Transformações Geométricas 2D

SCC0250 - Computação Gráfica

Prof. Rosane Minghim
https://sites.google.com/site/computacaograficaicmc2017t2/
rminghim@icmc.usp.br
P.A.E. Nícolas Roque *nrsantos@usp.br*

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

19 de março de 2017

Sumário

Sumário

Introdução

- **Transformações Geométricas** são operações aplicadas à descrição geométrica de um objeto para mudar sua
  - posição (translação)
  - orientação (rotação)
  - tamanho (escala)

- Além dessas **transformações básicas**, existem outras
  - reflexão
  - cisalhamento

Sumário

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

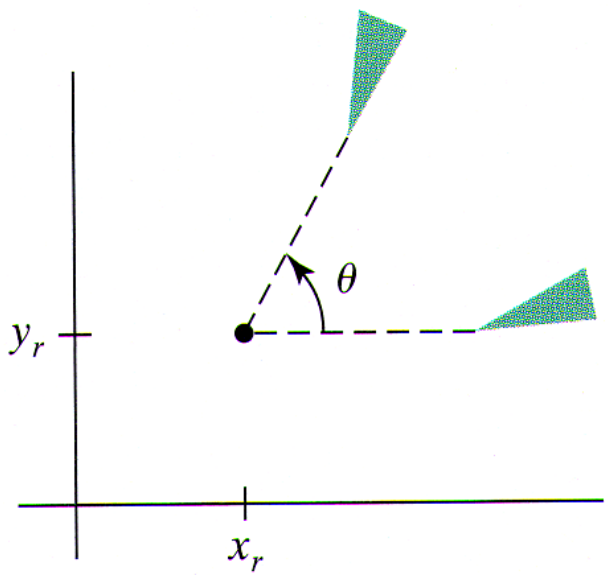
Rotação

- 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

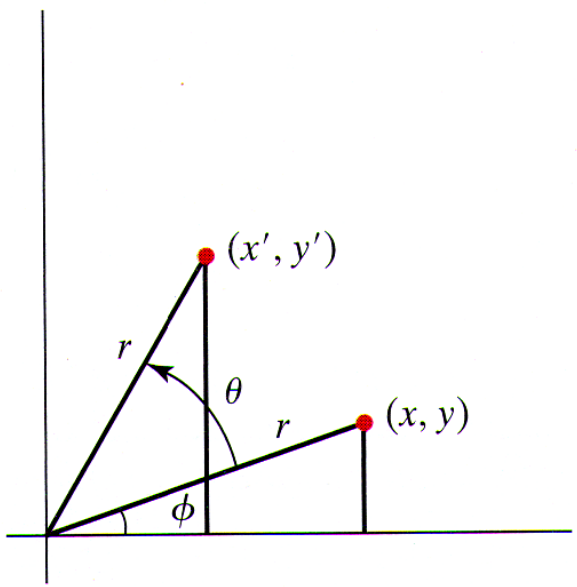
Rotação

- 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

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

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

- 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

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

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

Rotação

- $\mathbf{P} = (x, y)$ pode ser descrito por meio de coordenadas polares

$$x = r \cdot \cos\phi, \qquad y = r \cdot \operatorname{sen}\phi$$

- Então por substituição

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

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

- Escrevendo na forma matricial temos

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

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

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

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

- 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

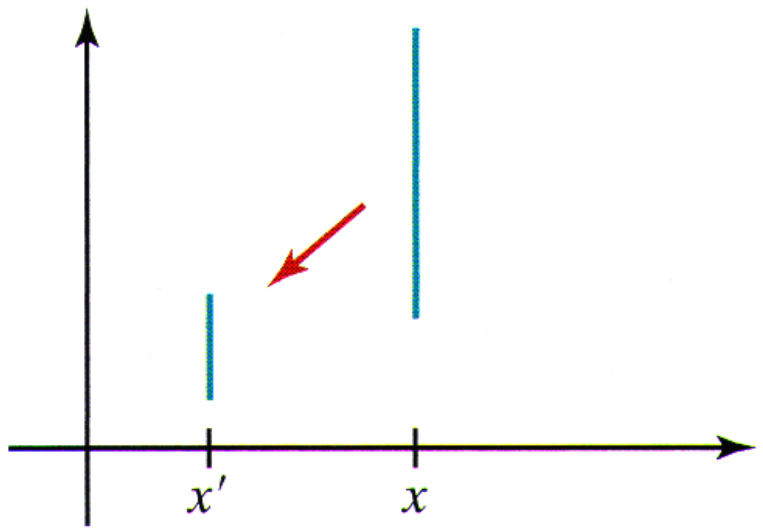


Figura: Escala de uma linha usando $s_x = s_y = 0.5$

Sumário

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
  - Eliminar a adição de matrizes permite escrever uma sequencia de transformações como uma multiplicação de matrizes

