

Transformações Geométricas 2D

SCC0250 - Computação Gráfica

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

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

12 de abril de 2010



Sumário

- 1 Introdução
- 2 Transformações Básicas
- 3 Coordenadas Homogêneas
- 4 Transformações Inversas
- 5 Transformações 2D Compostas
- 6 Outras Transformações 2D
- 7 Transformações entre Sistemas de Coordenadas 2D
- 8 Transformações 2D e OpenGL

Sumário

- 1 Introdução
- 2 Transformações Básicas
- 3 Coordenadas Homogêneas
- 4 Transformações Inversas
- 5 Transformações 2D Compostas
- 6 Outras Transformações 2D
- 7 Transformações entre Sistemas de Coordenadas 2D
- 8 Transformações 2D e OpenGL

Introdução

- **Transformações Geométricas** são operações aplicadas à descrição geométrica de um objeto para mudar sua
 - posição
 - orientação
 - tamanho

- Transformações geométricas básicas
 - translação
 - rotação
 - escala

- Outras: reflexão e cisalhamento

Sumário

- 1 Introdução
- 2 Transformações Básicas**
- 3 Coordenadas Homogêneas
- 4 Transformações Inversas
- 5 Transformações 2D Compostas
- 6 Outras Transformações 2D
- 7 Transformações entre Sistemas de Coordenadas 2D
- 8 Transformações 2D e OpenGL

Translação

Translação

- A translação consiste em adicionar *offsets* às coordenadas que definem um objeto

$$x' = x + t_x$$

$$y' = y + t_y$$

- Usando notação matricial, uma translação 2D pode ser descrita como

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

$$\mathbf{P} = \begin{bmatrix} x \\ y \end{bmatrix}, \quad \mathbf{P}' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

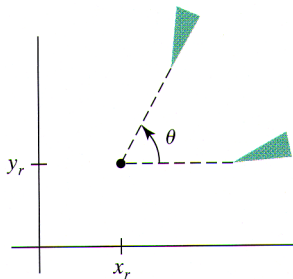
Rotação

Rotação

- Define-se uma transformação de rotação por meio de um **eixo de rotação** e um **ângulo de rotação**
- Em 2D a rotação se dá em um caminho circular no plano, rotacionando o objeto considerando-se um eixo perpendicular ao plano xy

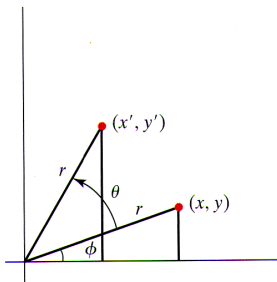
Rotação

- Parâmetros de rotação 2D são o ângulo θ de rotação e o ponto (x_r, y_r) de rotação, que é a intersecção do eixo de rotação com o plano xy
 - Se $\theta > 0$ a rotação é anti-horária
 - Se $\theta < 0$ a rotação é horária



Rotação

- Para simplificar considera-se que o ponto de rotação está na origem do sistema de coordenadas
 - O raio r é constante, ϕ é o ângulo original de $\mathbf{P} = (x, y)$ e θ é o ângulo de rotação



Rotação

- Sabendo que

$$\cos(\phi + \theta) = \frac{x'}{r} \Rightarrow x' = r \cos(\phi + \theta)$$

$$\text{sen}(\phi + \theta) = \frac{y'}{r} \Rightarrow y' = r \text{sen}(\phi + \theta)$$

- como

$$\cos(\alpha + \beta) = \cos \alpha \cdot \cos \beta - \text{sen} \alpha \cdot \text{sen} \beta$$

$$\text{sen}(\alpha + \beta) = \cos \alpha \cdot \text{sen} \beta + \text{sen} \alpha \cdot \cos \beta$$

- então

$$x' = r \cos \phi \cdot \cos \theta - r \text{sen} \phi \cdot \text{sen} \theta$$

$$y' = r \cos \phi \cdot \text{sen} \theta + r \text{sen} \phi \cdot \cos \theta$$

Rotação

- P pode ser descrito por meio de coordenadas polares

$$x = r \cos \phi, \quad y = r \sin \phi$$

- Então por substituição

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

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

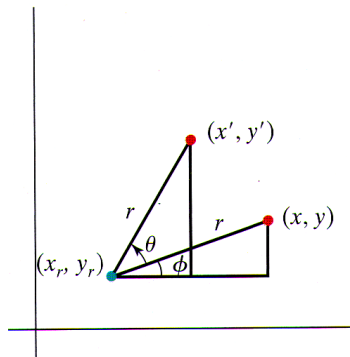
- Escrevendo na forma matricial temos

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

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

Rotação

- Rotação em torno de um ponto arbitrário (x_r, y_r)



Rotação

- Encontrando x'

$$\cos(\phi + \theta) = \frac{x' - x_r}{r}$$

$$x' = r \cos(\phi + \theta) + x_r$$

$$x' = x_r + r \cos \phi \cdot \cos \theta + r \sin \phi \cdot \sin \theta$$

- como

$$\cos \phi = \frac{x - x_r}{r}, \quad \sin \phi = \frac{y - y_r}{r}$$

- então

$$x' = x_r + (x - x_r) \cos \theta - (y - y_r) \sin \theta,$$

$$y' = y_r + (x - x_r) \sin \theta + (y - y_r) \cos \theta$$

Rotação

- A forma matricial pode ser conseguida criando-se um vetor coluna, mas existe uma forma melhor de se fazer isso que será apresentada mais adiante

