

# Transformações Geométricas 3D

SCC0250- Computação Gráfica

Prof<sup>a</sup>. Rosane Minghim

<https://edisciplinas.usp.br/>

*[rminghim@icmc.usp.br](mailto:rminghim@icmc.usp.br)*

P.A.E. Eric Macedo Cabral

*[cabral.eric@usp.br](mailto:cabral.eric@usp.br)*

Instituto de Ciências Matemáticas e de Computação (ICMC)

Universidade de São Paulo (USP)

baseado no material de anos anteriores, vários autores

27 de agosto de 2018



# Sumário

- 1 Introdução
- 2 Transformações Básicas
  - Translação 3D
  - Escala 3D
  - Rotação 3D
  - Composto Transformações 3D
- 3 Outras Transformações 3D
  - Reflexão 3D
  - Cisalhamento 3D
- 4 Transformações Afim
- 5 Programação OpenGL

# Sumário

- 1 Introdução
- 2 Transformações Básicas
  - Translação 3D
  - Escala 3D
  - Rotação 3D
  - Composto Transformações 3D
- 3 Outras Transformações 3D
  - Reflexão 3D
  - Cisalhamento 3D
- 4 Transformações Afim
- 5 Programação OpenGL

# Introdução

- Métodos para transformações geométricas 3D são extensões de métodos 2D, 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  $4 \times 4$

# Sumário

- 1 Introdução
- 2 Transformações Básicas**
  - Translação 3D
  - Escala 3D
  - Rotação 3D
  - Composto Transformações 3D
- 3 Outras Transformações 3D
  - Reflexão 3D
  - Cisalhamento 3D
- 4 Transformações Afim
- 5 Programação OpenGL

# Sumário

- 1 Introdução
- 2 Transformações Básicas
  - Translação 3D
  - Escala 3D
  - Rotação 3D
  - Composto Transformações 3D
- 3 Outras Transformações 3D
  - Reflexão 3D
  - Cisalhamento 3D
- 4 Transformações Afim
- 5 Programação OpenGL

# Translação 3D

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

$$x' = x + t_x$$

$$y' = y + t_y$$

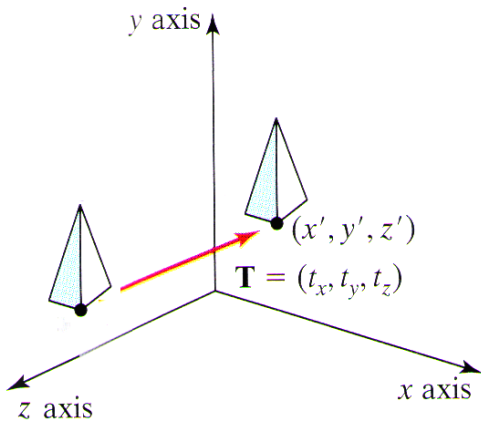
$$z' = z + t_z$$

- 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

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

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

# Sumário

- 1 Introdução
- 2 Transformações Básicas
  - Translação 3D
  - Escala 3D
  - Rotação 3D
  - Composto Transformações 3D
- 3 Outras Transformações 3D
  - Reflexão 3D
  - Cisalhamento 3D
- 4 Transformações Afim
- 5 Programação OpenGL

# 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

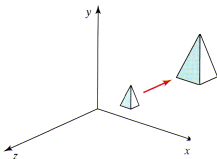


Figura: Dobrar o tamanho de um objeto também afasta o mesmo da origem

- 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

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

# Sumário

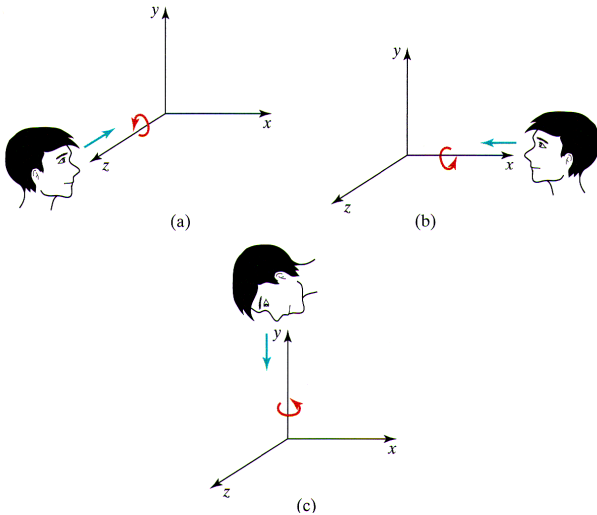
- 1 Introdução
- 2 Transformações Básicas**
  - Translação 3D
  - Escala 3D
  - Rotação 3D**
  - Composto Transformações 3D
- 3 Outras Transformações 3D
  - Reflexão 3D
  - Cisalhamento 3D
- 4 Transformações Afim
- 5 Programação OpenGL