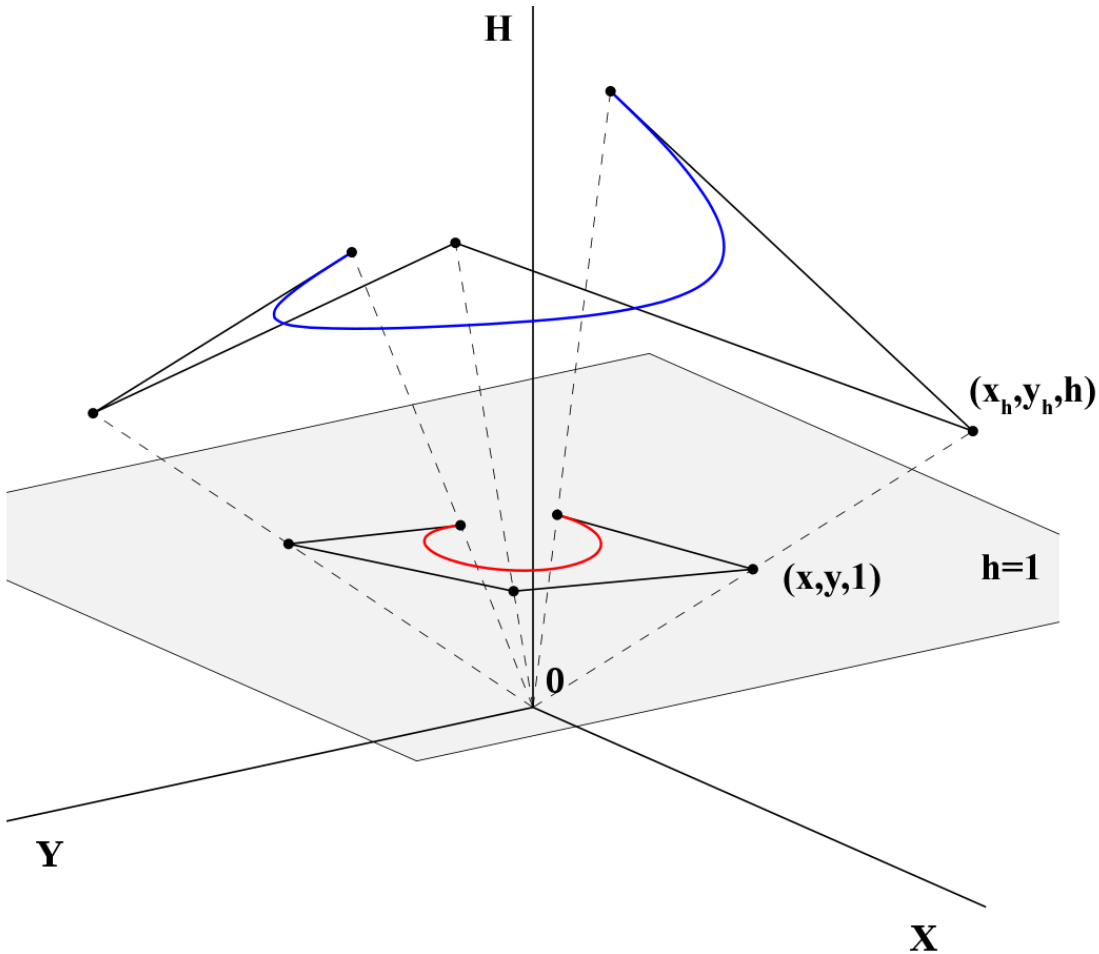
Coordenadas Homogêneas

- Isso pode ser feito expandindo-se o espaço **Cartesiano 2D** para o espaço de **Coordenadas Homogêneas 3D**

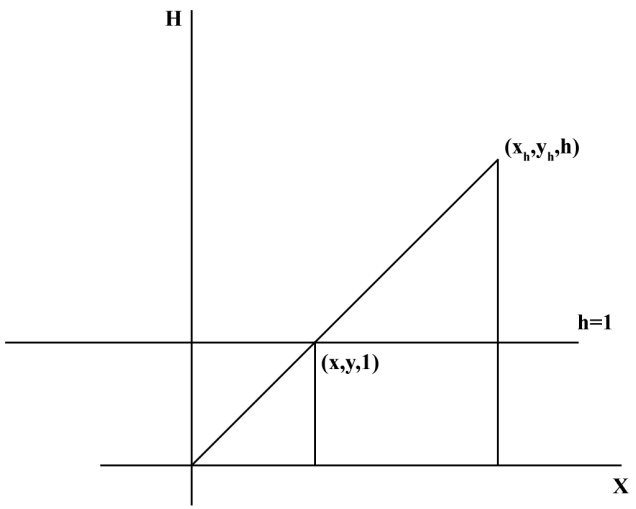
- Nessa expansão um ponto (x, y) é expandido para (x_h, y_h, h), onde h é o parâmetro homogêneo $(h \neq 0)$

- As coordenadas cartesianas são recuperados projetando as coordenadas homogêneas no plano $h = 1$

Coordenadas Homogêneas



Coordenadas Homogêneas



- Por semelhança de triângulos, a projeção do sistema homogêneo para o sistema Cartesiano se dá pela seguinte relação

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

- Nas coordenadas homogêneas, h pode ser qualquer valor diferente de zero, mas escolhemos $h = 1$ para a transformação ser mais simples

Coordenadas Homogêneas – Translação 2D

- Usando coordenadas homogêneas, as transformações são convertidas em multiplicações de matrizes