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

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_r - x_r \cos \theta + y_r \sin \theta \\ y_r - x_r \sin \theta - y_r \cos \theta \end{bmatrix}$$

Transformação de Corpo Rígido

Transformação de Corpo Rígido

- A **rotação** e a **translação** é uma **Transformação de Corpo Rígido** pois direcionam ou movem um objeto sem deformá-lo
 - Mantém ângulos e distâncias entre as coordenadas do objeto

Escala

Escala

- Para alterar o tamanho de um objeto aplica-se o operador de escala
- Multiplica-se as coordenadas de um objeto por fatores de escala

$$x' = x \cdot s_x, \quad y' = y \cdot s_y$$

- Na forma matricial

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

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Escala

- Propriedades de s_x e s_y
 - s_x e s_y devem ser maiores que zero
 - Se $s_x > 1$ e $s_y > 1$ o objeto aumenta
 - Se $s_x < 1$ e $s_y < 1$ o objeto diminui
 - Se $s_x = s_y$ a escala é uniforme
 - Se $s_x \neq s_y$ a escala é diferencial

Escala

- Pela formulação definida, o objeto é escalado e movido

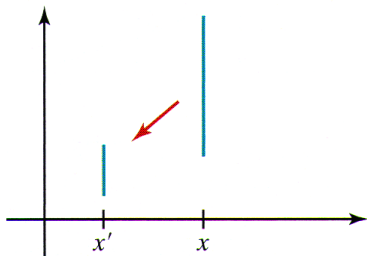


Figura: Escala de uma linha usando $s_x = s_y = 0.5$

Escala

- Para se manter a posição do objeto, escolhe-se uma posição fixa (x_f, y_f) , normalmente o centróide do objeto, e **escala-se a distância entre as coordenadas do objeto e esse ponto fixo**

$$x' - x_f = (x - x_f) \cdot s_x$$

$$y' - y_f = (y - y_f) \cdot s_y$$

- ou seja

$$x' = x \cdot s_x + x_f(1 - s_x)$$

$$y' = y \cdot s_y + y_f(1 - s_y)$$

Escala

- Na forma matricial podemos escrever adicionando um vetor coluna

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_f(1 - s_x) \\ y_f(1 - s_y) \end{bmatrix}$$

- $x_f(1 - s_x)$ e $y_f(1 - s_y)$ são constantes para todas as coordenadas do objeto

Sumário

- 1 Introdução
- 2 Transformações Básicas
- 3 Coordenadas Homogêneas**
- 4 Transformações Inversas
- 5 Transformações 2D Compostas
- 6 Outras Transformações 2D
- 7 Transformações entre Sistemas de Coordenadas 2D
- 8 Transformações 2D e OpenGL

Coordenadas Homogêneas

- As três transformações básicas podem ser expressas por

$$\mathbf{P}' = \mathbf{M}_1 \cdot \mathbf{P} + \mathbf{M}_2$$

- \mathbf{M}_1 : matriz 2×2 com fatores multiplicativos
- \mathbf{M}_2 : matriz coluna com termos para translação

- Para se aplicar uma sequencia de transformações, esse formato não ajuda
- Eliminando a adição de matrizes, uma sequencia de transformações pode ser definida por uma multiplicação de matrizes

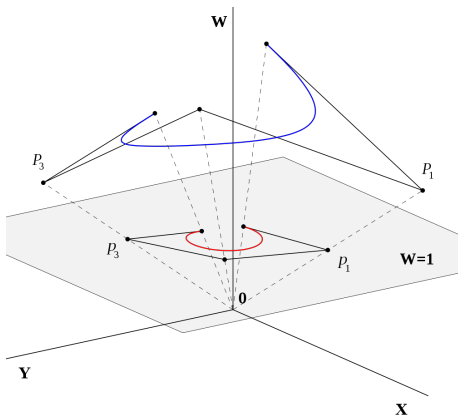
Coordenadas Homogêneas

- Isso pode ser feito expandindo-se para uma representação 3×3
- A terceira coluna é usada para os termos da translação

Coordenadas Homogêneas

- Uma forma de expansão é conhecida como coordenadas homogêneas
- Um ponto no espaço 2D representado em coordenadas homogêneas é descrito por 3 valores (x_h, y_h, h) , onde h é o parâmetro homogêneo ($h \neq 0$)
- Pode ser vista com a projeção de um ponto 3D no plano (Cartesiano) h

Coordenadas Homogêneas



Coordenadas Homogêneas

- A projeção do sistema homogêneo para o sistema Cartesiano se dá pela seguinte relação

$$x = \frac{x_h}{h}, \quad y = \frac{y_h}{h}$$

- h pode ser qualquer valor diferente de zero, mas por conveniência, escolhemos $h = 1$
- Então as coordenadas homogêneas (x_h, y_h, h) ficam $(x, y, 1)$
- Usando coordenadas homogêneas, as transformações são convertidas em multiplicações de matrizes

