Transformações Geométricas 3D

SCC0250/0650 - Computação Gráfica

Profª. Rosane Minghim
https://edisciplinas.usp.br/course/view.php?id=61213
https://edisciplinas.usp.br/course/view.php?id=61210
rminghim@icmc.usp.br
P.A.E. Diego Cintra e Fábio Felix
diegocintra@usp.br, f_diasfabio@usp.br

Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)
baseado no material de anos anteriores, vários autores

16 de abril de 2018

Sumário

Sumário

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 4×4

Sumário

Sumário

Translação 3D

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

$$x' = x + t_x$$

$$y' = y + t_y$$

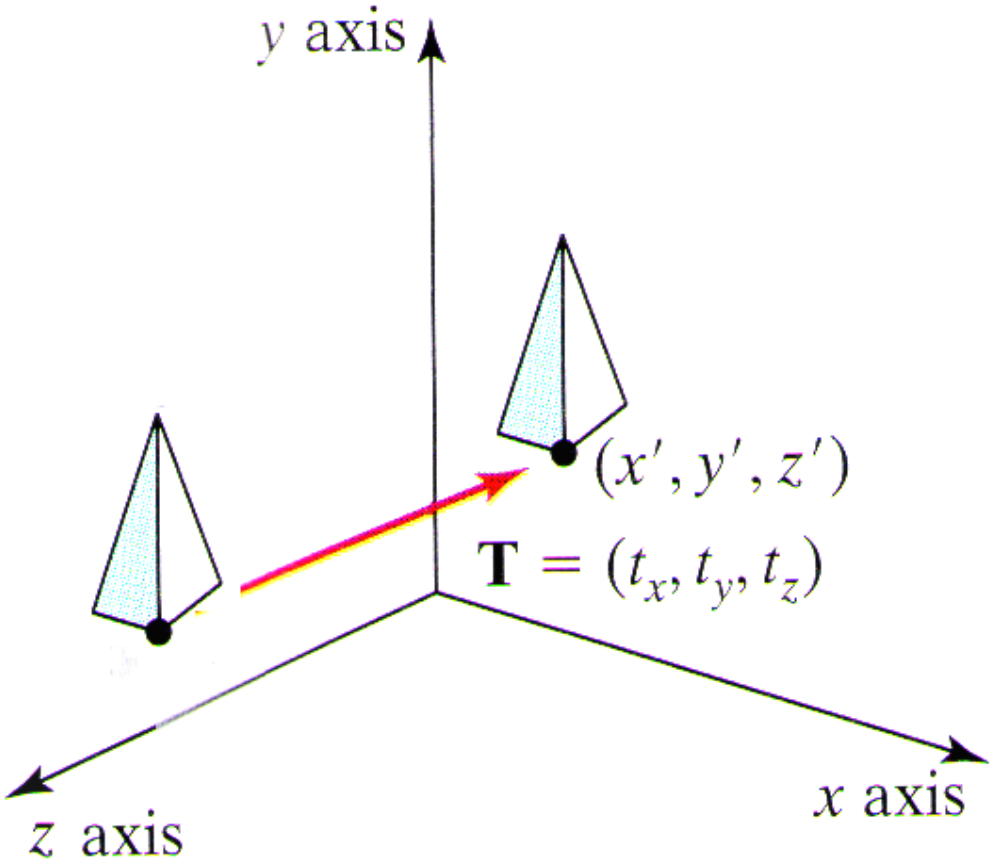
$$z' = z + t_z$$

- 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 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

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

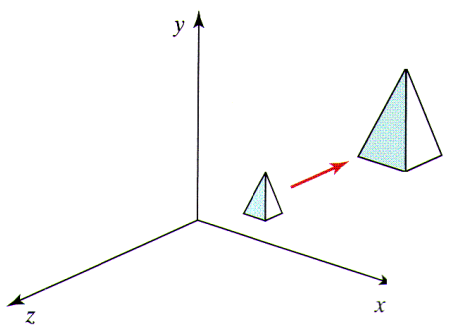


Figura: Dobrar o tamanho de um objeto também afasta o mesmo da origem

- 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

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

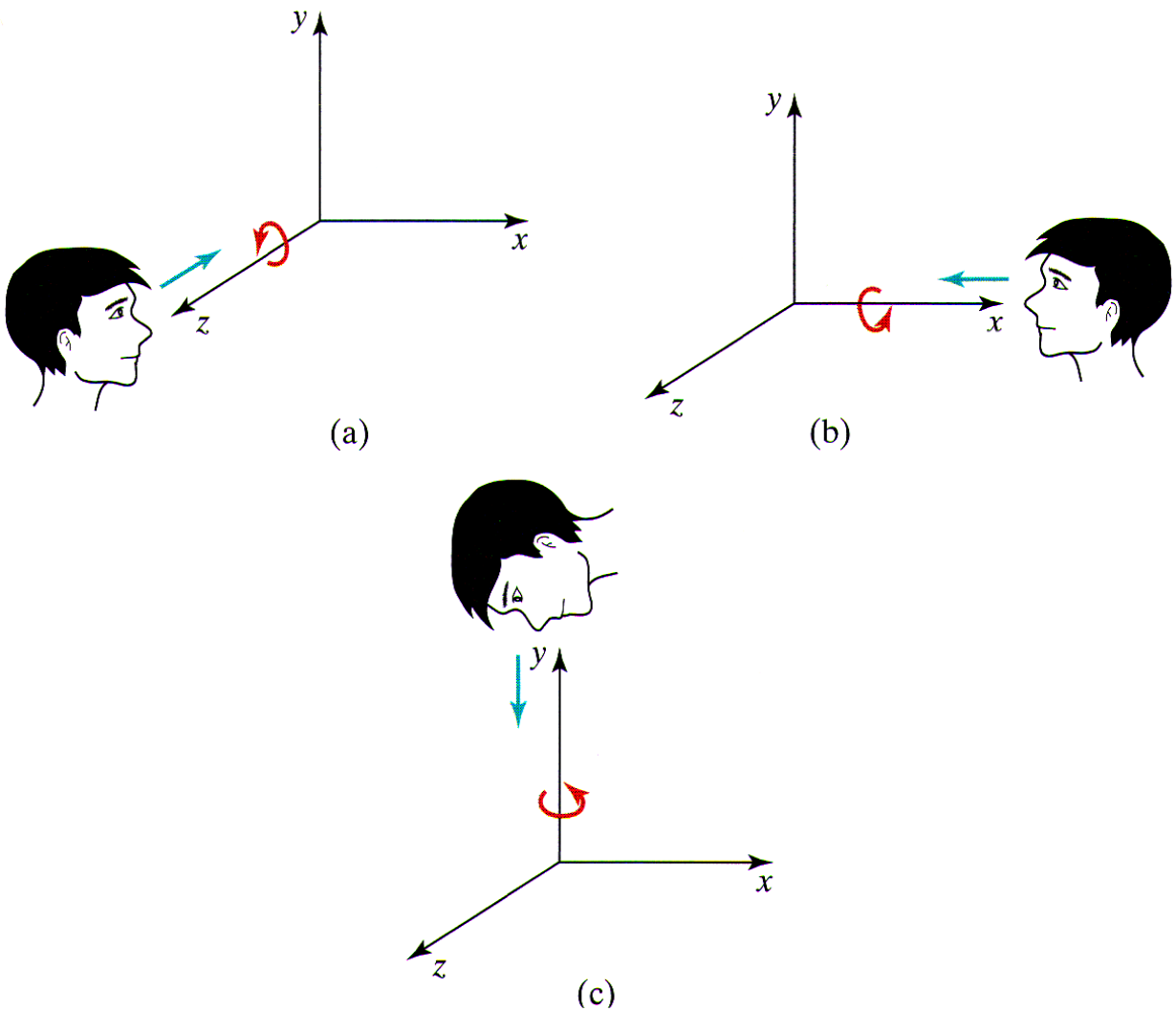
Sumário

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 para 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