- 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

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

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

Coordenadas Homogêneas – Rotação 2D

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

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

$$\begin{bmatrix} x'_h \\ y'_h \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\operatorname{sen}\theta & 0 \\ \operatorname{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_h \\ y_h \\ 1 \end{bmatrix}$$

Coordenadas Homogêneas – Escala 2D

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

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

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

Transformando Vértices

- Exemplo de utilização de uma matriz de transformação

```
1  #version 150
2
3  in vec3 a_position;
4
5  void main(void)
6  {
7    //criando uma matriz de escala 2D
8    mat3 model = mat3(1.5, 0.0, 0.0, //primeira coluna
9                      0.0, 1.5, 0.0, //segunda coluna
10                     0.0, 0.0, 1.0); //terceira coluna
11
12   //multiplicando a matriz de transformacao pelo vetor
13   //em coordenadas homogeneas
14   vec3 pos = model * vec3(a_position[0], a_position[1], 1.0);
15
16   //convertendo as coordenadas homogeneas para euclideanas
17   gl_Position = vec4(pos[0]/pos[2], pos[1]/pos[2], 0.0, 1.0);
18 }
```

Sumário

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

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

- 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

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

$$\begin{aligned} \mathbf{P'_h} &= \mathbf{M_2} \cdot \mathbf{M_1} \cdot \mathbf{P_h} \\ &= (\mathbf{M_2} \cdot \mathbf{M_1}) \cdot \mathbf{P} \\ &= \mathbf{M} \cdot \mathbf{P_h} \end{aligned}$$

- 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

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

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

Compondo Rotações

- Para se compor duas rotações podemos fazer

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

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

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

Compondo Escalas

- Para se compor duas escalas podemos fazer

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

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

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

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

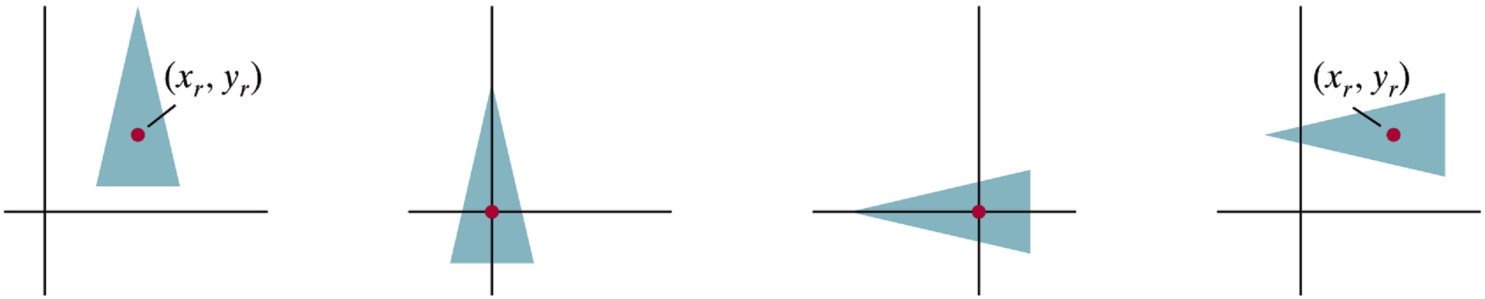
$$\mathbf{R}(x_r, y_r, \theta) = \mathbf{T}(x_r, y_r) \cdot \mathbf{R}(\theta) \cdot \mathbf{T}(-x_r, -y_r)$$