Coordenadas Homogêneas – Translação 2D

- A translação no espaço homogêneo é dada por

$$x'_h = 1 \cdot x_h + 0 \cdot y_h + t_x \cdot h$$

$$y'_h = 0 \cdot x_h + 1 \cdot y_h + t_y \cdot h$$

$$h = 0 \cdot x_h + 0 \cdot y_h + 1 \cdot h$$

Coordenadas Homogêneas – Translação 2D

- Definindo na forma matricial temos

$$\begin{bmatrix} x'_h \\ y'_h \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_h \\ y_h \\ h \end{bmatrix}$$

- Voltando ao espaço Cartesiano

$$x'_h/h = (1 \cdot x_h + 0 \cdot y_h + t_x \cdot h)/h \Rightarrow x' = x + t_x$$

$$y'_h/h = (0 \cdot x_h + 1 \cdot y_h + t_y \cdot h)/h \Rightarrow y' = y + t_y$$

$$h/h = (0 \cdot x_h + 0 \cdot y_h + 1 \cdot h)/h \Rightarrow 1 = 1$$

Coordenadas Homogêneas – Translação 2D

- Por conveniência, com $h = 1$, definimos a translação no espaço Cartesiano como

$$\mathbf{P}' = \mathbf{T}(t_x, t_y) \cdot \mathbf{P}$$

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

Coordenadas Homogêneas – Rotação 2D

- Uma rotação pode ser definida usando coordenadas homogêneas da seguinte forma

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

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

Coordenadas Homogêneas – Escala 2D

- Uma escala pode ser definida usando coordenadas homogêneas da seguinte forma

$$\mathbf{P}' = \mathbf{S}(s_x, s_y) \cdot \mathbf{P}$$

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

Sumário

- 1 Introdução
- 2 Transformações Básicas
- 3 Coordenadas Homogêneas
- 4 Transformações Inversas**
- 5 Transformações 2D Compostas
- 6 Outras Transformações 2D
- 7 Transformações entre Sistemas de Coordenadas 2D
- 8 Transformações 2D e OpenGL

Translação Inversa

- Para a translação, inverte-se o sinal das translações

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

Rotação Inversa

- Uma rotação inversa é obtida trocando o ângulo de rotação por seu negativo

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

- Isso rotaciona no sentido horário
- $\mathbf{R}^{-1} = \mathbf{R}^T$

Escala Inversa

- O inverso da escala é obtido trocando os parâmetros por seus inversos

$$\mathbf{S}^{-1}(s_x, s_y) = \begin{bmatrix} \frac{1}{s_x} & 0 & 1 \\ 0 & \frac{1}{s_y} & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Sumário

- 1 Introdução
- 2 Transformações Básicas
- 3 Coordenadas Homogêneas
- 4 Transformações Inversas
- 5 Transformações 2D Compostas**
- 6 Outras Transformações 2D
- 7 Transformações entre Sistemas de Coordenadas 2D
- 8 Transformações 2D e OpenGL

Introdução

- Usando representações matriciais homogêneas, uma sequencia de transformações pode ser representada como uma única matriz obtida a partir de multiplicações de matrizes de transformação

$$\begin{aligned}\mathbf{P}' &= \mathbf{M}_2 \cdot \mathbf{M}_1 \cdot \mathbf{P} \\ &= (\mathbf{M}_2 \cdot \mathbf{M}_1) \cdot \mathbf{P} \\ &= \mathbf{M} \cdot \mathbf{P}\end{aligned}$$

- A transformação é dada por \mathbf{M} ao invés de \mathbf{M}_1 e \mathbf{M}_2

Compondo Translações

- Para se compor duas translações podemos fazer

$$\begin{aligned} \mathbf{P}' &= \mathbf{T}(t_{2_x}, t_{2_y}) \cdot \{\mathbf{T}(t_{1_x}, t_{1_y}) \cdot \mathbf{P}\} \\ &= \{\mathbf{T}(t_{2_x}, t_{2_y}) \cdot \mathbf{T}(t_{1_x}, t_{1_y})\} \cdot \mathbf{P} \\ &= \mathbf{T}(t_{2_x} + t_{1_x}, t_{2_y} + t_{1_y}) \cdot \mathbf{P} \end{aligned}$$

$$\begin{bmatrix} 1 & 0 & t_{2_x} \\ 0 & 1 & t_{2_y} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_{1_x} \\ 0 & 1 & t_{1_y} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{1_x} + t_{2_x} \\ 0 & 1 & t_{1_y} + t_{2_y} \\ 0 & 0 & 1 \end{bmatrix}$$

Compondo Rotações

- Para se compor duas rotações podemos fazer

$$\begin{aligned} \mathbf{P}' &= \mathbf{R}(\theta_2) \cdot \{\mathbf{R}(\theta_1) \cdot \mathbf{P}\} \\ &= \{\mathbf{R}(\theta_2) \cdot \mathbf{R}(\theta_1)\} \cdot \mathbf{P} \\ &= \mathbf{R}(\theta_1 + \theta_2) \cdot \mathbf{P} \end{aligned}$$

$$\begin{bmatrix} \cos \theta_1 & -\text{sen} \theta_1 & 0 \\ \text{sen} \theta_1 & \cos \theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta_2 & -\text{sen} \theta_2 & 0 \\ \text{sen} \theta_2 & \cos \theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} \cos(\theta_1 + \theta_2) & -\text{sen}(\theta_1 + \theta_2) & 0 \\ \text{sen}(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Compondo Escalas

