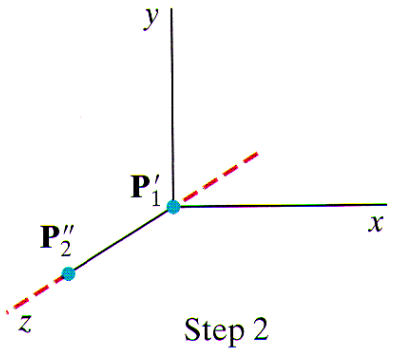
Rotação 3D Geral



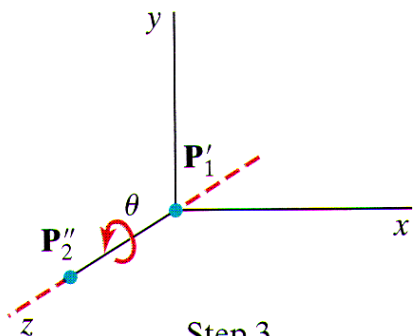
Initial Position



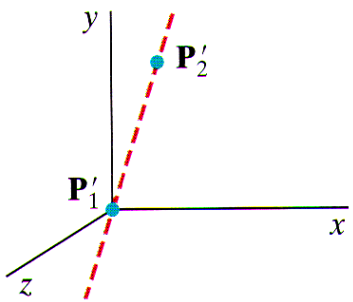
Step 1
Translate
$\mathbf{P}_1$ to the Origin



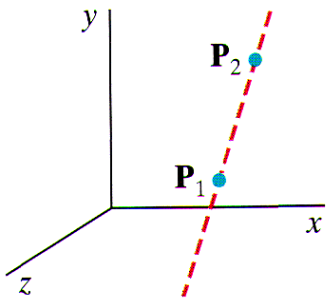
Step 2
Rotate $\mathbf{P}_2'$
onto the z Axis



Step 3
Rotate the
Object Around the
z Axis



Step 4
Rotate the Axis
to its Original
Orientation



Step 5
Translate the
Rotation Axis
to its Original
Position

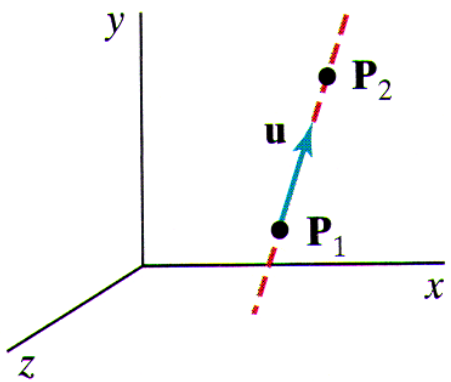
Rotação 3D Geral

- 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

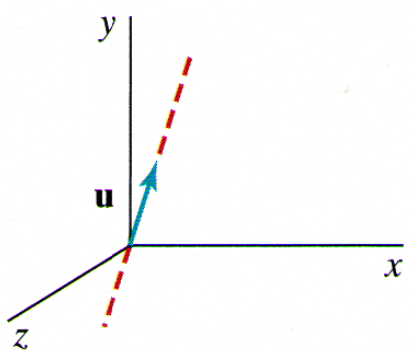
$$\mathbf{u} = \frac{\mathbf{V}}{|\mathbf{V}|} = (a, b, c)$$



Rotação 3D Geral

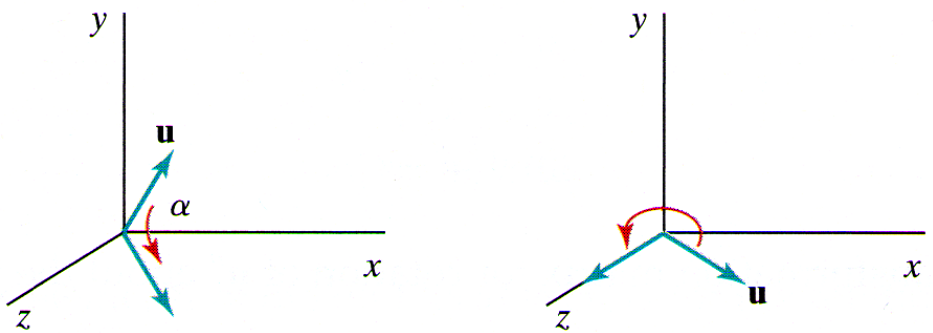
- Na sequencia de passos que define a rotação 3D geral, o primeiro passo consiste em definir a matriz de translação que reposiciona o eixo de rotação dado pelo vetor $\mathbf{u}$ de modo que ele passe pela origem
  - Como a rotação se dá no sentido anti-horário, movemos o ponto $\mathbf{P_1}$ de coordenadas $\mathbf{P_1} = (x_1, y_1, z_1)$ para a origem, ou seja