Rotação 2D com Ponto de Rotação

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

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

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

$$\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix}$$

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

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

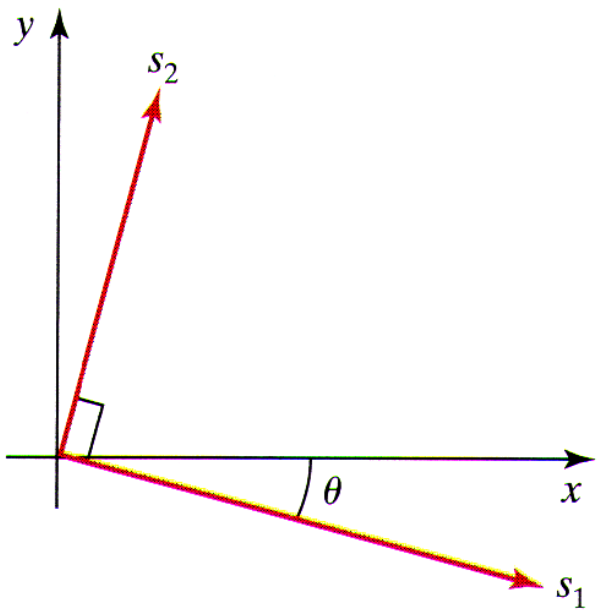
(c) Scale Object with Respect to Origin

(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

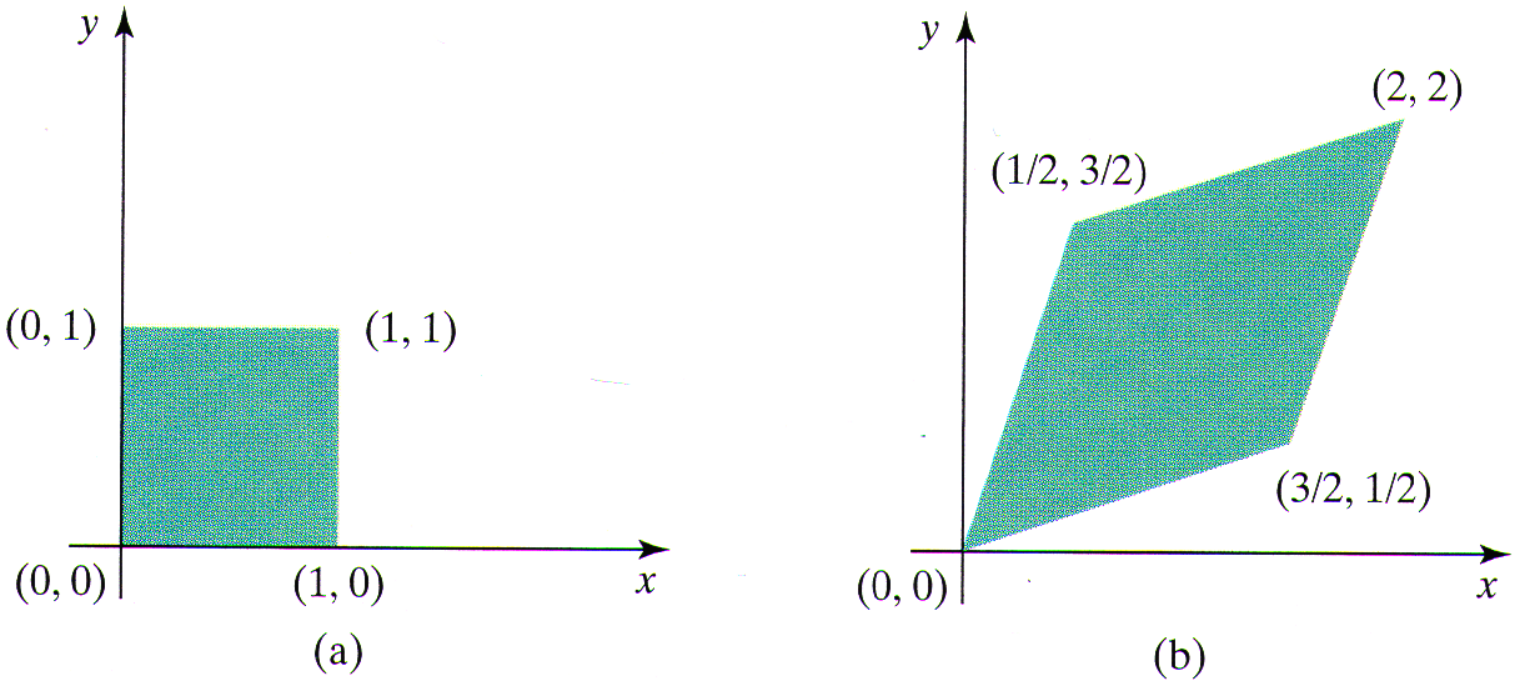


Figura: Transformação com $s_1 = 1$, $s_2 = 2$ e $\theta = 45^0$

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 inversa 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 não é comutativa $\mathbf{M_2} \cdot \mathbf{M_1} \neq \mathbf{M_1} \cdot \mathbf{M_2}$

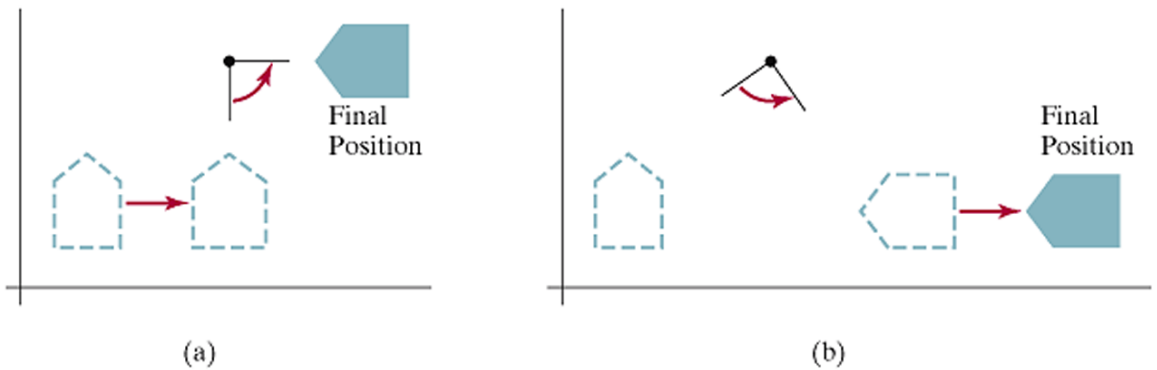


Figura: (a) primeiro o objeto é transladado depois rotacionado em 45^0 (b) primeiro o objeto é rotacionado em 45^0, depois transladado.

