Cap. 3: Transformações Geométricas

Ensino de Informática (3326) - 4º ano, 2º semestre
Engenharia Electrotécnica (2287) - 5º ano, 2º semestre
Engenharia Informática (2852) - 4º ano, 2º semestre

Sumário

- Motivação.
- Transformações métricas euclidianas: translação e rotação.
- Geometria métrica euclidiana.
- Coordenadas homogéneas.
- Transformações afins: translação, rotação, variação de tamanho e cisalhamento.
- Representação matricial de transformações afins.
- Composição de transformações geométricas 2D e 3D.
- Transformações afins em OpenGL.
- Operações com matrizes em OpenGL e transformações geométricas arbitrárias.
- Exemplos em OpenGL.

Motivação

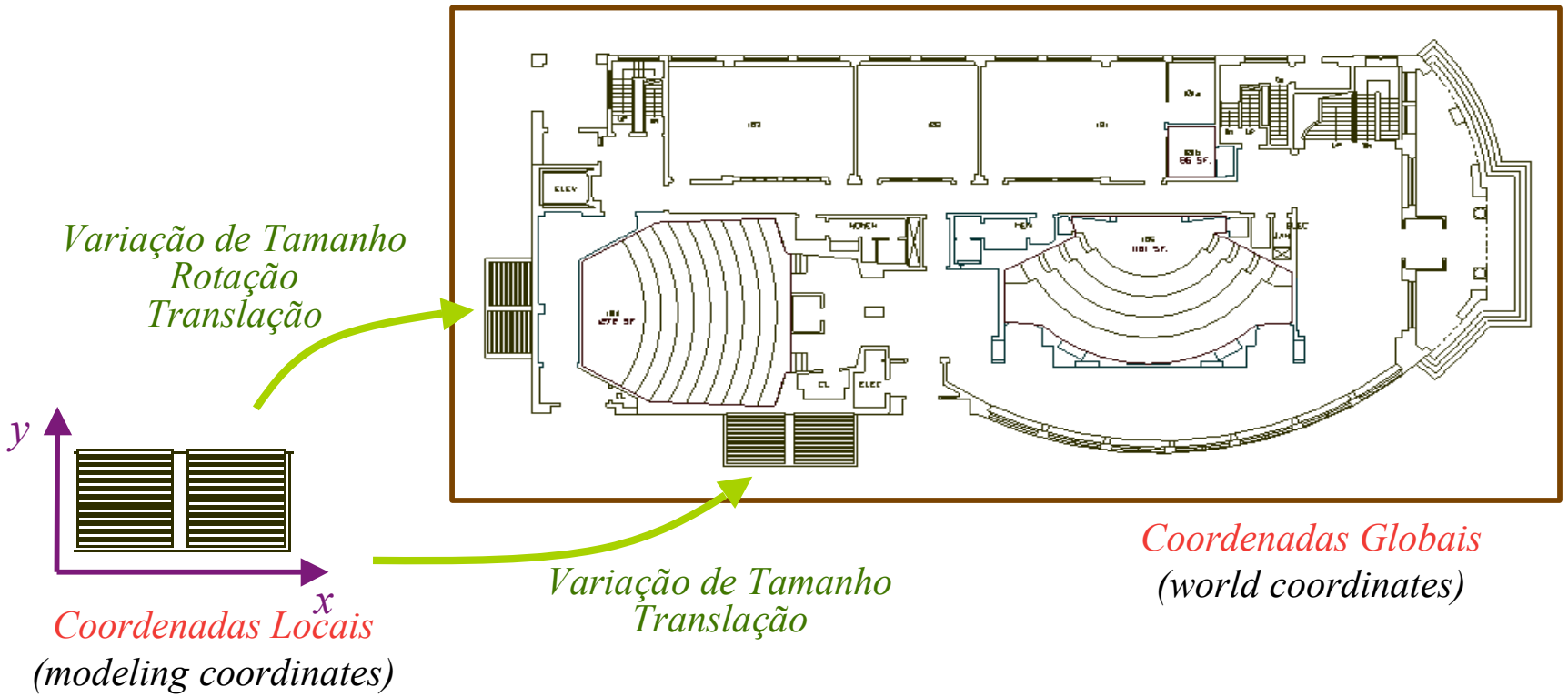
- Transformações geométricas
  - Translação, Rotação, Reflexão
  - Variação de Tamanho (scaling), Cisalhamento (shearing)
  - Projecção Ortogonal, Projecção Perspectiva

- Motivação – Porque é que as transformações geométricas são necessárias?
  - Como operações de **posicionamento** de objectos em 2D e 3D.
  - Como operações de **modelação** de objectos em 2D e 3D.
  - Como operações de **visualização** em 2D e 3D.