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



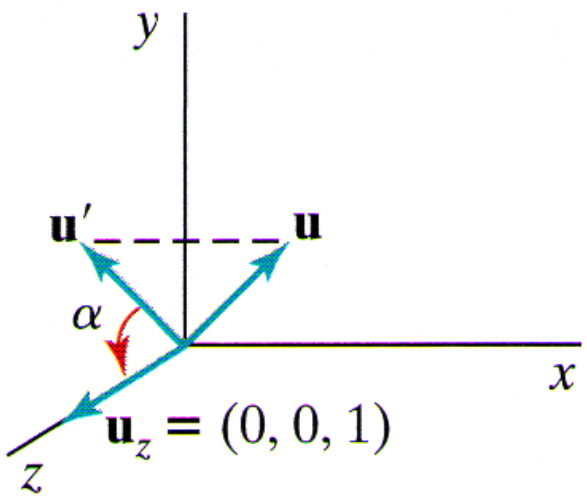
Rotação 3D Geral

- Em seguida, encontramos a transformação que alinha o eixo de rotação com o eixo z
  - Existem várias maneiras de fazer esse alinhamento, por exemplo, primeiro rotacionamos o vetor (eixo de rotação) $\mathbf{u}$ em torno do eixo x, depois em torno do eixo y
  - A rotação em torno do eixo x reposiciona o vetor no plano xz, e a rotação em torno do eixo y o reposiciona alinhado ao eixo z



Rotação 3D Geral

- Os parâmetros da rotação em torno do eixo x são 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 o vetor $\mathbf{u}$ projetado no plano yz (vetor $\mathbf{u}'$) e o eixo z positivo



Rotação 3D Geral

- Seja $\mathbf{u}' = (0, b, c)$ a projeção de $\mathbf{u}$ no plano yz. 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

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

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

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

Rotação 3D Geral

- É possível determinar de maneira similar 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

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

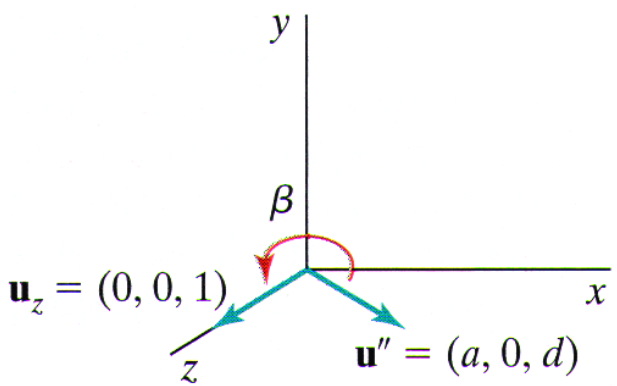
Rotação 3D Geral

- Determinados o seno e o cosseno de α, podemos definir a matriz de rotação de $\mathbf{u}$ em relação ao eixo x, para reposicioná-lo 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 que rotaciona o vetor unitário $\mathbf{u}'' = (u''_x, u''_y, u''_z)$ (resultante da rotação anterior) em torno do eixo y, alinhando-o com o eixo z (no plano xz)
  - Como $\mathbf{u} = (a, b, c)$, então $\mathbf{u}'' = (a, 0, d)$, pois a rotação em torno do eixo x não altera a componente u''_x; a componente u''_y é zerada como resultado da rotação, que reposiciona $\mathbf{u}$ no plano xz; e a componente $u''_z = d$ porque $|\mathbf{u}''| = |\mathbf{u}|$



Rotação 3D Geral

- Novamente, podemos determinar o seno e o cosseno 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