Sumário

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

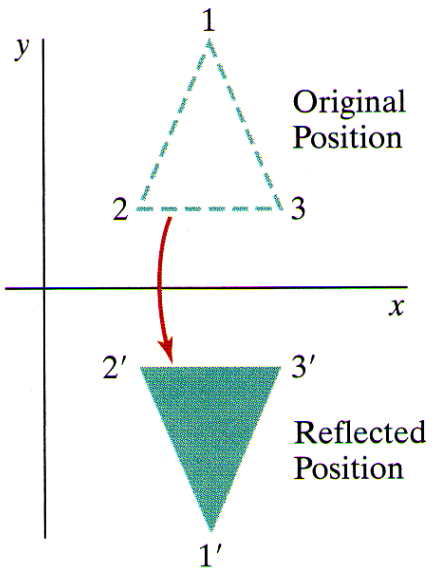
- Reflexão em $x = 0$

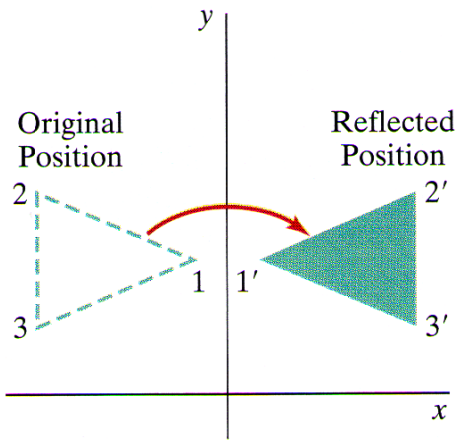
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

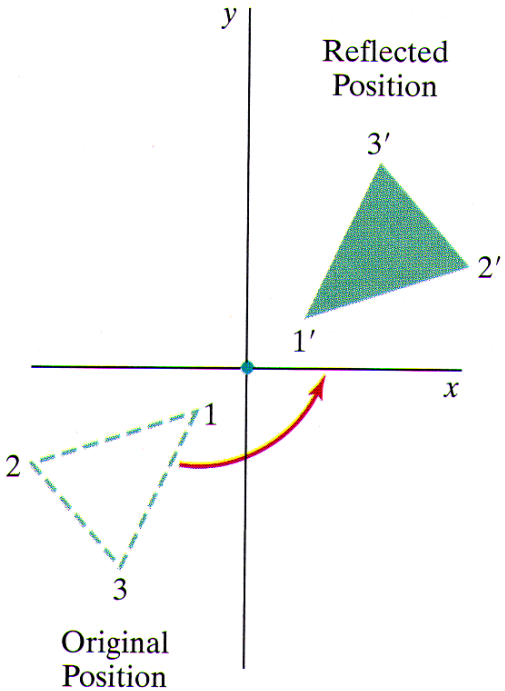
- Reflexão em $x = 0$ e $y = 0$

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

Reflexão


y
1
Original
Position
2
3
x
2′
3′
Reflected
Position
1′


y
Original
Position
Reflected
Position
2
2′
1
1′
3
3′
x


y
Reflected
Position
3′
2′
1′
x
1
2
3
Original
Position

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

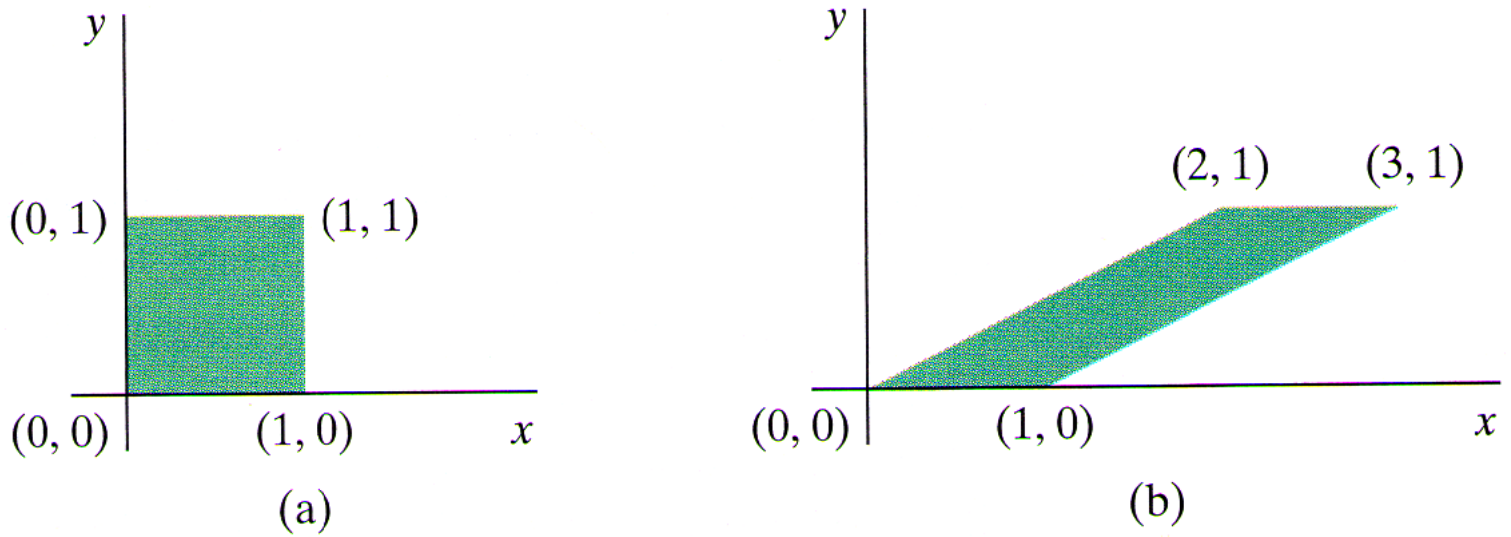


Figura: Convertendo um quadrado em um paralelogramo usando $sh_x = 2$.