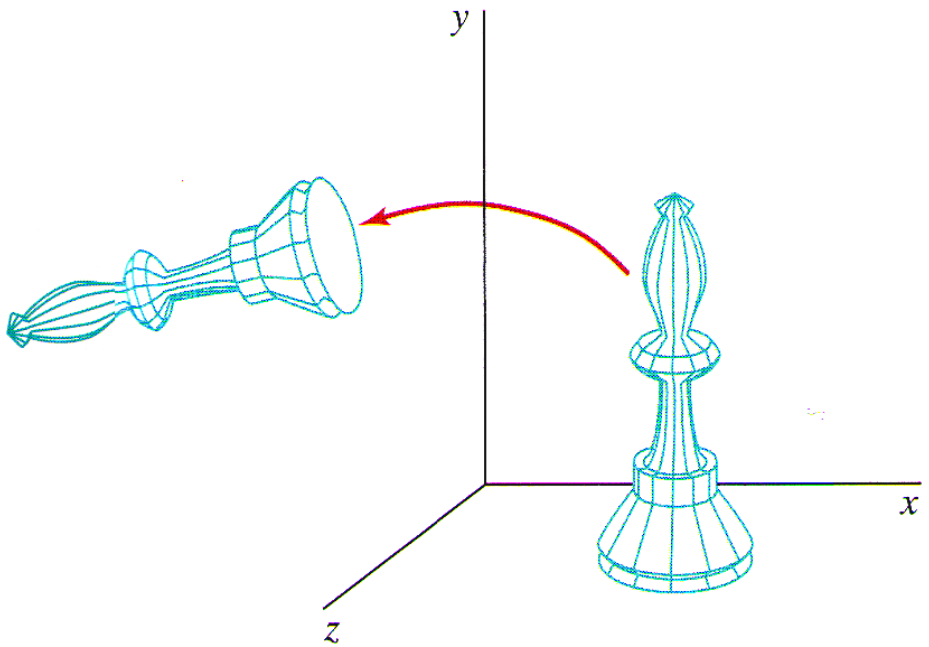
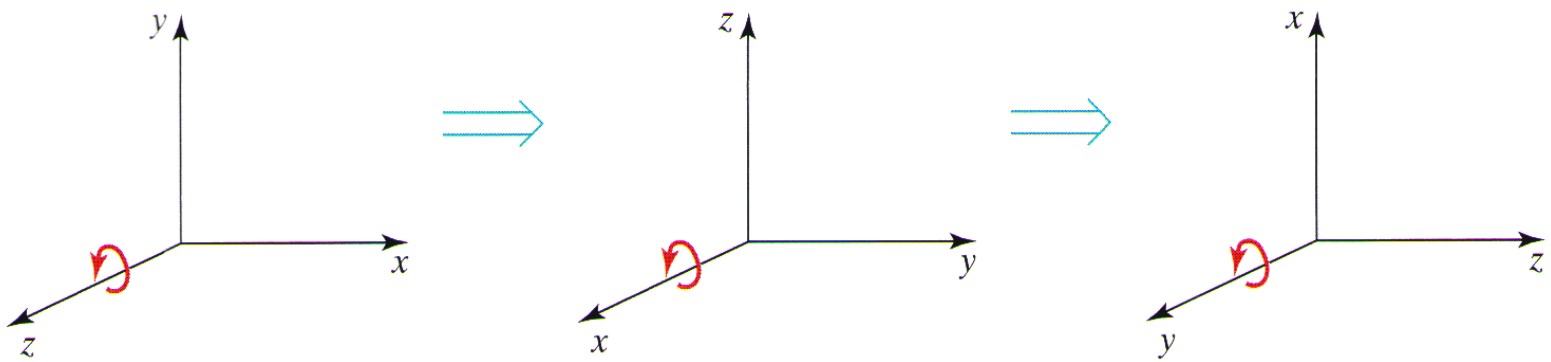


Figura: Rotação de um objeto em torno do eixo-z

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

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



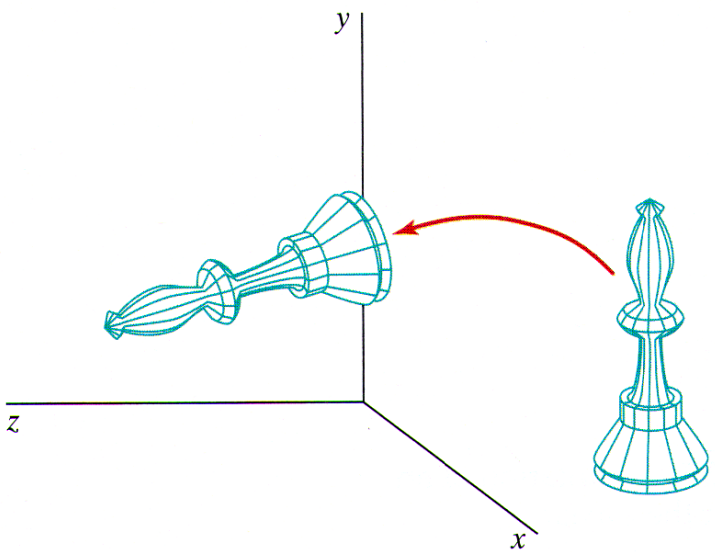
Rotação 3D nos Eixos Coordenados

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

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

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

$$x' = x$$



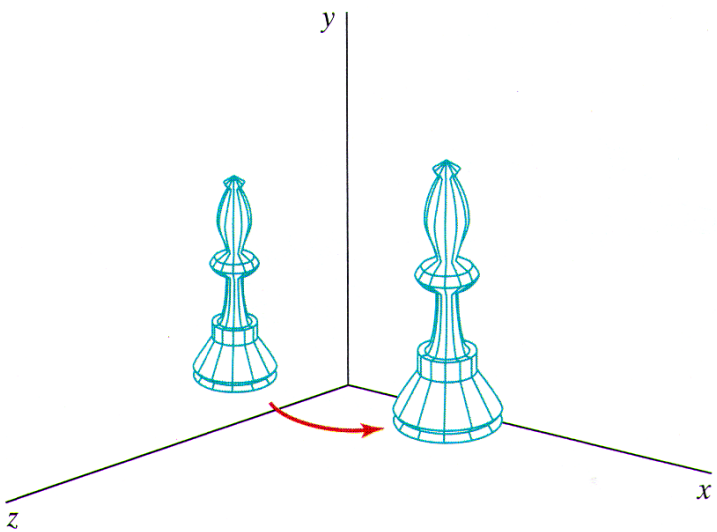
Rotação 3D nos Eixos Coordenados

- 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

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

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

Rotação 3D Inversa

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

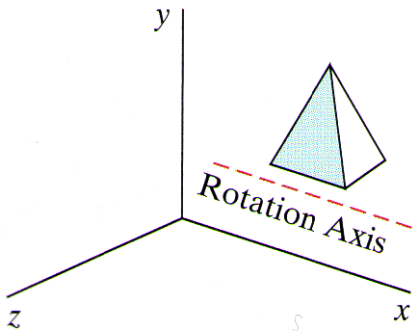
- Como somente o sinal do seno é alterado, a inversa pode ser obtida trocando as linhas pelas colunas, isto é $\mathbf{R}^{-1} = \mathbf{R}^{T}$