# 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 torno 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 estendida para uma rotação 3D ao redor do eixo  $z$

$$x' = x \cos \theta - y \sin \theta$$

$$y' = x \sin \theta + y \cos \theta$$

$$z' = z$$

- Na forma matricial usando coordenadas homogêneas

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

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

# Rotação 3D nos Eixos Coordenados

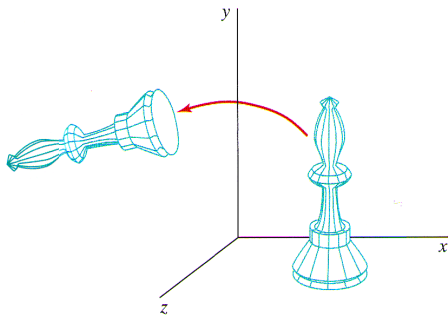
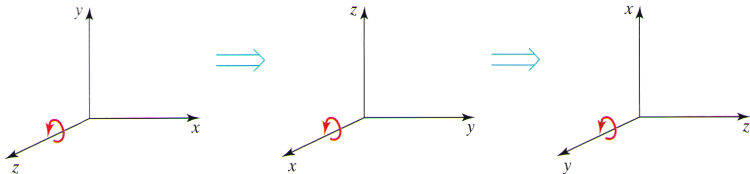


Figura: Rotação de um objeto em torno do eixo- $z$

# 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  $z$

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



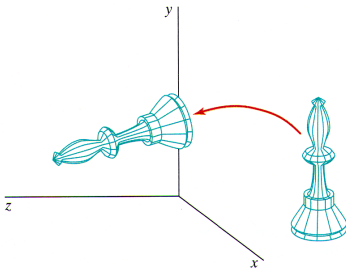
# Rotação 3D nos Eixos Coordenados

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

$$y' = y \cos \theta - z \sin \theta$$

$$z' = y \sin \theta + z \cos \theta$$

$$x' = x$$



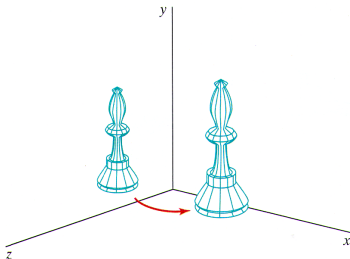
# Rotação 3D nos Eixos Coordenados

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

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

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

$$y' = y$$



# Rotação 3D nos Eixos Coordenados

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

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

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



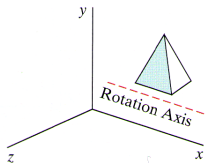
# Rotação 3D Inversa

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

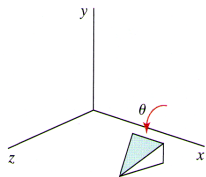
# Rotação 3D Geral

- 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

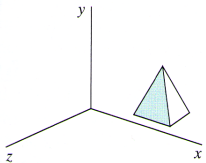
# Rotação 3D Geral



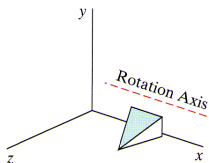
(a)  
Original Position of Object



(c)  
Rotate Object Through Angle  $\theta$



(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

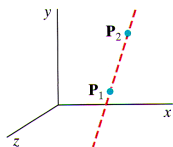
# Rotação 3D Geral

- 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

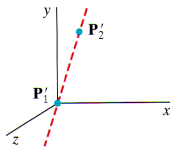
# Rotação 3D Geral

- 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- $z$

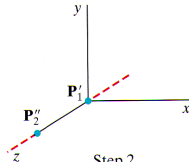
# Rotação 3D Geral



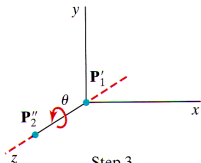
Initial  
Position



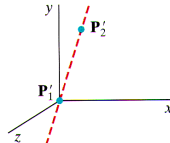
Step 1  
Translate  
 $P_1$  to the Origin