Sumário

Exemplo de Transformações OpenGL

```
1  //armazena os vértices de um objeto
2  public class Vertice {
3
4    public Vertice(float x, float y) {
5      this.x = x;
6      this.y = y;
7    }
8
9    public float x;
10   public float y;
11 }
```

Exemplo de Transformações OpenGL

```
1  //armazena a descrição geométrica de um objeto
2  public abstract class Objeto {
3
4    public Objeto() {
5      vertices = new ArrayList<Vertice>();
6    }
7
8    public abstract void create();
9
10   public void draw(GL gl) {
11     gl.glBegin(GL.GL_POLYGON);
12     for (int i = 0; i < vertices.size(); i++) {
13       gl.glVertex2f(vertices.get(i).x, vertices.get(i).y);
14     }
15     gl.glEnd();
16   }
17
18   public Vertice getFixedPoint() {
19     if (vertices != null && !vertices.isEmpty()) {
20       return vertices.get(0);
21     }
22     return null;
23   }
24
25   protected ArrayList<Vertice> vertices;
26 }
```

Exemplo de Transformações OpenGL

```
1  public class Casa extends Objeto {
2
3    public void create() {
4      vertices.add(new Vertice(110, 50));
5      vertices.add(new Vertice(110, 70));
6      vertices.add(new Vertice(100, 80));
7      vertices.add(new Vertice(90, 70));
8      vertices.add(new Vertice(90, 50));
9    }
10 }
```

Exemplo de Transformações OpenGL

```
1  public class Renderer implements GLEventListener {
2
3    public Renderer() {
4      casa = new Casa();
5      casa.create();
6    }
7
8    public void init(GLAutoDrawable drawable) {
9      GL gl = drawable.getGL();
10     gl.glClearColor(1.0f, 1.0f, 1.0f, 0.0f); //define cor de fundo
11     gl.glMatrixMode(GL.GL_PROJECTION); //carrega a matriz de projeção
12     gl.glLoadIdentity(); //lê a matriz identidade
13
14     GLU glu = new GLU();
15     glu.gluOrtho2D(0, 200, 0, 150); //define projeção ortogonal 2D
16   }
17
18   public void reshape(GLAutoDrawable drawable, int x,
19                       int y, int width, int height) {
20   }
21
22   ...
23 }
```

Exemplo de Transformações OpenGL

```
1  public class Renderer implements GLEventListener {
2    ...
3
4    public void display(GLAutoDrawable drawable) {
5      GL gl = drawable.getGL();
6      gl.glClear(GL.GL_COLOR_BUFFER_BIT); //desenha o fundo (limpa a janela)
7      gl.glColor3f(1.0f, 0.0f, 0.0f); //altera o atributo de cor
8
9      gl.glMatrixMode(GL.GL_MODELVIEW); //carrega a matriz de modelo
10     casa.draw(gl);
11
12     gl.glFlush(); //processa as rotinas OpenGL o mais rápido possível
13   }
14
15   public void displayChanged(GLAutoDrawable drawable,
16                       boolean modeChanged, boolean deviceChanged) {
17   }
18
19   private Objeto casa;
20 }
```

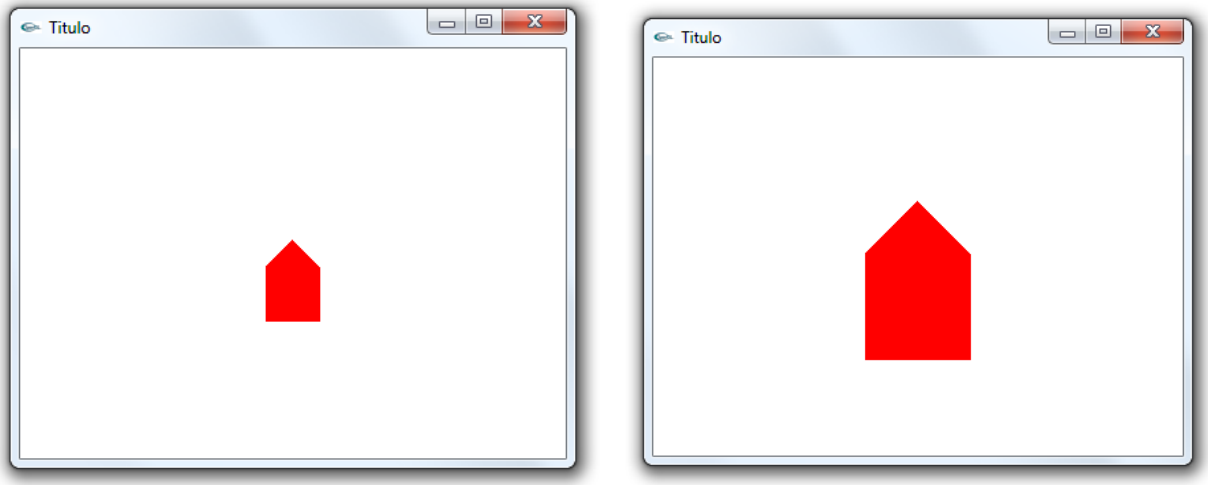
Exemplo de Transformações OpenGL

```
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    GLCanvas canvas = new GLCanvas(caps);
9    canvas.addGLEventListener(new Renderer());
10
11   //cria uma janela e adiciona o painel
12   JFrame frame = new JFrame("Aplicação JOGL Simples");
13   frame.getContentPane().add(canvas);
14   frame.setSize(400, 300);
15   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16
17   //inicializa o sistema e chama display() a 60 fps
18   Animator animator = new FPSAnimator(canvas, 60);
19   frame.setLocationRelativeTo(null);
20   frame.setVisible(true);
21   animator.start();
22 }
```