Rotação 3D Geral

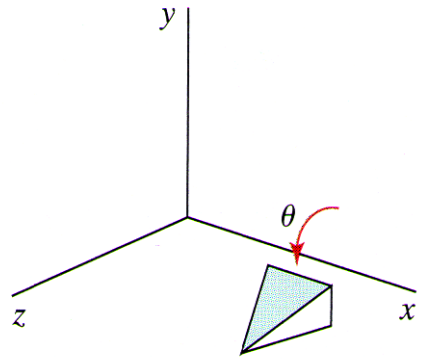
- A rotação em torno de qualquer eixo pode ser obtida como a combinação de rotações e translações

- 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

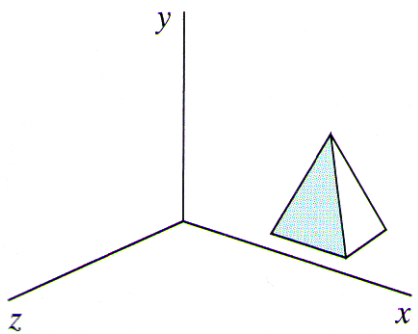
Rotação 3D Geral



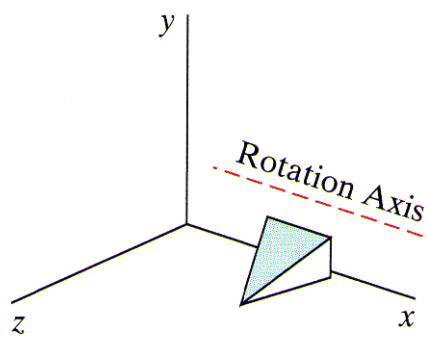
(a)
Original Position of Object



(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

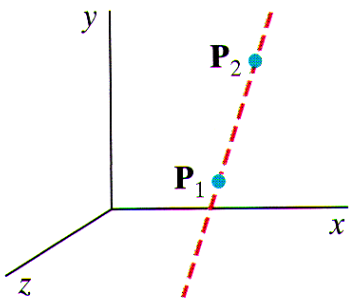
- Quando o eixo de rotação não é paralelo aos eixos de coordenadas, algumas transformações adicionais são necessárias
  - Também são necessárias rotações para alinhar o eixo de rotação com o eixo de coordenadas escolhido e para trazer de volta o eixo de rotação para a posição original

Rotação 3D Geral

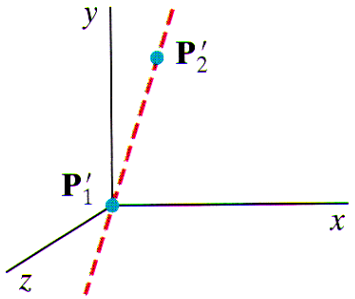
- Dado o eixo de rotação e o ângulo de rotação, isso pode ser feito como
  1. Transladar o objeto de forma que o eixo de rotação passe pela origem do sistema de coordenadas
  2. Rotacionar o objeto para que o eixo de rotação coincida com um dos eixos de coordenadas
  3. Realizar a rotação sobre o eixo 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

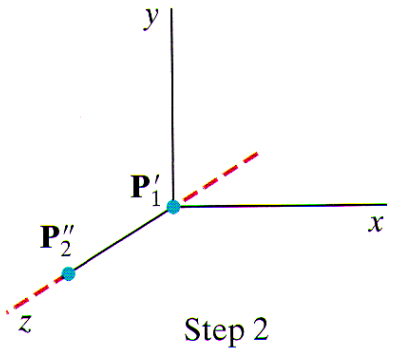
Rotação 3D Geral



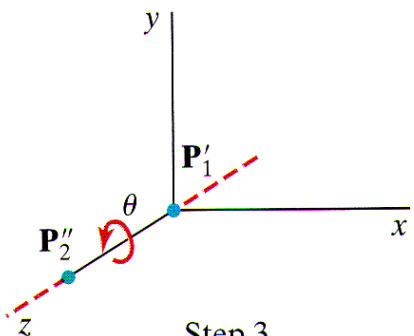
Initial Position



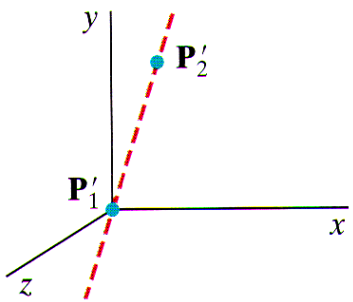
Step 1
Translate
$\mathbf{P}_1$ to the Origin



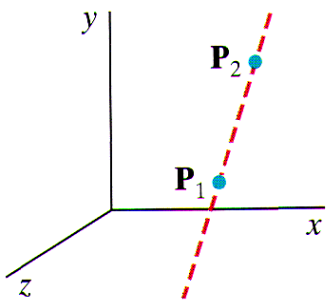
Step 2
Rotate $\mathbf{P}_2'$
onto the z Axis



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

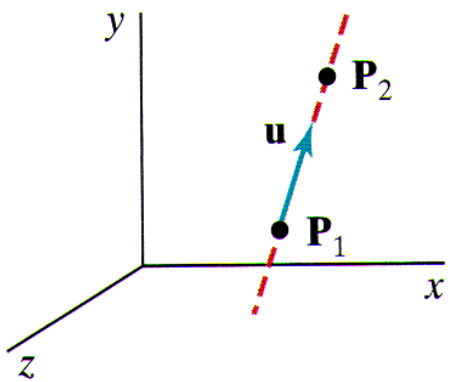
Rotação 3D Geral

- 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

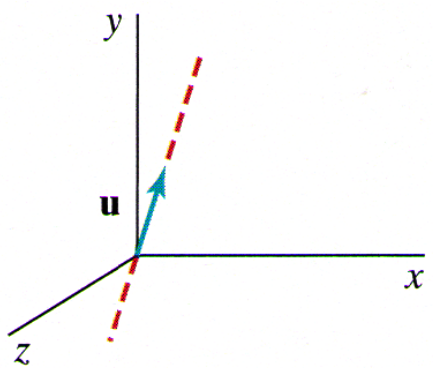
$$\mathbf{u} = \frac{\mathbf{V}}{|\mathbf{V}|} = (a, b, c)$$



Rotação 3D Geral

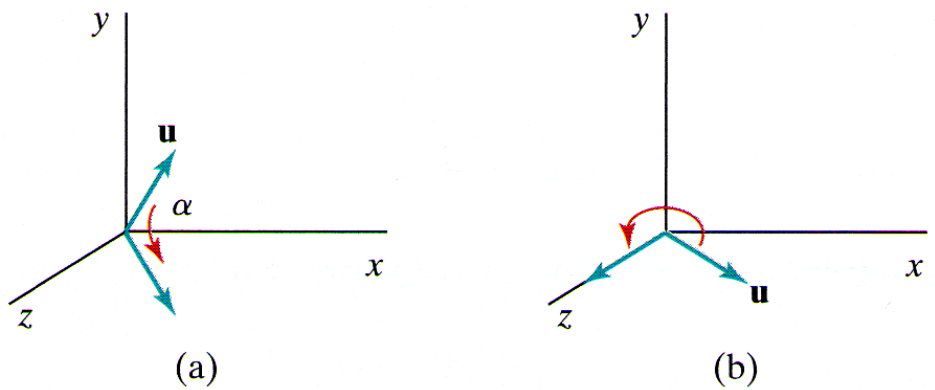
- O primeiro passo da sequencia de rotação é definir uma matriz de translação para reposicionar o eixo de rotação de forma que esse passe pela origem
  - Como a rotação se dá no sentido anti-horário, movemos o ponto $\mathbf{P_1}$ para a origem, ou seja