- Aplicada essa sequencia de duas rotações de ângulos α e β, o eixo de rotação foi alinhado com o eixo z, agora podemos aplicar a rotação desejada, de um ângulo θ

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

Rotação 3D Geral

- E assim, a matriz completa de rotação em torno de um eixo arbitrário P_1P_2 é dada por

$$\mathbf{R}(\theta) = \mathbf{T}(x_1, y_1, z_1)\cdot\mathbf{R_x}^{-1}(\alpha)\cdot\mathbf{R_y}^{-1}(\beta)\cdot\mathbf{R_z}(\theta)\cdot\mathbf{R_y}(\beta)\cdot\mathbf{R_x}(\alpha)\cdot\mathbf{T}^{-1}(x_1, y_1, z_1)$$

Rotação 3D Geral

- Há uma maneira menos intuitiva, porém mais simples, de obter a matriz de rotação composta $\mathbf{R_y}(\beta)\mathbf{R_x}(\alpha)$ lembrando que qualquer sequencia de rotações 3D é descrita por uma matriz na forma

$$\mathbf{R} = \begin{bmatrix} r_{1x} & r_{1y} & r_{1z} & 0 \\ r_{2x} & r_{2y} & r_{2z} & 0 \\ r_{3x} & r_{3y} & r_{3z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Em que a sub-matriz superior 3×3 é ortonormal
- Ou seja, os vetores unitários $\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}$ definem uma base ortonormal (e um sistema de coordenadas local)

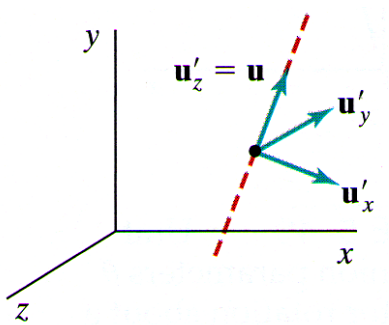
Rotação 3D Geral

- Portanto, dado o eixo de rotação como um vetor unitário, e dois outros eixos (vetores unitários) perperdiculares entre si e ao eixo de rotação, esses três vetores podem ser usados diretamente para construir uma matriz de rotação
- Assumindo que o eixo de rotação não é paralelo a qualquer dos eixos principais, esses vetores poderiam ser calculados como

$$\mathbf{u'_z} = \mathbf{u} = (u'_{zx}, u'_{zy}, u'_{zz})$$

$$\mathbf{u'_y} = \frac{\mathbf{u} \times \mathbf{u_x}}{|\mathbf{u} \times \mathbf{u_x}|} = (u'_{yx}, u'_{yy}, u'_{yz})$$

$$\mathbf{u'_x} = \mathbf{u'_y} \times \mathbf{u'_z} = (u'_{xx}, u'_{xy}, u'_{xz})$$



Rotação 3D Geral

- Então a matriz de rotação desejada $\mathbf{R_y}(\beta)\mathbf{R_x}(\alpha)$ é

$$\mathbf{R} = \begin{bmatrix} u'_{xx} & u'_{xy} & u'_{xz} & 0 \\ u'_{yx} & u'_{yy} & u'_{yz} & 0 \\ u'_{zx} & u'_{zy} & u'_{zz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Essa matriz 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 $\mathbf{u'_z} = \mathbf{u}$

Sumário

Compondo Transformações 3D

- Assim como no caso das transformações 2D, transformações 3D podem ser compostas multiplicando matrizes

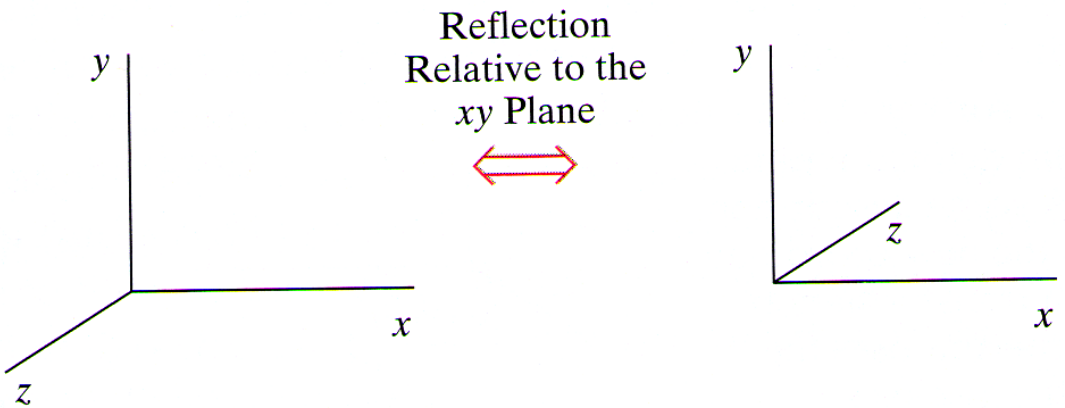
- Novamente, a tranformação mais a direita será a primeira a ser aplicada, e é necessário observar se a API gráfica utilizada adota pós- ou pré-multiplicação (é apenas uma questão de notação)