Motivação (cont.): modelação de objectos em 2D

- Transformações geométricas podem especificar operações de modelação de objectos
  - Permitem a definição dum objecto no seu próprio sistema de coordenadas locais (modeling coordinates ou coordenadas de modelação)
  - Permite usar a definição dum objecto várias vezes numa cena com um sistema de coordenadas globais (world coordinates ou coordenadas do domínio da cena)
    - A OpenGL fornece uma pilha de transformações que permite a sua utilização frequente

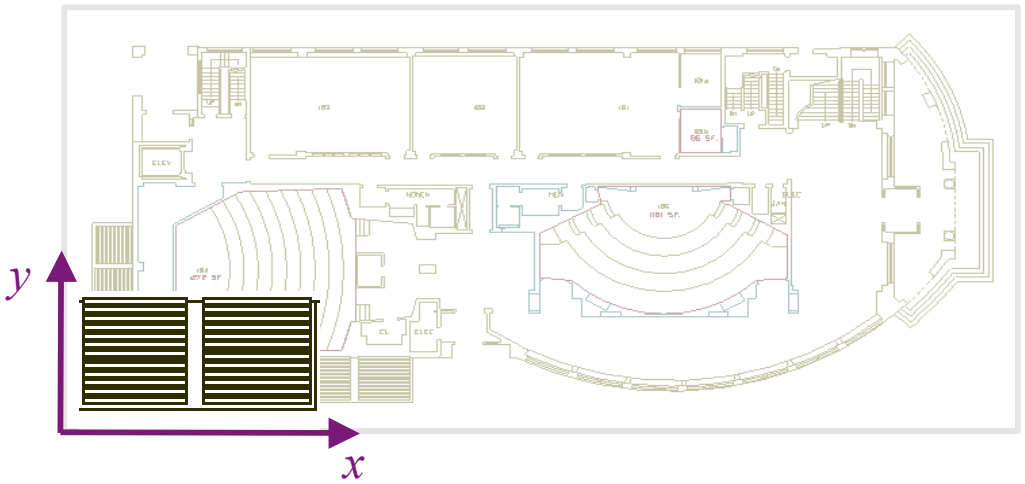
Motivação (cont.): modelação de objectos em 2D