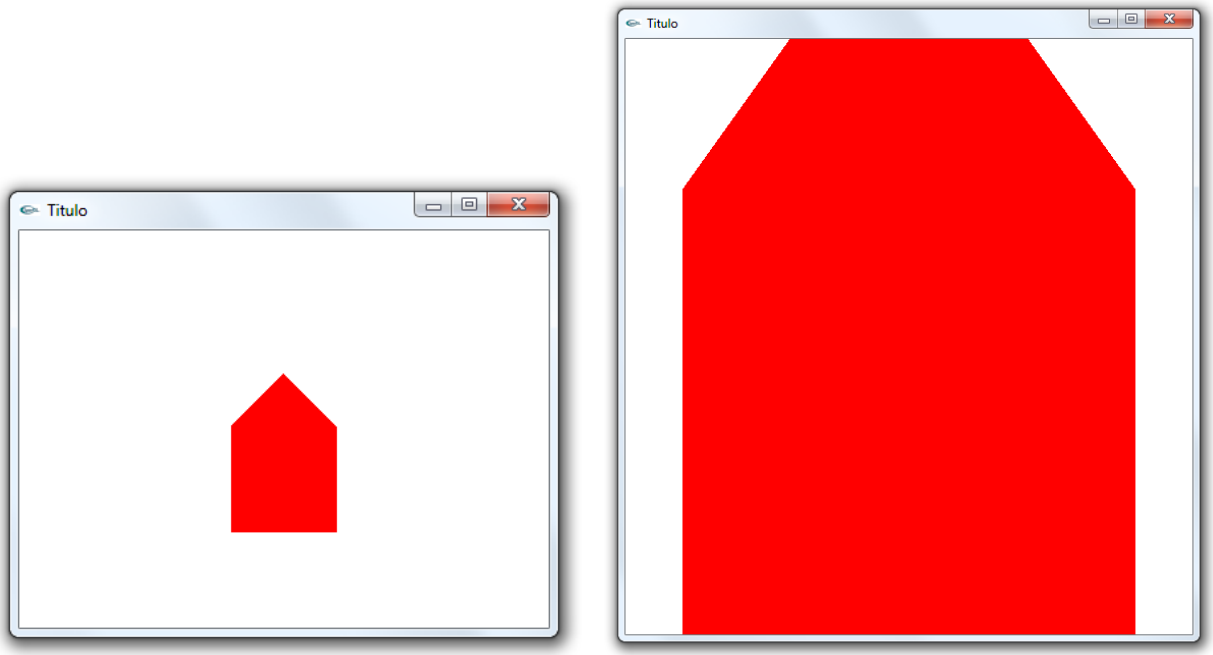
OpenGL – Pós-Multiplicação

```
1  public class Renderer implements GLEventListener {
2    ...
3
4    public void display(GLAutoDrawable drawable) {
5      GL gl = drawable.getGL();
6      gl.glClear(GL.GL_COLOR_BUFFER_BIT); //desenha o fundo (limpa a janela)
7      gl.glColor3f(1.0f, 0.0f, 0.0f); //altera o atributo de cor
8
9      Vertice v = casa.getFixedPoint(); //recuperando ponto fixo
10
11     gl.glMatrixMode(GL.GL_MODELVIEW); //carrega a matriz de modelo
12     gl.glTranslatef(v.x, v.y, 0); //move o point fixo para a posição original
13     gl.glScalef(2, 2, 0); //faz a escala
14     gl.glTranslatef(-v.x, -v.y, 0); //move o ponto fixo para a origem
15
16     casa.draw(gl);//desenho o objeto
17
18     gl.glFlush(); //processa as rotinas OpenGL o mais rápido possível
19   }
20 }
```