- Para se compor duas escalas podemos fazer

$$\begin{aligned}
 \mathbf{P}' &= \mathbf{S}(s_{2_x}, s_{2_y}) \cdot \{\mathbf{S}(s_{1_x}, s_{1_y}) \cdot \mathbf{P}\} \\
 &= \{\mathbf{S}(s_{2_x}, s_{2_y}) \cdot \mathbf{S}(s_{1_x}, s_{1_y})\} \cdot \mathbf{P} \\
 &= \mathbf{S}(s_{1_x} s_{2_x}, s_{1_y} s_{2_y}) \cdot \mathbf{P}
 \end{aligned}$$

$$\begin{bmatrix} s_{2_x} & 0 & 0 \\ 0 & s_{2_y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_{1_x} & 0 & 0 \\ 0 & s_{1_y} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{1_x} \cdot s_{2_x} & 0 & 0 \\ 0 & s_{1_y} \cdot s_{2_y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotação 2D com Ponto de Rotação

- Rotação com ponto de rotação é feita combinando-se múltiplas transformações
 - Movo o ponto de rotação para a origem
 - Executo a rotação
 - Movo o ponto de rotação para a posição inicial

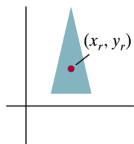
$$\mathbf{R}(x_r, y_r, \theta) = \mathbf{T}(x_r, y_r) \cdot \mathbf{R}(\theta) \cdot \mathbf{T}^{-1}(x_r, y_r)$$

$$\mathbf{R}(x_r, y_r, \theta) = \mathbf{T}(x_r, y_r) \cdot \mathbf{R}(\theta) \cdot \mathbf{T}(-x_r, -y_r)$$

Rotação 2D com Ponto de Rotação

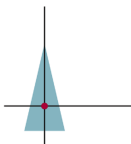
$$\begin{aligned} & \begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & -\operatorname{sen} \theta & 0 \\ \operatorname{sen} \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \cos \theta & -\operatorname{sen} \theta & x_r - x_r \cos \theta + y_r \operatorname{sen} \theta \\ \operatorname{sen} \theta & \cos \theta & y_r - y_r \cos \theta - x_r \operatorname{sen} \theta \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Rotação 2D com Ponto de Rotação



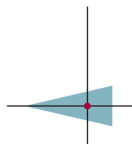
(a)

Original Position
of Object and
Pivot Point



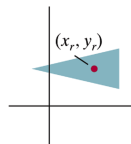
(b)

Translation of
Object so that
Pivot Point
 (x_r, y_r) is at
Origin



(c)

Rotation
about
Origin



(d)

Translation of
Object so that
the Pivot Point
is Returned
to Position
 (x_r, y_r)

Escala 2D com Ponto Fixo

- Escala com ponto fixo é feita combinando-se múltiplas transformações
 - Movo o ponto fixo para a origem
 - Executo a escala
 - Movo o ponto fixo para sua posição original

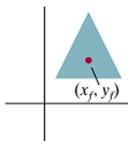
$$\mathbf{S}(x_f, y_f, s_x, s_y) = \mathbf{T}(x_f, y_f) \cdot \mathbf{S}(s_x, s_y) \cdot \mathbf{T}^{-1}(x_f, y_f)$$

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

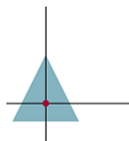
Escala 2D com Ponto Fixo

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

Escala 2D com Ponto Fixo



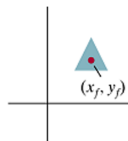
(a)
Original Position
of Object and
Fixed Point



(b)
Translate Object
so that Fixed Point
(x_f, y_f) is at Origin



(c)
Scale Object
with Respect
to Origin

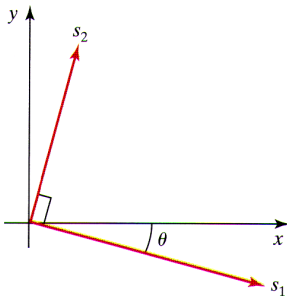


(d)
Translate Object
so that the Fixed
Point is Returned
to Position (x_f, y_f)

Escala 2D em Direções Gerais

- Os parâmetros s_x e s_y realizam a escala nas direções de x e y
- Para outras direções, rotaciona, escala e rotaciona de volta

$$\mathbf{S}(s_1, s_2, \theta) = \mathbf{R}^{-1}(\theta) \cdot \mathbf{S}(s_1, s_2) \cdot \mathbf{R}(\theta)$$