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



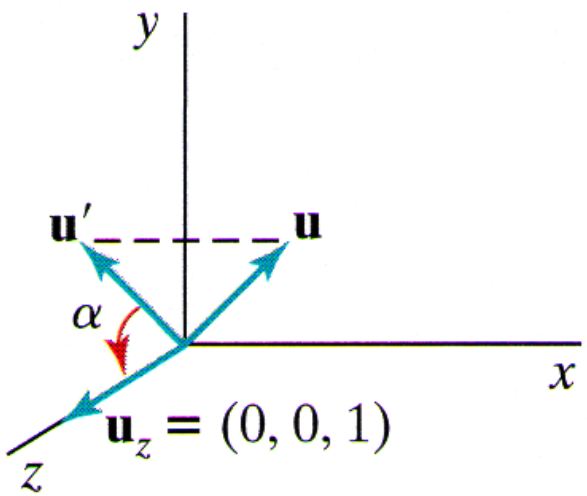
Rotação 3D Geral

- 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

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

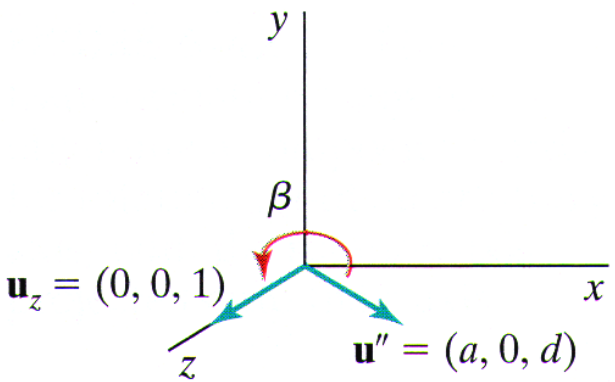
Rotação 3D Geral

- 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

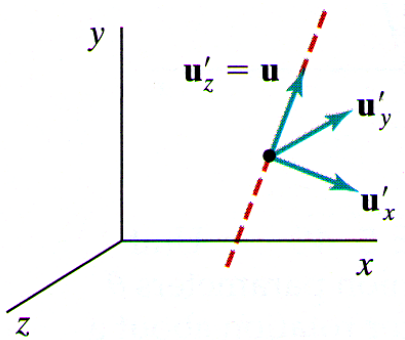
Rotação 3D Geral

- 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

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

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

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



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

Sumário

Compondo Transformações 3D

- Assim como nas transformações 2D, as transformações 3D são compostas multiplicando matrizes

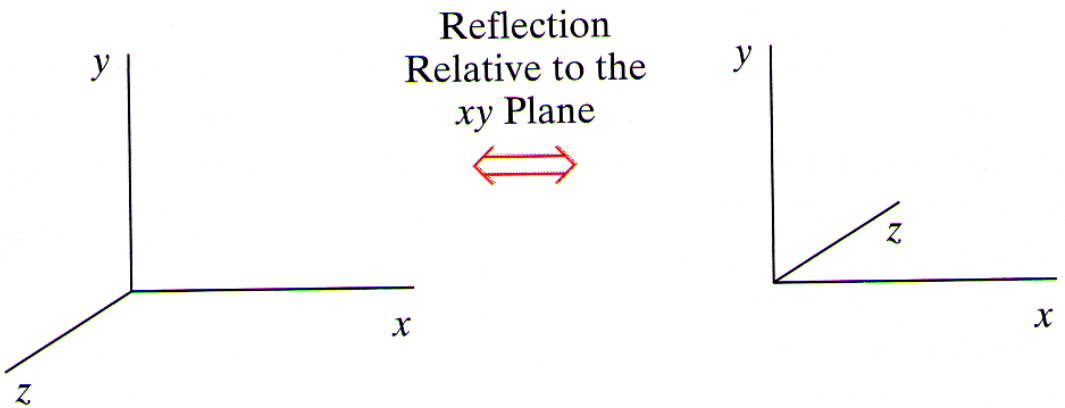
- Novamente a tranformações mais a direita será a primeira a ser aplicada, e é necessário observar se a API gráfica utilizada é pós- ou pré-multiplicada

Sumário

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)

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

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

Sumário

Cisalhamento 3D

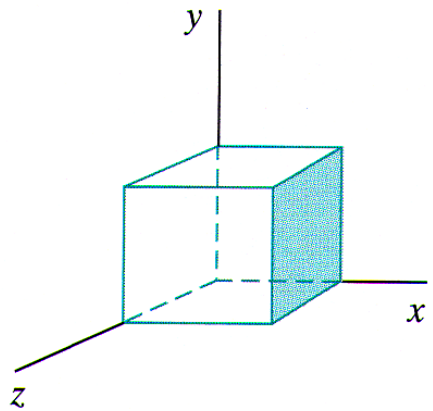
- Cisalhamento relativo aos eixos x e y é o mesmo que o já discutido em 2D, mas em 3D também é possível realizar o cisalhamento relativo ao eixo z
- O cisalhamento geral em torno do eixo-z, dado um ponto de referência z_{ref} é produzido pela seguinte matriz

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

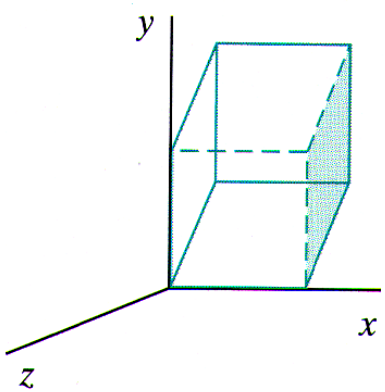
- O efeito de sh_{zx} e sh_{zy} é alterar os valores das coordenadas x e y uma quantidade proporcional a distância de z_{ref}, enquanto mantém a coordenada z inalterada