Motivação (cont.): modelação de objectos em 2D

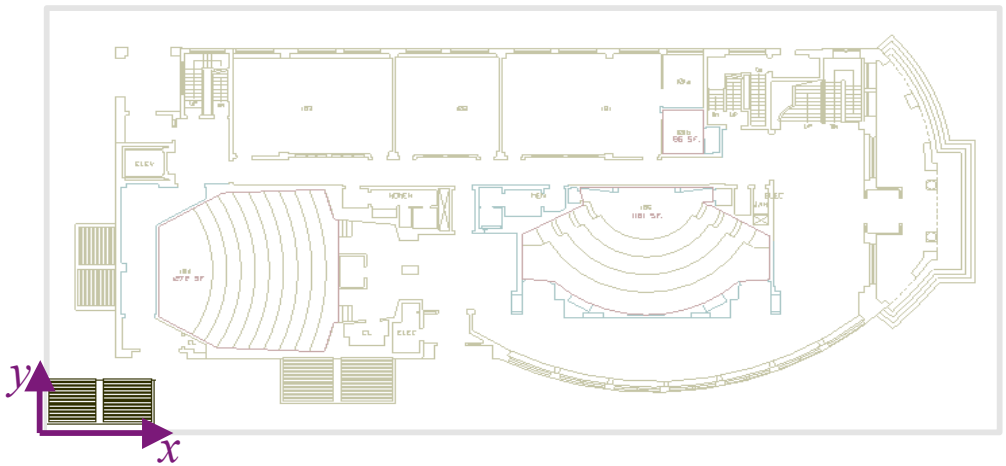
Coordenadas Locais

Posicionamento



Coordenadas Globais

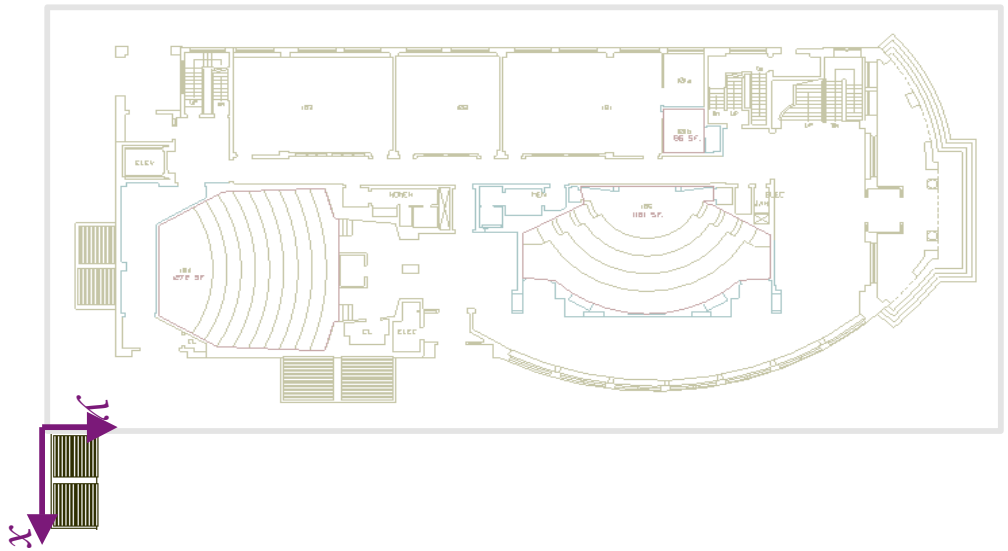
Variação de Tamanho



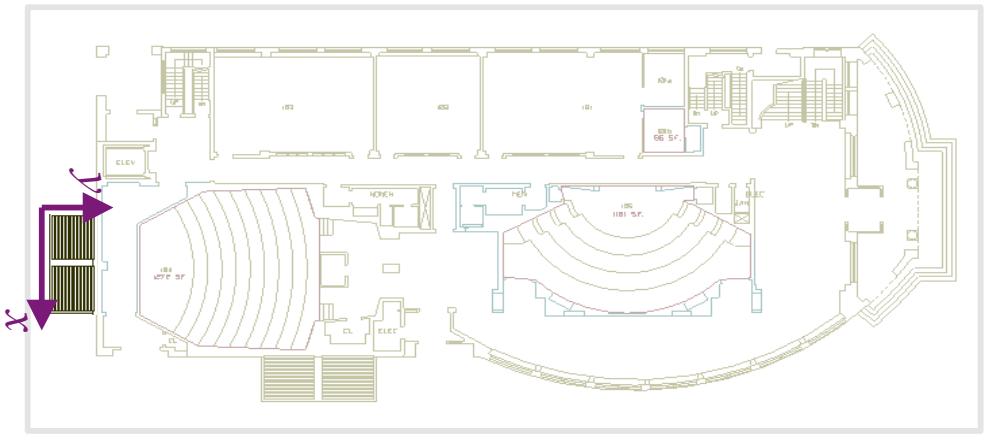
Motivação (cont.): modelação de objectos em 2D