Escala 2D em Direções Gerais

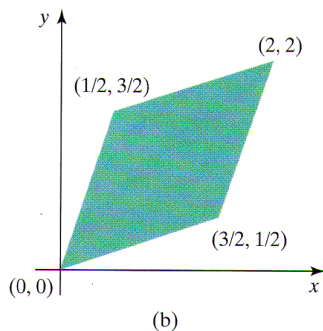
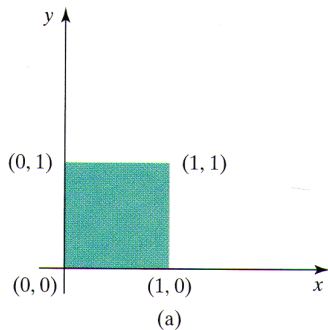


Figura: Transformação com $s_1 = 1$, $s_2 = 2$ e $\theta = 45^\circ$

Propriedade da Concatenação de Matrizes

- Multiplicação de matriz é associativa

$$\mathbf{M}_3 \cdot \mathbf{M}_2 \cdot \mathbf{M}_1 = (\mathbf{M}_3 \cdot \mathbf{M}_2) \cdot \mathbf{M}_1 = \mathbf{M}_3 \cdot (\mathbf{M}_2 \cdot \mathbf{M}_1)$$

- Multiplicação nos dois sentidos é possível, da esquerda para a direita e da direita para a esquerda
 - **Pré-multiplicação:** da esquerda para a direita – as transformação são especificadas na ordem em que são aplicadas ($\mathbf{M}_1 \rightarrow \mathbf{M}_2 \rightarrow \mathbf{M}_3$)
 - **Pós-multiplicação:** da direita para a esquerda – as transformação são especificadas na ordem reversa em que são aplicadas ($\mathbf{M}_3 \rightarrow \mathbf{M}_2 \rightarrow \mathbf{M}_1$)
 - OpenGL usa pós-multiplicação

Propriedade da Concatenação de Matrizes

- Multiplicação de matrizes podem não ser comutativas
 $M_2 \cdot M_1 \neq M_1 \cdot M_2$

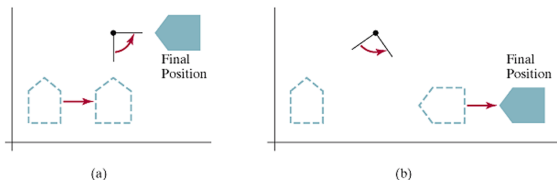


Figura: (a) primeiro o objeto é transladado depois rotacionado em 45^0 (b) primeiro o objeto é rotacionado em 45^0 , depois transladado.

Sumário

- 1 Introdução
- 2 Transformações Básicas
- 3 Coordenadas Homogêneas
- 4 Transformações Inversas
- 5 Transformações 2D Compostas
- 6 Outras Transformações 2D**
- 7 Transformações entre Sistemas de Coordenadas 2D
- 8 Transformações 2D e OpenGL

Reflexão

- Espelha-se as coordenadas de um objeto relativo a um eixo de reflexão, rotacionando em um ângulo de 180^0
- Reflexão em $y = 0$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Reflexão

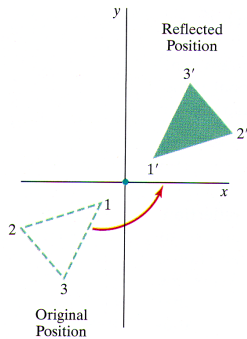
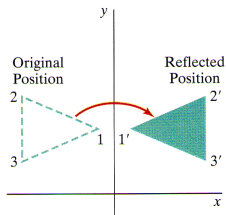
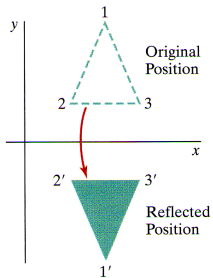
- Reflexão em $x = 0$

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Reflexão em $x = 0$ e $y = 0$

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Reflexão



Cisalhamento

- Distorce o formato do objeto na direção de x ou y
- Cisalhamento na direção de x

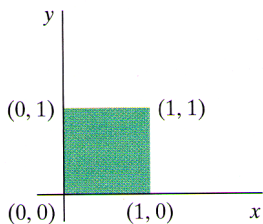
$$\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- O que transforma as coordenadas como

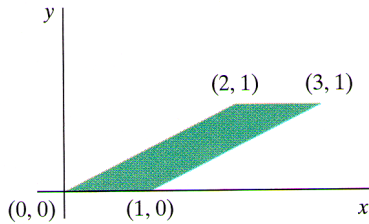
$$x' = x + sh_x \cdot y$$

$$y' = y$$

Cisalhamento



(a)



(b)

Figura: Convertendo um quadrado em um paralelogramo usando $sh_x = 2$.