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

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

Sumário

Ambiente Visual 2D

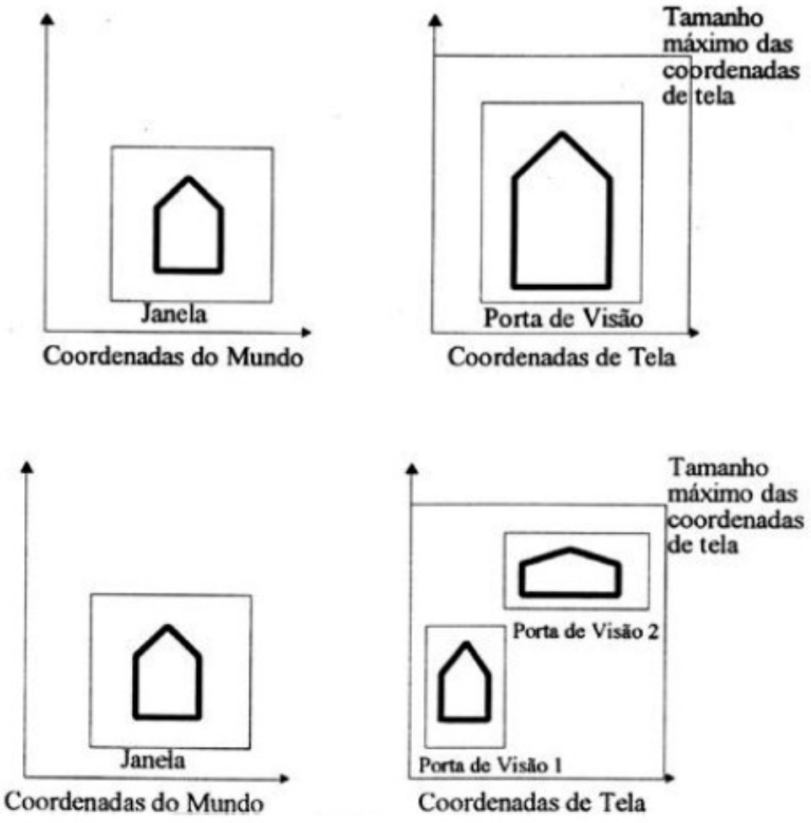
- No caso 2D, é necessário definir em uma janela a porção o universo que desejamos mapear na tela.
  - Essa área é chamada de janela de seleção, ou window.

```
1 void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top)
```

- É necessário definir também em que parte do monitor deseja-se exibir o conteúdo da window.
  - Chamamos essa região de viewport (ou janela de exibição).

```
1 void glViewport(GLint x, GLint y, GLsizei width, GLsizei height)
```

Ambiente Visual 2D



Exemplo

- Baixar e compilar o arquivo (ExemploViewport2D).

Transformações Geométricas

- O objeto transformado deve inicialmente ser transladado para a origem.
- Então deve ser aplicada a transformação.
- Por fim o objeto deve ser transladado novamente para sua posição inicial.
- No OpenGL a sequência de comandos segue a ordem invertida.

Matriz de Transformação

- Todos os comandos de transformação são compostos em uma matriz de transformação.
- Cada novo comando é acumulado, alterando a configuração da matriz.
- Ao especificar um novo vértice, a sua posição é calculada aplicando-se a matriz de transformação corrente às suas coordenadas.
- A matriz de transformação é inicializada com a matriz identidade (não altera os objetos)
  - glLoadIdentity();

Transformações Geométricas

- Translação
  - void glTranslatef(GLfloat x, y, z)
  - x, y, z: representam a quantidade a ser transladada no respectivo eixo.
- Rotação
  - void glRotatef(GLfloat angulo, x, y, z)
  - angulo: especifica o ângulo de rotação (em graus).
  - x, y, z: o eixo a ser realizada a rotação.
- Escala
  - void glScalef(GLfloat x, y, z)
  - x, y, z: Representam o fator de escala nos respectivos eixos.

Programação OpenGL

```
1  #include <GLUT/glut.h>
2
3  float alpha = 0;
4  float beta = 0;
5  float delta = 1;
6
7  void init() {
8      glClearColor(0, 0, 0, 0); //define a cor de fundo
9      glEnable(GL_DEPTH_TEST); //remocao de superficie oculta
10     glMatrixMode(GL_PROJECTION); //define que a matrix eh a de projecao
11     glLoadIdentity(); //carrega a matrix de identidade
12     glOrtho(-5, 5, -5, 5, -5, 5); //define uma projecao ortografica
13 }
```

Programação OpenGL

```
1  void display() {
2      glClearColor(0.0,0.0,0.0,0.0);
3      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
4
5      //define que a matrix eh a de modelo
6      glMatrixMode(GL_MODELVIEW);
7      glLoadIdentity(); //carrega matrix identidade
8
9      //rotaciona e escala uma esfera 'aramado'
10     glRotatef(beta, 0, 1, 0);
11     glRotatef(alpha, 1, 0, 0);
12     glScalef(delta, delta, delta);
13     glColor3f(1, 1, 0);
14     glutWireSphere(1.0f, 20, 20);
15
16     //desenha um 'piso' sob a esfera
17     glTranslatef(0, -1, 0);
18     glScalef(4, 0.1f, 4);
19     glColor3f(0, 0, 1);
20     glutSolidCube(1.0f);
21
22     //forca o desenho das primitivas
23     glFlush();
24 }
```

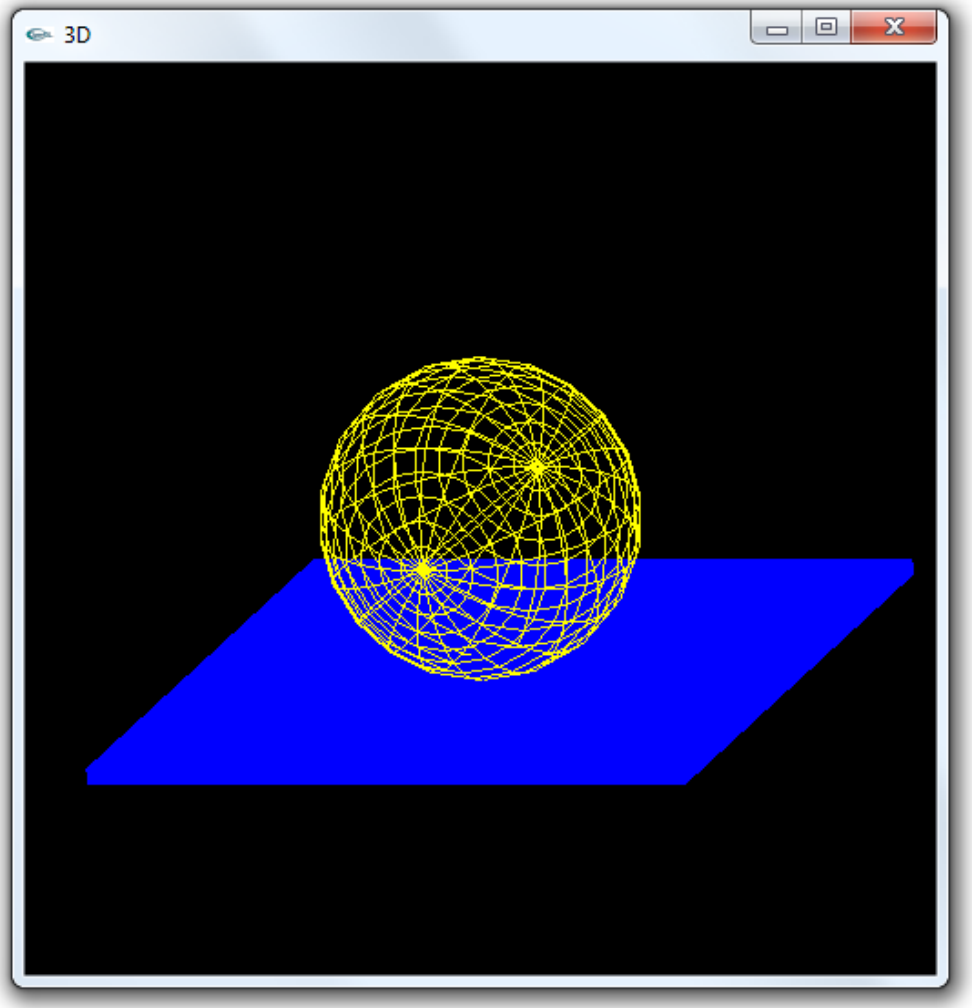
Programação OpenGL

```
1  void keyboard(unsigned char key, int x, int y)
2  {
3      if(key == 'q')
4          delta = delta * 1.1f;
5      if(key == 'w')
6          delta = delta * 0.809f;
7      if(key == GLUT_KEY_UP)
8              alpha = alpha - 1;
9      if(key == GLUT_KEY_DOWN)
10             alpha = alpha + 1;
11     if(key == GLUT_KEY_LEFT)
12             beta = beta + 1;
13     if(key == GLUT_KEY_RIGHT)
14             beta = beta - 1;
15
16     glutPostRedisplay();
17 }
```