Coordenadas Locais

Rotação



Translação

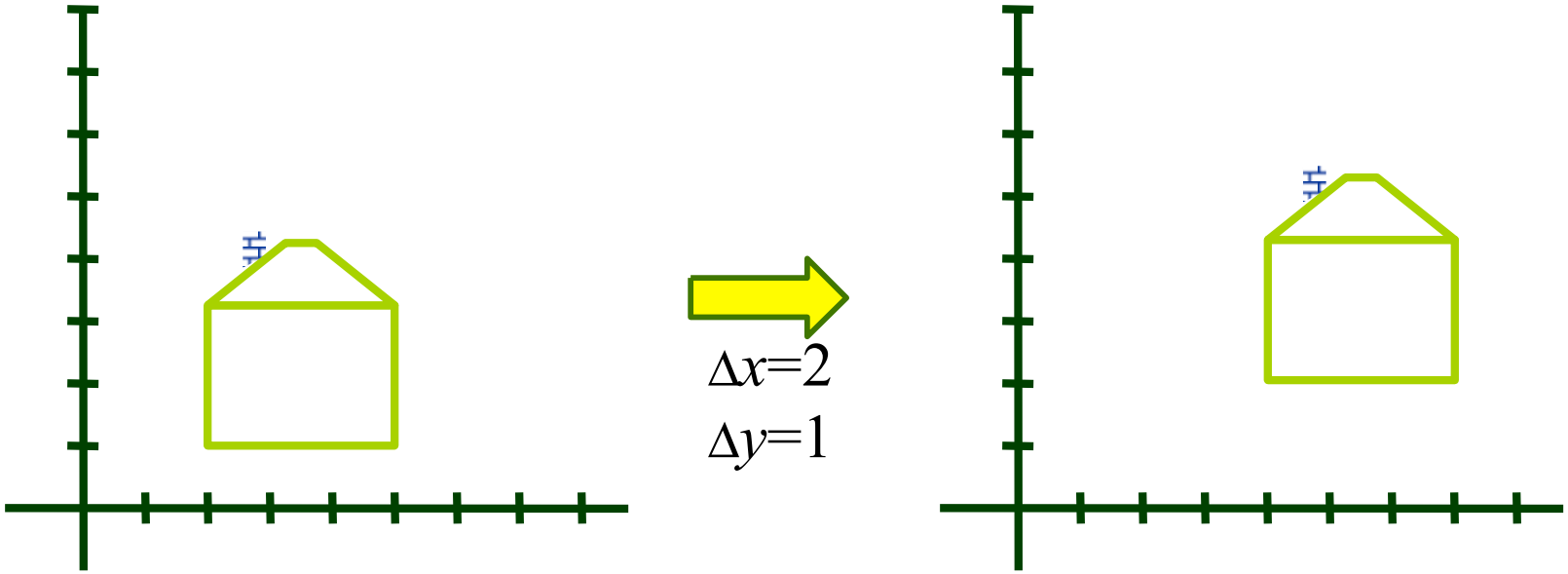


Coordenadas Globais

Translação 2D

$$\begin{cases} x' = x + \Delta x \\ y' = y + \Delta y \end{cases}$$

Transladar um ponto (x, y) significa deslocá-lo de uma quantidade de movimento linear $(\Delta x, \Delta y)$.



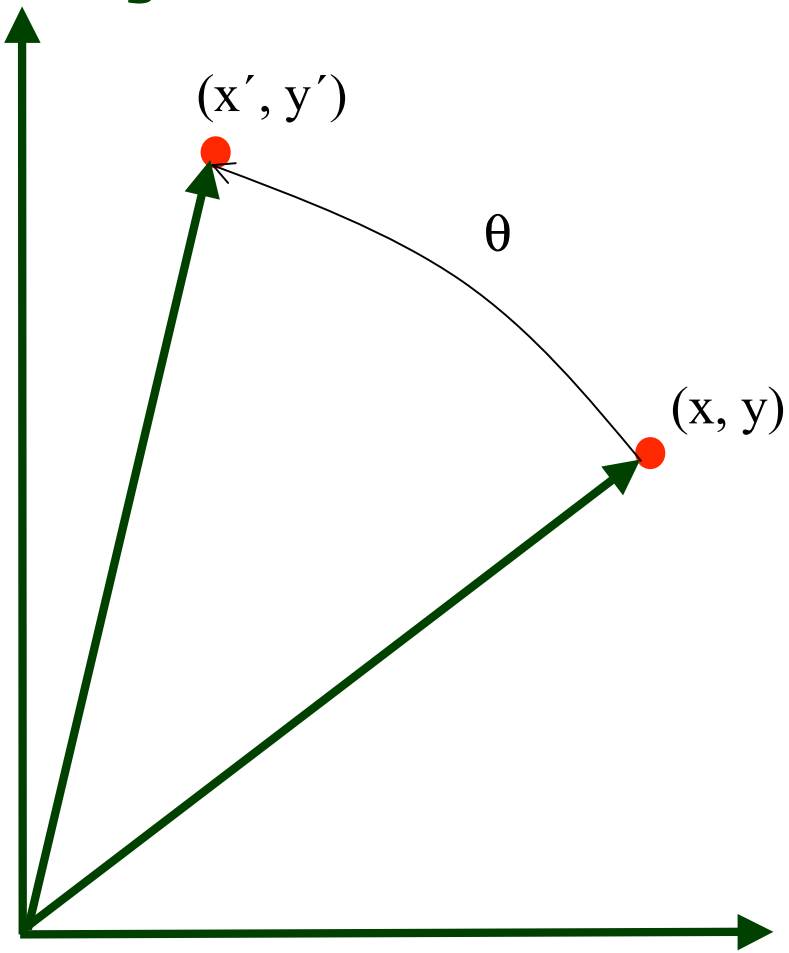
http://encyclopedia.laborlawtalk.com/Linear_combination