Sumário

Sumário

Reflexão 3D

- É semelhante à reflexão 2D: rotação de 180^0 ao redor de um eixo (ou plano) de rotação
- Quando o plano de rotação coincide com um dos planos principais (xy, xz ou yz), essa transformação corresponde a uma conversão entre um sistema de coordenadas orientado pela mão-esquerda e um sistema orientado pela mão-direita (ou vice-versa)



Reflexão 3D

- Essa conversão de um sistema de coordenadas da mão-direita para um sistema de coordenadas da mão-esquerda é feita invertendo o sinal das coordenadas z e preservando as coordenadas x e y (reflexão relativa ao plano xy)

$$\mathbf{M}_{\mathbf{z}_{\mathbf{reflect}}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Reflexões relativas ao planos yz e xz são obtidas de forma análoga

Sumário

Cisalhamento 3D

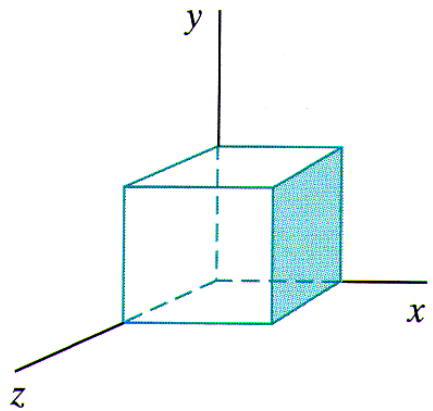
- Cisalhamento relativo aos eixos x e y é similar ao caso 2D, já discutido, mas em 3D também é possível realizar o cisalhamento relativo ao eixo z
- O cisalhamento geral em torno do eixo-z, dado um ponto de referência z_{ref}, é descrito pela seguinte matriz

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

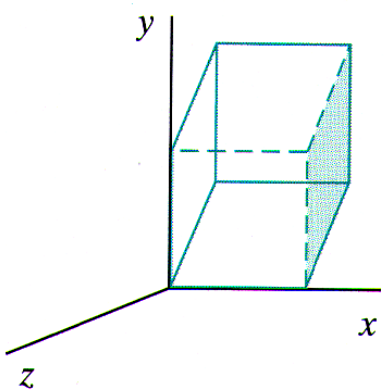
- O efeito dos parâmetro sh_{zx} e sh_{zy} é alterar os valores das coordenadas x e y por uma quantidade proporcional à distância de z_{ref}, mantnedo a coordenada z inalterada

Cisalhamento 3D

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



(a)



(b)

Sumário

Transformações Afim

- Uma transformação afim tem a 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 das transformações afim é que elas preservam o paralelismo de linhas (linhas paralelas continuam paralelas após a transformação) e transformam pontos finitos em pontos finitos
- Translação, rotação, escala, reflexão e cisalhamento, ou suas combinações, são transformações afins