Programação OpenGL

```
1  int main(int argc, char **argv)
2  {
3      glutInit(&argc, argv);
4      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
5      glutInitWindowPosition(100, 100);
6      glutInitWindowSize(400, 400);
7      glutCreateWindow("Aplicacao JOGL Simples");
8      gluOrtho2D(0.0, 400, 0.0, 600);
9      init();
10     glutKeyboardFunc(keyboard);
11     glutSpecialFunc(keySpecial);
12     glutDisplayFunc(display);
13     glutMainLoop();
14 }
```

Programação OpenGL

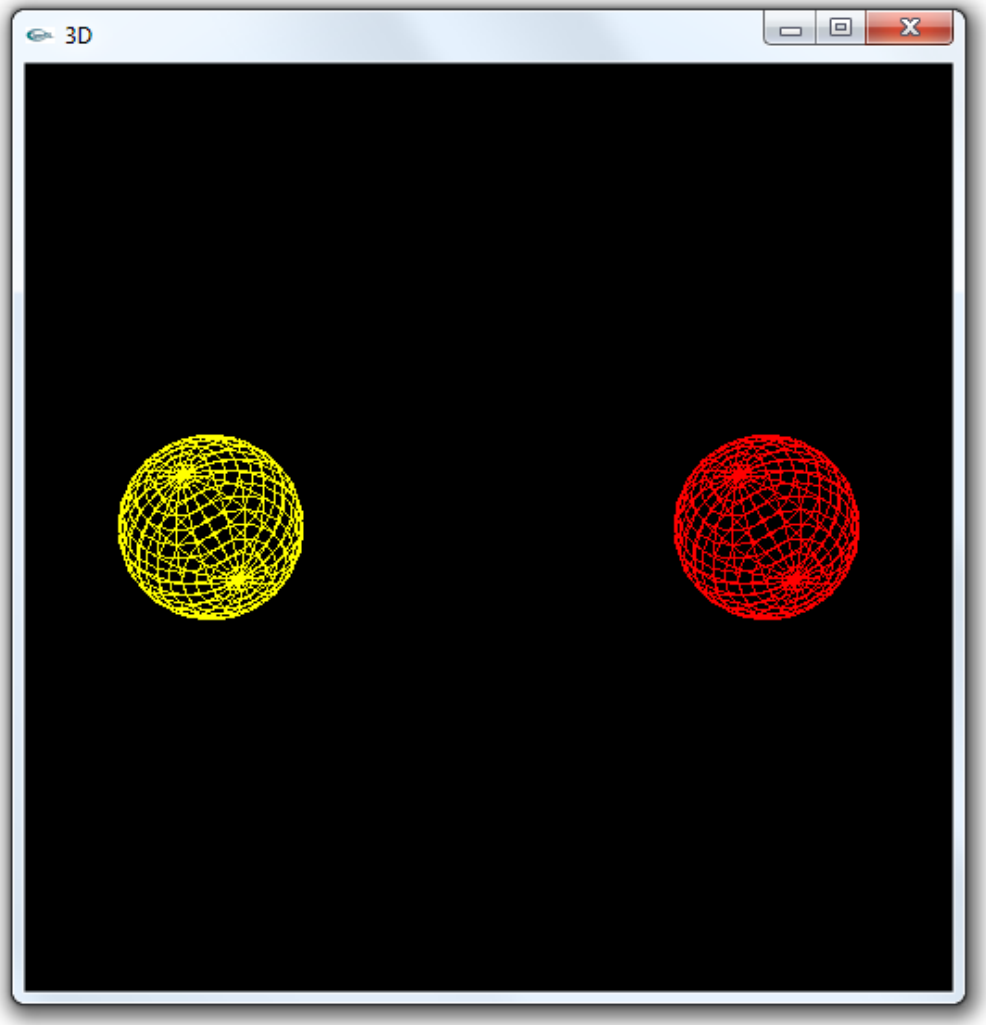


Programação OpenGL

- Armazenando e restaurando transformações

```
1  void display() {
2
3      glClearColor(0.0,0.0,0.0,0.0);
4      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
5
6      //define que a matrix eh a de modelo
7      glMatrixMode(GL_MODELVIEW);
8      glLoadIdentity();
9
10     glScalef(delta, delta, delta); //faca a escala de todos objetos
11
12     glPushMatrix(); //armazena a matriz corrente
13     glTranslatef(-3, 0, 0);
14     glRotatef(beta, 0, 1, 0);
15     glRotatef(alpha, 1, 0, 0);
16     glColor3f(1, 1, 0);
17     glutWireSphere(1, 20, 20);
18     glPopMatrix(); //restaura a matriz anterior
19
20     glPushMatrix(); //armazena a matriz corrente
21     glTranslatef(3, 0, 0);
22     glRotatef(beta, 0, 1, 0);
23     glRotatef(alpha, 1, 0, 0);
24     glColor3f(1, 0, 0);
25     glutWireSphere(1, 20, 20);
26     glPopMatrix(); //restaura a matriz anterior
27
28     //forca o desenho das primitivas
29     glFlush();
30 }
```

Programação OpenGL



Programação OpenGL

```
1  #include <GLUT/glut.h>
2  #include <stdlib.h>
3
4  // Variaveis para controles de navegacao
5  GLfloat angle, fAspect;
6  GLfloat rotX, rotY, rotX_ini, rotY_ini;
7  GLfloat obsX, obsY, obsZ, obsX_ini, obsY_ini, obsZ_ini;
8  int x_ini,y_ini,bot;
9
10 // Funcao que desenha um cubo cujas cores dos vertices
11 // mostram como eh o espaco RGB
12 void DesenhaCuboRGB(void)
13 {
14     // Desenhas as linhas das "bordas" do cubo
15     glColor3f(0.0f, 0.0f, 0.0f);
16     glLineWidth(1.6f);
17     glBegin(GL_LINE_LOOP); // frontal
18     glVertex3f(40.0, -40.0, 40.0);
19     glVertex3f(-40.0, -40.0, 40.0);
20     glVertex3f(-40.0, 40.0, 40.0);
21     glVertex3f(40.0, 40.0, 40.0);
22     glEnd();
23     glBegin(GL_LINE_LOOP); // posterior
24     glVertex3f(40.0, 40.0, -40.0);
25     glVertex3f(40.0, -40.0, -40.0);
26     glVertex3f(-40.0, -40.0, -40.0);
27     glVertex3f(-40.0, 40.0, -40.0);
28     glEnd();
29     glBegin(GL_LINES); // laterais
30     glVertex3f(-40.0, 40.0, -40.0);
31     glVertex3f(-40.0, 40.0, 40.0);
32     glVertex3f(-40.0, -40.0, -40.0);
33     glVertex3f(-40.0, -40.0, 40.0);
34     glVertex3f(40.0, 40.0, -40.0);
35     glVertex3f(40.0, 40.0, 40.0);
36     glVertex3f(40.0, -40.0, -40.0);
37     glVertex3f(40.0, -40.0, 40.0);
38     glEnd();
```