Translação 2D: forma matricial

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

- x’ não é uma combinação linear de x e y
- y’ não é uma combinação linear de x e y

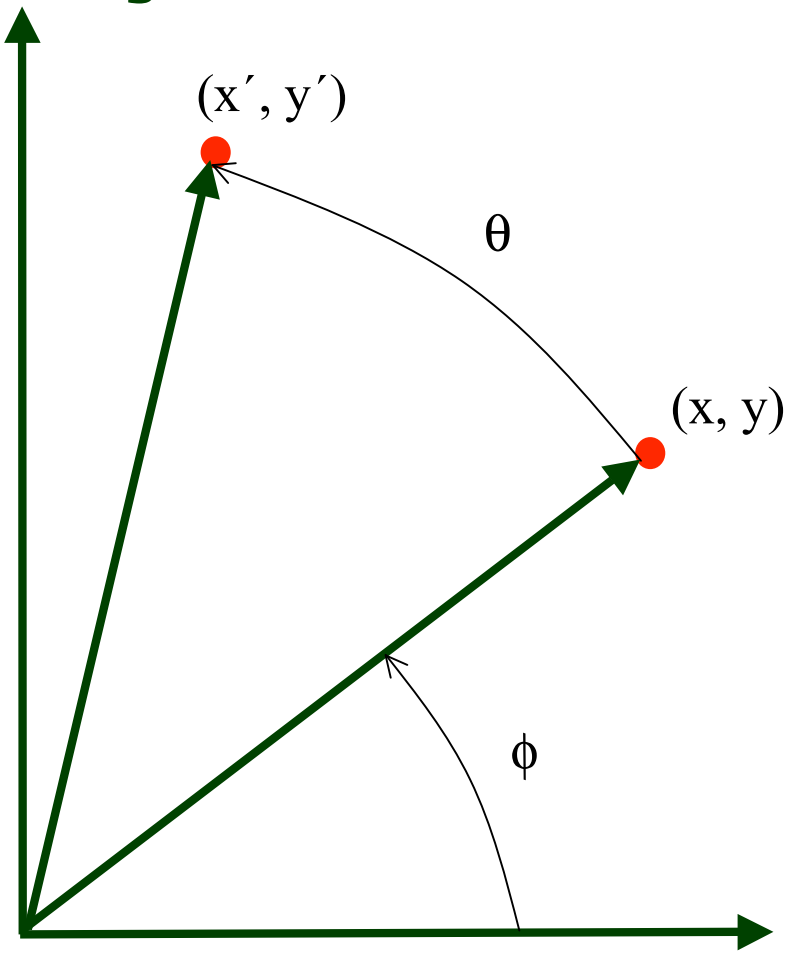
Rotação 2D



$$\begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \end{cases}$$

Rodar um ponto P=(x,y) de um ângulo θ relativamente à origem significa encontrar outro ponto Q=(x´,y´) sobre uma circunferência centrada na origem que passa pelos dois pontos, com θ=∠POQ.