Cisalhamento

- É possível gerar cisalhamentos na direção de x relativos a outras linhas de referência

$$\begin{bmatrix} 1 & sh_x & -sh_x \cdot y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Transformando

$$x' = x + sh_x(y - y_{ref})$$

$$y' = y$$

Cisalhamento

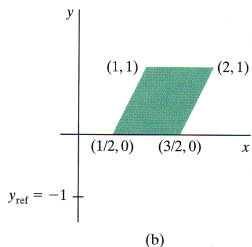
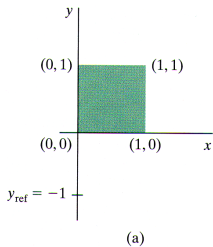


Figura: Exemplo de cisalhamento com $sh_x = \frac{1}{2}$ e $y_{ref} = -1$

Cisalhamento

- O cisalhamento na direção de y relativo a linha $x = x_{ref}$ é dado por

$$\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & -sh_y \cdot x_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Transformando

$$y' = y + sh_y(x - x_{ref})$$

$$x' = x$$

- O que transforma as coordenadas verticalmente uma quantidade proporcional a distância da linha de referência $x = x_{ref}$

Cisalhamento

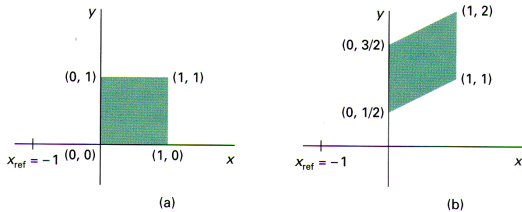


Figura: Exemplo de cisalhamento com $sh_y = \frac{1}{2}$ e $x_{ref} = -1$

Sumário

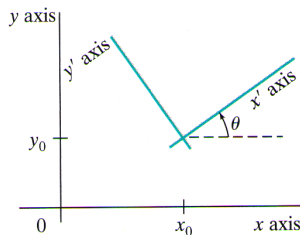
- 1 Introdução
- 2 Transformações Básicas
- 3 Coordenadas Homogêneas
- 4 Transformações Inversas
- 5 Transformações 2D Compostas
- 6 Outras Transformações 2D
- 7 Transformações entre Sistemas de Coordenadas 2D**
- 8 Transformações 2D e OpenGL

Transformações entre Sistemas de Coordenadas

- Aplicações de computação gráfica envolvem a transformação de um sistema de coordenadas em outro em vários estágios do processamento da cena

- Para se transformar um sistema de coordenadas em outro
 - Translado (x_0, y_0) para a origem $(0, 0)$
 - Rotaciono em $-\theta$

$$\mathbf{M}_{xy,x'y'} = \mathbf{R}(-\theta) \cdot \mathbf{T}(-x_0, -y_0)$$



Transformações entre Sistemas de Coordenadas

- Uma propriedade importante da matriz de transformação é que a sub-matriz de rotação 2×2 é ortonormal

$$\begin{bmatrix} r_{xx} & r_{xy} & tr_x \\ r_{yx} & r_{yy} & tr_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Isto é, cada linha (ou coluna) (r_{xx}, r_{xy}) e (r_{yx}, r_{yy}) formam um conjunto de vetores unitários ortogonais (ortonormais)

$$r_{xx}^2 + r_{xy}^2 = r_{yx}^2 + r_{yy}^2 = 1$$

$$r_{xx} \cdot r_{yx} + r_{xy} \cdot r_{yy} = 0$$

Transformações entre Sistemas de Coordenadas

- Isso é facilmente verificado porque

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

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

- e

$$(\cos \theta \times \operatorname{sen} \theta) + (-\operatorname{sen} \theta \times \cos \theta) = 0$$

Transformações entre Sistemas de Coordenadas

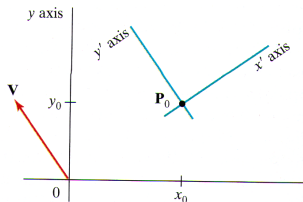
- Assim, se esses vetores forem transformados pela submatriz de rotação temos
 - (r_{xx}, r_{xy}) é transformado em um vetor unitário ao longo do eixo- x
 - (r_{yx}, r_{yy}) é transformado em um vetor unitário ao longo do eixo- y

$$\begin{bmatrix} r_{xx} & r_{xy} & tr_x \\ r_{yx} & r_{yy} & tr_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{xx} \\ r_{xy} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} r_{xx} & r_{xy} & tr_x \\ r_{yx} & r_{yy} & tr_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{yx} \\ r_{yy} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

Transformações entre Sistemas de Coordenadas

- Usando essa propriedade, outro método para fazer a transformação de um sistema de coordenadas em outro pode ser derivado
 - Para isso, inicialmente descrevemos a orientação do sistema de coordenadas $x'y'$ por meio de um vetor \mathbf{V} indicando a direção positiva do eixo y



Transformações entre Sistemas de Coordenadas

- Então, podemos especificar \mathbf{V} como um ponto no sistema de coordenadas xy relativo a origem, descrito como um vetor unitário

$$\mathbf{v} = \frac{\mathbf{V}}{|\mathbf{V}|} = (v_x, v_y)$$

- e podemos obter o vetor unitário u ortogonal a v na direção do eixo x'

$$\mathbf{u} = (v_y, -v_x) = (u_x, u_y)$$

Transformações entre Sistemas de Coordenadas

- Como qualquer matriz de rotação pode ser expressa por um conjunto de vetores ortonormais, então podemos escrever a matriz de rotação que faz $x'y'$ coincidir com xy como

$$\begin{bmatrix} u_x & u_y & 0 \\ v_x & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Transformações entre Sistemas de Coordenadas