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  public class Renderer implements GLEventListener {
2    ...
3
4    public void display(GLAutoDrawable drawable) {
5      GL gl = drawable.getGL();
6      gl.glClear(GL.GL_COLOR_BUFFER_BIT); //desenha o fundo (limpa a janela)
7      gl.glColor3f(1.0f, 0.0f, 0.0f); //altera o atributo de cor
8
9      Vertice v = casa.getFixedPoint(); //recuperando ponto fixo
10
11     gl.glMatrixMode(GL.GL_MODELVIEW); //carrega a matriz de modelo
12     gl.glLoadIdentity(); //carrega a matrix identidade
13     gl.glTranslatef(v.x, v.y, 0); //move o point fixo para a posição original
14     gl.glScalef(2, 2, 0); //faz a escala
15     gl.glTranslatef(-v.x, -v.y, 0); //move o ponto fixo para a origem
16
17     casa.draw(gl);//desenho o objeto
18
19     gl.glFlush(); //processa as rotinas OpenGL o mais rápido possível
20   }
21 }
```

OpenGL – Ordem de Transformações

Alterando a Ordem das Transformações

- Primeiro rotaciono, depois faço a translação

```
1  public class Renderer implements GLEventListener {
2    ...
3
4    public void display(GLAutoDrawable drawable) {
5      GL gl = drawable.getGL();
6      gl.glClear(GL.GL_COLOR_BUFFER_BIT); //desenha o fundo (limpa a janela)
7      gl.glColor3f(1.0f, 0.0f, 0.0f); //altera o atributo de cor
8
9      Vertice v = casa.getFixedPoint(); //recuperando ponto fixo
10
11     gl.glMatrixMode(GL.GL_MODELVIEW); //carrega a matriz de modelo
12     gl.glLoadIdentity(); //carrega a matrix identidade
13     gl.glTranslatef(50, 0, 0); //faço a translação
14     gl.glTranslatef(v.x, v.y, 0); //move o point fixo para a posição original
15     gl.glRotatef(90, 0, 0, 1); //rotaciono
16     gl.glTranslatef(-v.x, -v.y, 0); //move o ponto fixo para a origem
17
18     casa.draw(gl);//desenho o objeto
19
20     gl.glFlush(); //processa as rotinas OpenGL o mais rápido possível
21   }
22 }
```

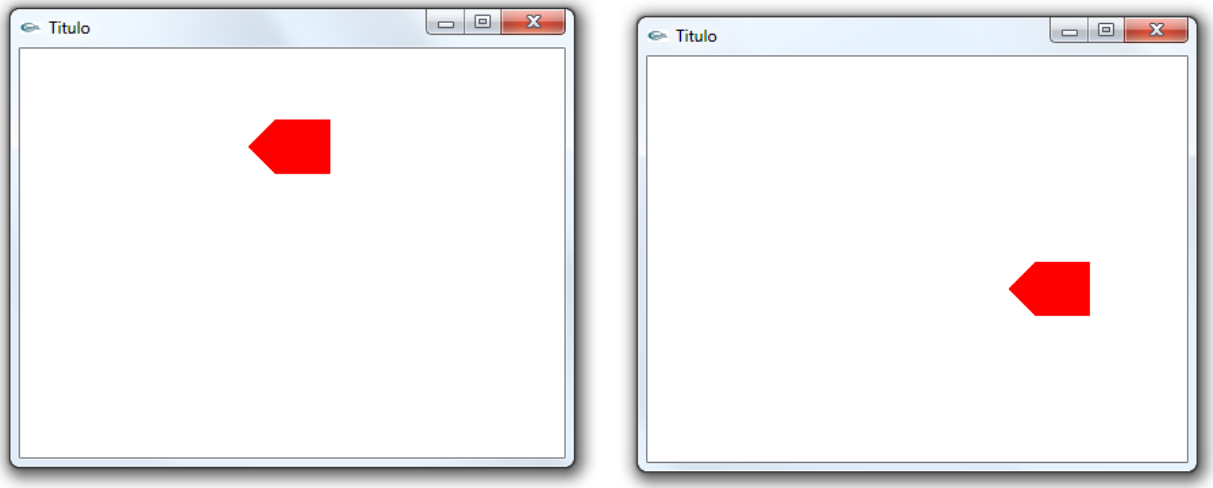
OpenGL – Ordem de Transformações

Alterando a Ordem das Transformações

- Primeiro faço a translação, depois rotaciono

```
1  public class Renderer implements GLEventListener {
2    ...
3
4    public void display(GLAutoDrawable drawable) {
5      GL gl = drawable.getGL();
6      gl.glClear(GL.GL_COLOR_BUFFER_BIT); //desenha o fundo (limpa a janela)
7      gl.glColor3f(1.0f, 0.0f, 0.0f); //altera o atributo de cor
8
9      Vertice v = casa.getFixedPoint(); //recuperando ponto fixo
10
11     gl.glMatrixMode(GL.GL_MODELVIEW); //carrega a matriz de modelo
12     gl.glLoadIdentity(); //carrega a matrix identidade
13     gl.glTranslatef(v.x, v.y, 0); //move o point fixo para a posição original
14     gl.glRotatef(90, 0, 0, 1); //rotaciono
15     gl.glTranslatef(-v.x, -v.y, 0); //move o ponto fixo para a origem
16     gl.glTranslatef(50, 0, 0); //faço a translação
17
18     casa.draw(gl);//desenho o objeto
19
20     gl.glFlush(); //processa as rotinas OpenGL o mais rápido possível
21   }
22 }
```

OpenGL – Ordem de Transformações



- A ordem das transformações leva a resultados completamente diferentes