Programação OpenGL

```
    glBegin(GL_QUADS); // Desenha as faces do cubo preenchidas
    // Face frontal
    glColor3f(1.0f, 1.0f, 1.0f);
    glVertex3f(40.0, 40.0, 40.0);
    glVertex3f(-40.0, 40.0, 40.0);
    glVertex3f(-40.0, -40.0, 40.0);
    glVertex3f(40.0, -40.0, 40.0);
    // Face posterior
    glColor3f(0.0f, 0.0f, 0.0f);
    glVertex3f(40.0, 40.0, -40.0);
    glVertex3f(40.0, -40.0, -40.0);
    glVertex3f(-40.0, -40.0, -40.0);
    glVertex3f(-40.0, 40.0, -40.0);
    // Face lateral esquerda
    glColor3f(0.0f, 1.0f, 0.0f);
    glVertex3f(-40.0, 40.0, 40.0);
    glVertex3f(-40.0, 40.0, -40.0);
    glVertex3f(-40.0, -40.0, -40.0);
    glVertex3f(-40.0, -40.0, 40.0);
    // Face lateral direita
    glColor3f(1.0f, 0.0f, 0.0f);
    glVertex3f(40.0, 40.0, 40.0);
    glVertex3f(40.0, -40.0, 40.0);
    glVertex3f(40.0, -40.0, -40.0);
    glVertex3f(40.0, 40.0, -40.0);
    // Face superior
    glColor3f(1.0f, 1.0f, 0.0f);
    glVertex3f(-40.0, 40.0, -40.0);
    glVertex3f(-40.0, 40.0, 40.0);
    glVertex3f(40.0, 40.0, 40.0);
    glVertex3f(40.0, 40.0, -40.0);
    // Face inferior
    glColor3f(0.0f, 0.0f, 1.0f);
    glVertex3f(-40.0, -40.0, -40.0);
    glVertex3f(40.0, -40.0, -40.0);
    glVertex3f(40.0, -40.0, 40.0);
    glVertex3f(-40.0, -40.0, 40.0);
    glEnd();
}
```

Programação OpenGL

```
1  // Funcao callback de redesenho da janela de visualizacao
2  void Desenha(void)
3  {
4      // Limpa a janela de visualizacao com a cor
5      // de fundo definida previamente
6      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
7
8      // Chama a funcao que especifica os parametros de iluminacao
9      DesenhaCuboRGB();
10     // Executa os comandos OpenGL
11     glutSwapBuffers();
12 }
13
14 // Funcao usada para especificar a posicao do observador virtual
15 void PosicionaObservador(void)
16 {
17     // Especifica sistema de coordenadas do modelo
18     glMatrixMode(GL_MODELVIEW);
19     // Inicializa sistema de coordenadas do modelo
20     glLoadIdentity();
21     // Posiciona e orienta o observador
22     glTranslatef(-obsX,-obsY,-obsZ);
23     glRotatef(rotX,1,0,0);
24     glRotatef(rotY,0,1,0);
25 }
```

Programação OpenGL

```
1  // Funcao callback de redesenho da janela de visualizacao
2  void Desenha(void)
3  {
4      // Limpa a janela de visualizacao com a cor
5      // de fundo definida previamente
6      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
7
8      // Chama a funcao que especifica os parametros de iluminacao
9      DesenhaCuboRGB();
10     // Executa os comandos OpenGL
11     glutSwapBuffers();
12 }
13
14 // Funcao usada para especificar a posicao do observador virtual
15 void PosicionaObservador(void)
16 {
17     // Especifica sistema de coordenadas do modelo
18     glMatrixMode(GL_MODELVIEW);
19     // Inicializa sistema de coordenadas do modelo
20     glLoadIdentity();
21     // Posiciona e orienta o observador
22     glTranslatef(-obsX,-obsY,-obsZ);
23     glRotatef(rotX,1,0,0);
24     glRotatef(rotY,0,1,0);
25 }
```

Programação OpenGL

```
1  // Funcao usada para especificar o volume de visualizacao
2  void EspecificaParametrosVisualizacao(void)
3  {
4      // Especifica sistema de coordenadas de projecao
5      glMatrixMode(GL_PROJECTION);
6      // Inicializa sistema de coordenadas de projecao
7      glLoadIdentity();
8      // Especifica a projecao perspectiva
9      // (angulo, aspecto, zMin, zMax)
10     gluPerspective(angle,fAspect,0.5,1000);
11     PosicionaObservador();
12 }
13
14 // Funcao callback chamada para gerenciar eventos de teclas normais (ESC)
15 void Teclado (unsigned char tecla, int x, int y)
16 {
17     if(tecla==27) // ESC
18         exit(0);
19     if (tecla == 'p') {
20         glEnable(GL_DEPTH_TEST);
21         glutPostRedisplay();
22     }
23     if (tecla == 'P') {
24         glDisable(GL_DEPTH_TEST);
25         glutPostRedisplay();
26     }
27 }
```

Programação OpenGL

```
1
2  // Funcao callback para tratar eventos de teclas especiais
3  void TeclasEspeciais (int tecla, int x, int y)
4  {
5      switch (tecla) {
6          case GLUT_KEY_HOME: if(angle>=10) angle -=5;
7              break;
8          case GLUT_KEY_END: if(angle<=150) angle +=5;
9              break;
10     }
11     EspecificaParametrosVisualizacao();
12     glutPostRedisplay();
13 }
14
15 // Funcao callback para eventos de botoes do mouse
16 void GerenciaMouse(int button, int state, int x, int y)
17 {
18     if(state==GLUT_DOWN)
19     {
20         // Salva os par?metros atuais
21         x_ini = x;
22         y_ini = y;
23         obsX_ini = obsX;
24         obsY_ini = obsY;
25         obsZ_ini = obsZ;
26         rotX_ini = rotX;
27         rotY_ini = rotY;
28         bot = button;
29     }
30     else bot = -1;
31 }
```

Programação OpenGL

```
1  // Funcao callback para eventos de movimento do mouse
2  #define SENS_ROT 5.0
3  #define SENS_OBS 10.0
4  #define SENS_TRANSL 10.0
5  void GerenciaMovim(int x, int y)
6  {
7      if(bot==GLUT_LEFT_BUTTON) // Botao esquerdo
8      {
9          // Calcula diferencas
10         int deltax = x_ini - x;
11         int deltay = y_ini - y;
12         // E modifica ?ngulos
13         rotY = rotY_ini - deltax/SENS_ROT;
14         rotX = rotX_ini - deltay/SENS_ROT;
15     }
16     else if(bot==GLUT_RIGHT_BUTTON)// Botao direito
17     {
18         // Calcula diferenca
19         int deltaz = y_ini - y;
20         // E modifica distancia do observador
21         obsZ = obsZ_ini + deltaz/SENS_OBS;
22     }
23
24     else if(bot==GLUT_MIDDLE_BUTTON)// Botao do meio
25     {
26         // Calcula diferencas
27         int deltax = x_ini - x;
28         int deltay = y_ini - y;
29         // E modifica posicoes
30         obsX = obsX_ini + deltax/SENS_TRANSL;
31         obsY = obsY_ini - deltay/SENS_TRANSL;
32     }
33     PosicionaObservador();
34     glutPostRedisplay();
35 }
```