Sumário

Programação OpenGL

```
1  public class Renderer extends KeyAdapter implements GLEventListener{
2
3      public Renderer() {
4          this.alpha = 0;
5          this.beta = 0;
6          this.delta = 1;
7      }
8
9      public void init(GLAutoDrawable drawable) {
10         GL gl = drawable.getGL();
11         gl.glClearColor(0, 0, 0, 0); //define a cor de fundo
12         gl.glEnable(GL.GL_DEPTH_TEST); //remoo de superfcie oculta
13
14         gl.glMatrixMode(GL.GL_PROJECTION); //define que a matrix  a de projeo
15         gl.glLoadIdentity(); //carrega a matrix de identidade
16         gl.glOrtho(-5, 5, -5, 5, -5, 5); //define uma projeo ortogrfica
17     }
18
19     public void reshape(GLAutoDrawable drawable, int x, int y, int width, int ↩
           height) {
20     }
21
22     public void displayChanged(GLAutoDrawable drawable, boolean modeChanged, ↩
           boolean deviceChanged) {
23     }
24
25     ...
26
27     private float alpha;
28     private float beta;
29     private float delta;
30 }
```

Programação OpenGL

```
1  public class Renderer extends KeyAdapter implements GLEventListener {
2      ...
3
4      public void display(GLAutoDrawable drawable) {
5          GL gl = drawable.getGL();
6          GLUT glut = new GLUT();
7
8          //limpa o buffer
9          gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
10
11         //define que a matrix  a de modelo
12         gl.glMatrixMode(GL.GL_MODELVIEW);
13         gl.glLoadIdentity(); //carrega matrix identidade
14
15         //rotaciona e escala uma esfera 'aramado'
16         gl.glRotatef(beta, 0, 1, 0);
17         gl.glRotatef(alpha, 1, 0, 0);
18         gl.glScalef(delta, delta, delta);
19         gl.glColor3f(1, 1, 0);
20         glut.glutWireSphere(1.0f, 20, 20);
21
22         //desenha um 'piso' sob a esfera
23         gl.glTranslatef(0, -1, 0);
24         gl.glScalef(4, 0.1f, 4);
25         gl.glColor3f(0, 0, 1);
26         glut.glutSolidCube(1.0f);
27
28         //fora o desenho das primitivas
29         gl.glFlush();
30     }
31
32     private float alpha;
33     private float beta;
34     private float delta;
35 }
```

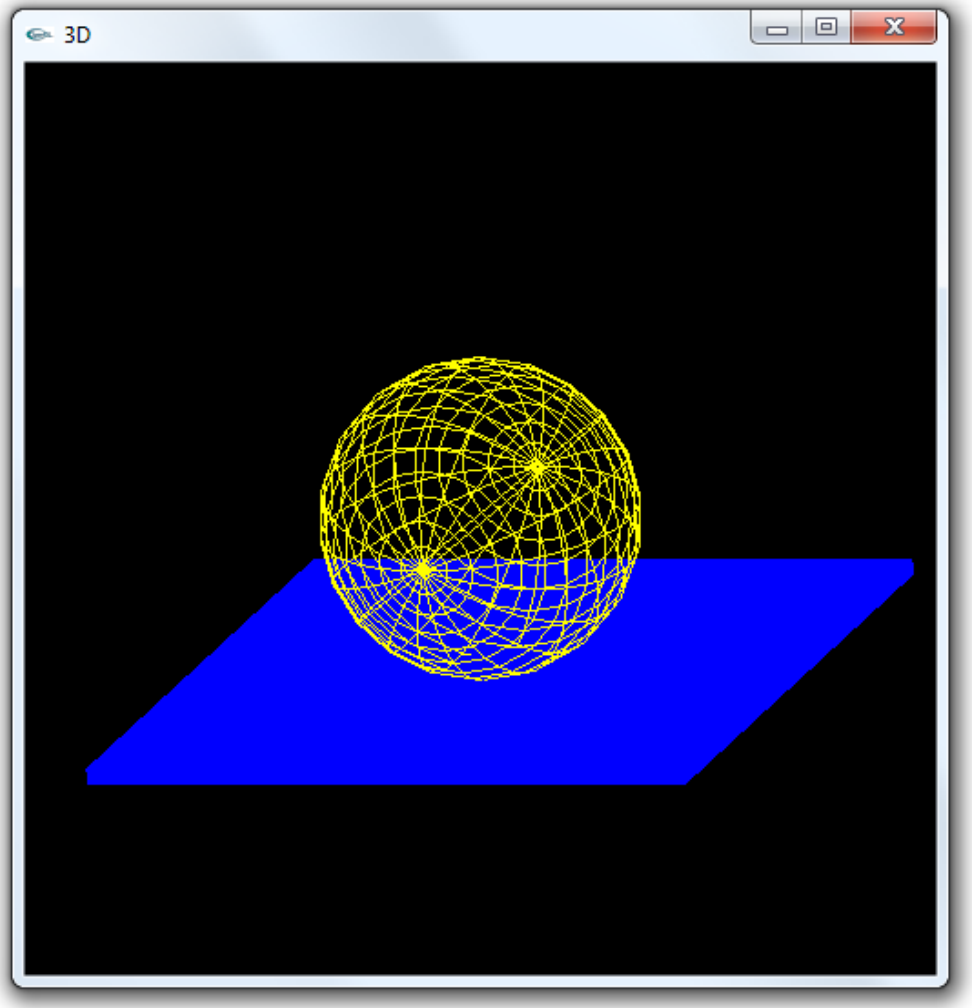
Programação OpenGL

```
1  public class Renderer extends KeyAdapter implements GLEventListener {
2      ...
3
4      @Override
5      public void keyPressed(KeyEvent e) {
6
7          switch (e.getKeyCode()) {
8              case KeyEvent.VK_PAGE_UP://faz zoom-in
9                  delta = delta * 1.1f;
10                 break;
11             case KeyEvent.VK_PAGE_DOWN://faz zoom-out
12                 delta = delta * 0.809f;
13                 break;
14             case KeyEvent.VK_UP://gira sobre o eixo-x
15                 alpha = alpha - 1;
16                 break;
17             case KeyEvent.VK_DOWN://gira sobre o eixo-x
18                 alpha = alpha + 1;
19                 break;
20             case KeyEvent.VK_LEFT://gira sobre o eixo-y
21                 beta = beta + 1;
22                 break;
23             case KeyEvent.VK_RIGHT://gira sobre o eixo-y
24                 beta = beta - 1;
25                 break;
26         }
27     }
28 }
```