- Isso porque

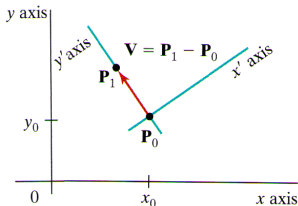
$$\begin{bmatrix} u_x & u_y & 0 \\ v_x & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u_x & u_y & 0 \\ v_x & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

Transformações entre Sistemas de Coordenadas

- É possível especificar \mathbf{V} relativo a um ponto \mathbf{P}_0 no sistema de coordenadas $x'y'$ ao invés de especificá-lo em relação a origem, para isso podemos fazer

$$\mathbf{v} = \frac{\mathbf{P}_1 - \mathbf{P}_0}{|\mathbf{P}_1 - \mathbf{P}_0|}$$



Sumário

- 1 Introdução
- 2 Transformações Básicas
- 3 Coordenadas Homogêneas
- 4 Transformações Inversas
- 5 Transformações 2D Compostas
- 6 Outras Transformações 2D
- 7 Transformações entre Sistemas de Coordenadas 2D
- 8 Transformações 2D e OpenGL

Exemplo de Transformações OPeNGL

```
1  #include <GL/glut.h>
2  #include <stdlib.h>
3
4  //armazena os vértices de um objeto
5  typedef struct VERTEX
6  {
7      int x;
8      int y;
9  };
10
11 //armazena a descrição geométrica de um objeto
12 typedef struct OBJECT
13 {
14     VERTEX *vertices;
15     int nrvertices;
16 };
17
18 OBJECT *object; //objeto global que será desenhado
```

Exemplo de Transformações OPeNGL

```
1 OBJECT *create_object()
2 {
3     OBJECT *obj = (OBJECT *)malloc(sizeof(OBJECT));
4     obj->nrvertices = 5;
5     obj->vertices = (VERTEX *)malloc(obj->nrvertices*sizeof(VERTEX));
6     obj->vertices[0].x = 110;
7     obj->vertices[0].y = 50;
8     obj->vertices[1].x = 110;
9     obj->vertices[1].y = 70;
10    obj->vertices[2].x = 100;
11    obj->vertices[2].y = 80;
12    obj->vertices[3].x = 90;
13    obj->vertices[3].y = 70;
14    obj->vertices[4].x = 90;
15    obj->vertices[4].y = 50;
16    return obj;
17 }
```


Exemplo de Transformações OpenGL

```
1 VERTEX calculate_centroid(OBJECT *obj)
2 {
3     int i;
4
5     VERTEX cent;
6     cent.x = 0;
7     cent.y = 0;
8
9     for (i=0; i < obj->nrvertices; i++)
10    {
11        cent.x += obj->vertices[i].x;
12        cent.y += obj->vertices[i].y;
13    }
14
15    cent.x /= obj->nrvertices;
16    cent.y /= obj->nrvertices;
17
18    return cent;
19 }
```

Exemplo de Transformações OpenGL

```
1 void init(void)
2 {
3     glClearColor(1.0, 1.0, 1.0, 0.0);
4     glMatrixMode(GL_PROJECTION);
5     glLoadIdentity();
6     gluOrtho2D(0.0, 200.0, 0.0, 150.0);
7
8     object = create_object(); //cria o objeto
9 }
10
11 void draw_object(OBJECT* obj)
12 {
13     int i;
14
15     glBegin(GL_POLYGON); //desenha uma linha
16     for (i=0; i < obj->nrvertices; i++)
17     {
18         glVertex2i(obj->vertices[i].x, obj->vertices[i].y);
19     }
20     glEnd();
21 }
22
23
24 void keyboard(unsigned char key, int x, int y)
25 {
26     if (key == 27)
27     {
28         if (object != NULL)
29         {
30             free(object->vertices); //elimina o objeto
31             free(object); //elimina o objeto
32             exit(1);
33         }
34     }
35 }
```

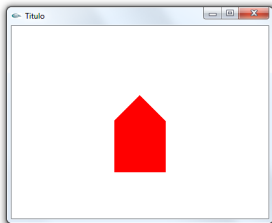
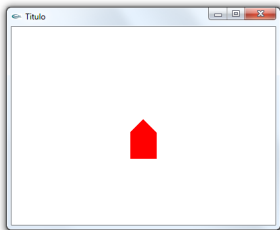
Exemplo de Transformações OpenGL

```
1 void draw(void)
2 {
3     glClear(GL_COLOR_BUFFER_BIT); //desenha o fundo (limpa a janela)
4     glColor3f(1.0, 0.0, 0.0); //altera o atributo de cor
5
6     glMatrixMode(GL_MODELVIEW); //garante que a matrix seja a ModelView
7     draw_object(object); //desenha o objeto
8
9     glFlush(); //processa as rotinas OpenGL o mais rápido possível
10 }
11
12 int main(int argc, char**argv)
13 {
14     glutInit(&argc, argv);
15     glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
16     glutInitWindowPosition(50, 100);
17     glutInitWindowSize(400, 300);
18     glutCreateWindow("Titulo");
19
20     init(); // inicialização (após a criação da janela)
21     glutDisplayFunc(draw); // registra a função de desenho
22     glutKeyboardFunc(keyboard);
23     glutMainLoop(); // desenha tudo e espera por eventos
24
25     return EXIT_SUCCESS;
26 }
```