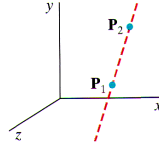
Step 2  
Rotate  $P'_2$   
onto the z Axis



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

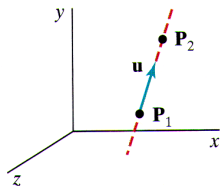
# Rotação 3D Geral

- 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

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

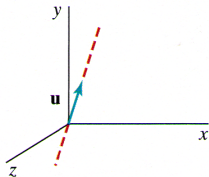




# Rotação 3D Geral

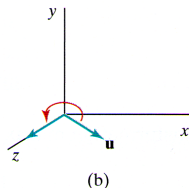
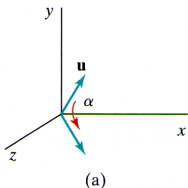
- O primeiro passo da sequência de rotação é definir uma matriz de translação para reposicionar o eixo de rotação de forma que esse passe pela origem
  - Como a rotação se dá no sentido anti-horário, movemos o ponto  $P_1$  para a origem, ou seja

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



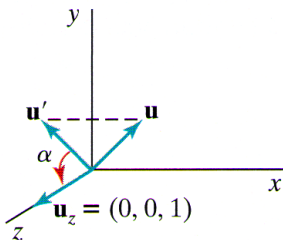
# Rotação 3D Geral

- 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 ( $\alpha$ ) 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  $\alpha$  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  $\alpha$  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

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

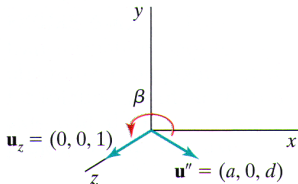
# Rotação 3D Geral

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

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

# 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  $\beta$  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  $\alpha$  e  $\beta$  nós alinhamos o eixo de rotação sobre o eixo  $z$ , então agora a rotação de um ângulo  $\theta$  pode ser aplicada

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos \theta & -\text{sen } \theta & 0 & 0 \\ \text{sen } \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# 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 sequência 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  $3 \times 3$  superior é ortonormal

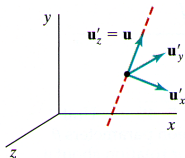
# Rotação 3D Geral

- 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

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

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

$$\mathbf{u}'_x = \mathbf{u}'_y \times \mathbf{u}'_z = (u'_{x1}, u'_{x2}, u'_{x3})$$



# 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  $\mathbf{u}'_z = \mathbf{u}$

# Sumário

- 1 Introdução
- 2 Transformações Básicas**
  - Translação 3D
  - Escala 3D
  - Rotação 3D
  - Compondo Transformações 3D
- 3 Outras Transformações 3D
  - Reflexão 3D
  - Cisalhamento 3D
- 4 Transformações Afim
- 5 Programação OpenGL



# Compondo Transformações 3D

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

# Sumário

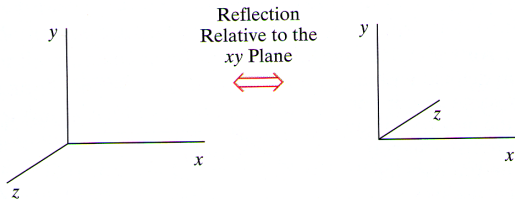
- 1 Introdução
- 2 Transformações Básicas
  - Translação 3D
  - Escala 3D
  - Rotação 3D
  - Composto Transformações 3D
- 3 Outras Transformações 3D**
  - Reflexão 3D
  - Cisalhamento 3D
- 4 Transformações Afim
- 5 Programação OpenGL

# Sumário

- 1 Introdução
- 2 Transformações Básicas
  - Translação 3D
  - Escala 3D
  - Rotação 3D
  - Composto Transformações 3D
- 3 Outras Transformações 3D
  - Reflexão 3D
  - Cisalhamento 3D
- 4 Transformações Afim
- 5 Programação OpenGL