Programação OpenGL

```
1  public static void main(String[] args) {
2    //acelera o rendering
3    GLCapabilities caps = new GLCapabilities();
4    caps.setDoubleBuffered(true);
5    caps.setHardwareAccelerated(true);
6
7    //cria o painel e adiciona um ouvinte GLEventListener
8    Renderer r = new Renderer();
9    GLCanvas canvas = new GLCanvas(caps);
10   canvas.addGLEventListener(r);
11
12   //cria uma janela e adiciona o painel
13   JFrame frame = new JFrame("Aplicao JOGL Simples");
14   frame.addKeyListener(r);
15   frame.getContentPane().add(canvas);
16   frame.setSize(400, 400);
17   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18
19   //inicializa o sistema e chama display() a 60 fps
20   Animator animator = new FPSAnimator(canvas, 60);
21   frame.setLocationRelativeTo(null);
22   frame.setVisible(true);
23   animator.start();
24 }
```

Programação OpenGL



Programação OpenGL

- Armazenando e restaurando transformações

```
1  public void display(GLAutoDrawable drawable) {
2    GL gl = drawable.getGL();
3    GLUT glut = new GLUT();
4
5    //limpa o buffer
6    gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
7
8    //define que a matrix  a de modelo
9    gl.glMatrixMode(GL.GL_MODELVIEW);
10   gl.glLoadIdentity();
11
12   gl.glScalef(delta, delta, delta); //faa a escala de todos objetos
13
14   gl.glPushMatrix(); //armazena a matriz corrente
15     gl.glTranslatef(-3, 0, 0);
16     gl.glRotatef(beta, 0, 1, 0);
17     gl.glRotatef(alpha, 1, 0, 0);
18     gl.glColor3f(1, 1, 0);
19     glut.glutWireSphere(1, 20, 20);
20   gl.glPopMatrix(); //restaura a matriz anterior
21
22   gl.glPushMatrix(); //armazena a matriz corrente
23     gl.glTranslatef(3, 0, 0);
24     gl.glRotatef(beta, 0, 1, 0);
25     gl.glRotatef(alpha, 1, 0, 0);
26     gl.glColor3f(1, 0, 0);
27     glut.glutWireSphere(1, 20, 20);
28   gl.glPopMatrix(); //restaura a matriz anterior
29
30   //fora o desenho das primitivas
31   gl.glFlush();
32 }
```

Programação OpenGL