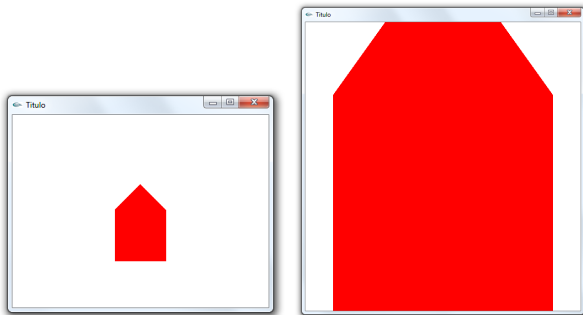
OpenGL – Pós-Multiplicação

```
1 void draw(void)
2 {
3     glClear(GL_COLOR_BUFFER_BIT); //desenha o fundo (limpa a janela)
4     glColor3f(1.0, 0.0, 0.0); //altera o atributo de cor
5
6     VERTEX cent = calculate_centroid(object); //calcula o centróide
7
8     glMatrixMode(GL_MODELVIEW); //garante que a matrix seja a ModelView
9     glTranslatef(cent.x, cent.y, 0); //movo o centróide para a posição original
10    glScalef(2, 2, 0); //faço a escala
11    glTranslatef(-cent.x, -cent.y, 0); //movo o centróide para a origem
12
13    draw_object(object);
14
15    glFlush(); //processa as rotinas OpenGL o mais rápido possível
16 }
```

OpenGL – Pós-Multiplicação



OpenGL – Cumulativo



- O método *draw(...)* é chamado mais de uma vez (modificação do tamanho da janela) – o objeto é escalado duas vezes

OpenGL – Cumulativo

Solução

- Carregar a matriz identidade (*glLoadIdentity()*)

```
1 void draw(void)
2 {
3     glClear(GL_COLOR_BUFFER_BIT); //desenha o fundo (limpa a janela)
4     glColor3f(1.0, 0.0, 0.0); //altera o atributo de cor
5
6     VERTEX cent = calculate_centroid(object); //calcula o centróide
7
8     glMatrixMode(GL_MODELVIEW); //garante que a matrix seja a ModelView
9     glLoadIdentity(); //carrega a matrix identidade
10    glTranslatef(cent.x, cent.y, 0); //movo o centróide para a posição original
11    glScalef(2, 2, 0); //faço a escala
12    glTranslatef(-cent.x, -cent.y, 0); //movo o centróide para a origem
13
14    draw_object(object);
15
16    glFlush(); //processa as rotinas OpenGL o mais rápido possível
17 }
```

OpenGL – Ordem de Transformações

Alterando a Ordem das Transformações

- Primeiro rotaciono, depois faço a translação

```
1 void draw(void)
2 {
3     glClear(GL_COLOR_BUFFER_BIT); //desenha o fundo (limpa a janela)
4     glColor3f(1.0, 0.0, 0.0); //altera o atributo de cor
5
6     VERTEX cent = calculate_centroid(object); //calcula o centróide
7
8     glMatrixMode(GL_MODELVIEW); //garante que a matrix seja a ModelView
9     glLoadIdentity(); //carrega a matrix identidade
10    glTranslatef(cent.x, cent.y, 0); //movo o centróide para a posição original
11    glRotatef(90, 0, 0, 1); //rotaciono
12    glTranslatef(-cent.x, -cent.y, 0); //movo o centróide para a origem
13    glTranslatef(50, 0, 0); //faço a translação
14
15    draw_object(object);
16
17    glFlush(); //processa as rotinas OpenGL o mais rápido possível
18 }
```

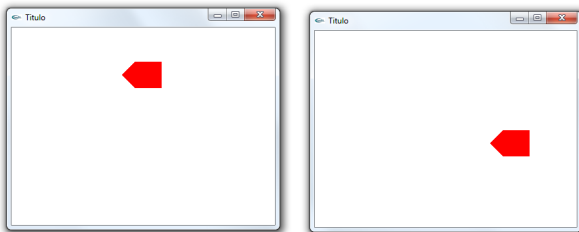

OpenGL – Ordem de Transformações

Alterando a Ordem das Transformações

- Primeiro faço a translação, depois rotaciono

```
1 void draw(void)
2 {
3     glClear(GL_COLOR_BUFFER_BIT); //desenha o fundo (limpa a janela)
4     glColor3f(1.0, 0.0, 0.0); //altera o atributo de cor
5
6     VERTEX cent = calculate_centroid(object); //calcula o centróide
7
8     glMatrixMode(GL_MODELVIEW); //garante que a matrix seja a ModelView
9     glLoadIdentity(); //carrega a matrix identidade
10    glTranslatef(50, 0, 0); //faço a translação
11    glTranslatef(cent.x, cent.y, 0); //movo o centróide para a posição original
12    glRotatef(90, 0, 0, 1); //rotaciono
13    glTranslatef(-cent.x, -cent.y, 0); //movo o centróide para a origem
14
15    draw_object(object);
16
17    glFlush(); //processa as rotinas OpenGL o mais rápido possível
18 }
```

OpenGL – Ordem de Transformações



- A ordem das transformações leva a resultados completamente diferentes