# 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  $xy$ )

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

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

# Sumário

- 1 Introdução
- 2 Transformações Básicas
  - Translação 3D
  - Escala 3D
  - Rotação 3D
  - Composto Transformações 3D
- 3 Outras Transformações 3D**
  - Reflexão 3D
  - Cisalhamento 3D**
- 4 Transformações Afim
- 5 Programação OpenGL

# Cisalhamento 3D

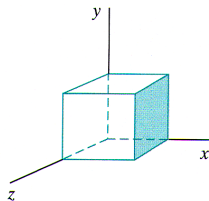
- Cisalhamento relativo aos eixos  $x$  e  $y$  é o mesmo que o já discutido em 2D, mas em 3D também é possível realizar o cisalhamento relativo ao eixo  $z$
- O cisalhamento geral em torno do eixo- $z$ , dado um ponto de referência  $z_{ref}$  é produzido pela seguinte matriz

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

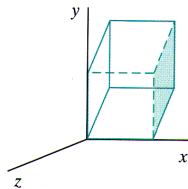
- O efeito de  $sh_{zx}$  e  $sh_{zy}$  é alterar os valores das coordenadas  $x$  e  $y$  uma quantidade proporcional a distância de  $z_{ref}$ , enquanto mantém a coordenada  $z$  inalterada

# Cisalhamento 3D

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



(a)



(b)



# Sumário

- 1 Introdução
- 2 Transformações Básicas
  - Translação 3D
  - Escala 3D
  - Rotação 3D
  - Composto Transformações 3D
- 3 Outras Transformações 3D
  - Reflexão 3D
  - Cisalhamento 3D
- 4 Transformações Afim
- 5 Programação OpenGL

# 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 afins

# Sumário

- 1 Introdução
- 2 Transformações Básicas
  - Translação 3D
  - Escala 3D
  - Rotação 3D
  - Composto Transformações 3D
- 3 Outras Transformações 3D
  - Reflexão 3D
  - Cisalhamento 3D
- 4 Transformações Afim
- 5 Programação OpenGL

# 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); //remoção de superfície oculta
13
14        gl.glMatrixMode(GL.GL_PROJECTION); //define que a matriz A é a de ←
15        //projecção
16        gl.glLoadIdentity(); //carrega a matriz de identidade
17        gl.glOrtho(-5, 5, -5, 5, -5, 5); //define uma projecção ortográfica
18    }
19
20    public void reshape(GLAutoDrawable drawable, int x, int y, int width, int ←
21        height) {
22
23    }
24
25    public void displayChanged(GLAutoDrawable drawable, boolean modeChanged, ←
26        boolean deviceChanged) {
27
28    }
29
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     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 matriz 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.glRotatf(beta, 0, 1, 0);
17        gl.glRotatf(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        //forÃa o desenho das primitivas
29        gl.glFlush();
30    }
31
32    private float alpha;
33    private float beta;
34    private float delta;
35 }
    
```

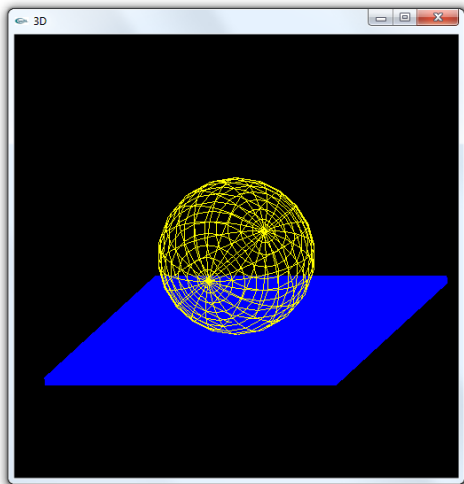
# Programação OpenGL

```
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("Aplicação 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 matriz  $\tilde{A}$  é a de modelo
9     gl.glMatrixMode(GL.GL_MODELVIEW);
10    gl.glLoadIdentity();
11
12    gl.glScalef(delta, delta, delta); //fa $\tilde{A}$  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    //for $\tilde{A}$  o desenho das primitivas
31    gl.glFlush();
32 }
```

# Programação OpenGL