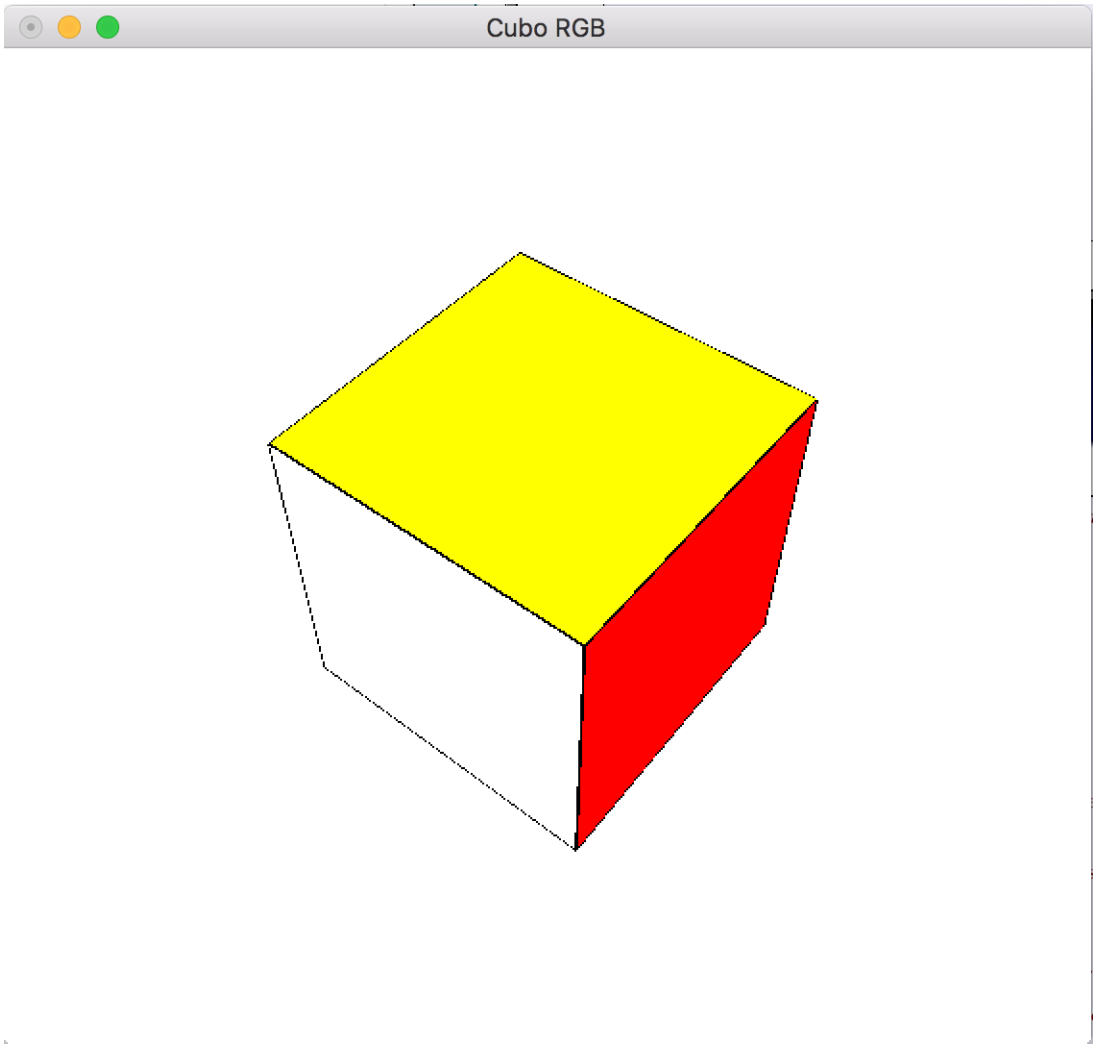
Programação OpenGL

```
1  // Funcao callback chamada quando o tamanho da janela eh alterado
2  void AlteraTamanhoJanela(GLsizei w, GLsizei h)
3  {
4      // Para previnir uma divisao por zero
5      if ( h == 0 ) h = 1;
6      // Especifica as dimensoes da viewport
7      glViewport(0, 0, w, h);
8      // Calcula a correcao de aspecto
9      fAspect = (GLfloat)w/(GLfloat)h;
10     EspecificaParametrosVisualizacao();
11 }
12
13 // Funcao responsavel por inicializar par?metros e variaveis
14 void Inicializa (void)
15 {
16     // Define a cor de fundo da janela de visualizacao como branca
17     glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
18     // Habilita o depth-buffering
19     glEnable(GL_DEPTH_TEST);
20     // Inicializa a variavel que especifica o angulo da projecao perspectiva
21     angle=45;
22     // Inicializa as variaveis usadas para alterar a posicao do observador virtual
23     rotX = 0;
24     rotY = 0;
25     obsX = obsY = 0;
26     obsZ = 250;
27 }
```

Programação OpenGL

```
1  // Programa Principal
2  int main(int argc, char **argv)
3  {
4      glutInit(&argc, argv);
5      // Define do modo de operacao da GLUT
6      glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
7      // Especifica a posicao inicial da janela GLUT
8      glutInitWindowPosition(5,5);
9      // Especifica o tamanho inicial em pixels da janela GLUT
10     glutInitWindowSize(450,450);
11     // Cria a janela passando como argumento o titulo da mesma
12     glutCreateWindow("Cubo RGB");
13     // Registra a funcao callback de redesenho da janela de visualizacao
14     glutDisplayFunc(Desenha);
15     // Registra a funcao callback de redimensionamento da janela de visualizacao
16     glutReshapeFunc(AlteraTamanhoJanela);
17     // Registra a funcao callback para tratamento das teclas normais
18     glutKeyboardFunc (Teclado);
19     // Registra a funcao callback para tratamento das teclas especiais
20     glutSpecialFunc (TeclasEspeciais);
21     // Registra a funcao callback para eventos de bot?es do mouse
22     glutMouseFunc(GerenciaMouse);
23     // Registra a funcao callback para eventos de movimento do mouse
24     glutMotionFunc(GerenciaMovim);
25     // Chama a funcao responsavel por fazer as inicializacoes
26     Inicializa();
27     // Inicia o processamento e aguarda interacoes do usuario
28     glutMainLoop();
29 }
```

Programação OpenGL



Bibliografia

- **Básica:**
  - Hearn, D. Baker, M. P. Computer Graphics with OpenGL, Prentice Hall, 2004. **(livro texto)**
  - Neider, J. Davis, T. Woo, M. OpenGL programming guide, 2007.
  - Angel, E. Interactive computer graphics: a top-down approach with OpenGL, Addison Wesley, 2000.
  - Foley, J. et. al. Introduction to Computer Graphics, Addison-Wesley, 1993.

Bibliografia

- **Complementar:**
  - Computer Graphics Comes of Age: An Interview with Andries van Dam. CACM, vol. 27, no. 7. 1982
  - The RenderMan – And the Oscar Goes to... IEEE Spectrum, vol. 38, no. 4, abril de 2001.
  - Material do ano passado: `https://sites.google.com/site/computacaograficaicmc2017t2/`
  - Apostilas antigas da disciplina Computação Gráfica
    - `http://www.gbdi.icmc.usp.br/material?q=system/files/apostilas.pdf`
  - Curso da ACM SIGGRAPH (on line)