Rotação 2D: cálculo de equações



$$\begin{cases} x = r\cos\phi \\ y = r\sin\phi \end{cases} \qquad \begin{cases} x' = r\cos(\phi+\theta) \\ y' = r\sin(\phi+\theta) \end{cases}$$

Desenvolvendo as expressões de x´ e y´, tem-se:

$$\begin{cases} x' = r\cos\phi\cos\theta - r\sin\phi\sin\theta \\ y' = r\cos\phi\sin\theta + r\sin\phi\cos\theta \end{cases}$$

Substituindo r cos(ϕ) e r sin(ϕ) por x e y nas equações anteriores tem-se:

$$\begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \end{cases}$$

Rotação 2D: forma matricial

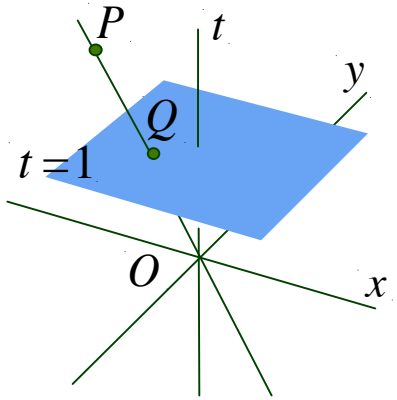
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Embora sin(θ) e cos(θ) sejam funções não-lineares de θ,
  - x' é uma combinação linear de x e y
  - y' é uma combinação linear de x e y

http://www.geom.uiuc.edu/docs/reference/CRC-formulas/node6.html

Coordenadas homogéneas

- Um triplo (x,y,t) de números reais com $t \neq 0$, é um conjunto de coordenadas homogéneas para o ponto P com coordenadas cartesianas (x/t,y/t).
- Portanto, o mesmo ponto tem muitos conjuntos de coordenadas homogéneas. Assim, (x,y,t) e (x',y',t') representam o mesmo ponto sse existe algum escalar α tal que $x' = \alpha x$, $y' = \alpha y$ e $t' = \alpha t$.
- Se P tem as coordenadas cartesianas (x,y), um dos seus conjuntos de coordenadas homogéneas é (x,y,1), conjunto este que é usado em computação gráfica.



Problema fundamental das transformações

- O facto de a translação não ser uma transformação linear de x e y impede que se possa efectuar uma série de transformações (translações e rotações por ordem arbitrária) através do produto de matrizes 2x2.
- Repare-se que é possível, no entanto, fazer *k* rotações através do produto de *k* matrizes de rotação.
- SOLUÇÃO: **coordenadas homogéneas**!

Translação 2D e Rotação 2D:
coordenadas homogéneas

Translação

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

Rotação

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

http://planetmath.org/encyclopedia/Geometry.html

Geometria métrica euclidiana

- Conjunto de transformações geométricas: **translações** e **rotações** (também designadas por isometrias).
- Com o uso de coordenadas homogéneas, as transformações da geometria métrica podem ser representadas por matrices 3x3, o que permite usar o operador produto para encontrar uma matriz de transformação que resulta da concatenação arbitrária de translações e rotações.
- O conjunto **I**(n) de isometrias em $\mathbf{R}^n$ e o operador de concatenação • formam um grupo **GI**(n)=(**I**(n),•).
- Invariante métrico fundamental:
  - distância entre pontos.
- Outros invariantes métricos:
  - ângulos
  - comprimentos
  - áreas
  - volumes
- Geometria euclidiana 2-dimensional: ($\mathbf{R}^2$,**GI**(2))

Definição de grupo: lembrete

Um conjunto C e uma operação $\circ$ formam um grupo $(C, \circ)$ se:

- Axioma de Fecho. Para quaisquer elementos $c_1, c_2 \in C$, $c_1 \circ c_2 \in C$.
- Axioma de Identidade. Existe um elemento identidade $i \in C$ tal que $c \circ i = c = i \circ c$, para qualquer $c \in C$.
- Axioma de Elemento Inverso. Para qualquer $c \in C$, existe um inverso $c^{-1} \in C$ tal que

$$c \circ c^{-1} = i = c^{-1} \circ c$$

- Axioma de Associatividade. Para quaisquer elementos $c_1, c_2, c_3 \in C$,

$$c_1 \circ (c_2 \circ c_3) = (c_1 \circ c_2) \circ c_3$$

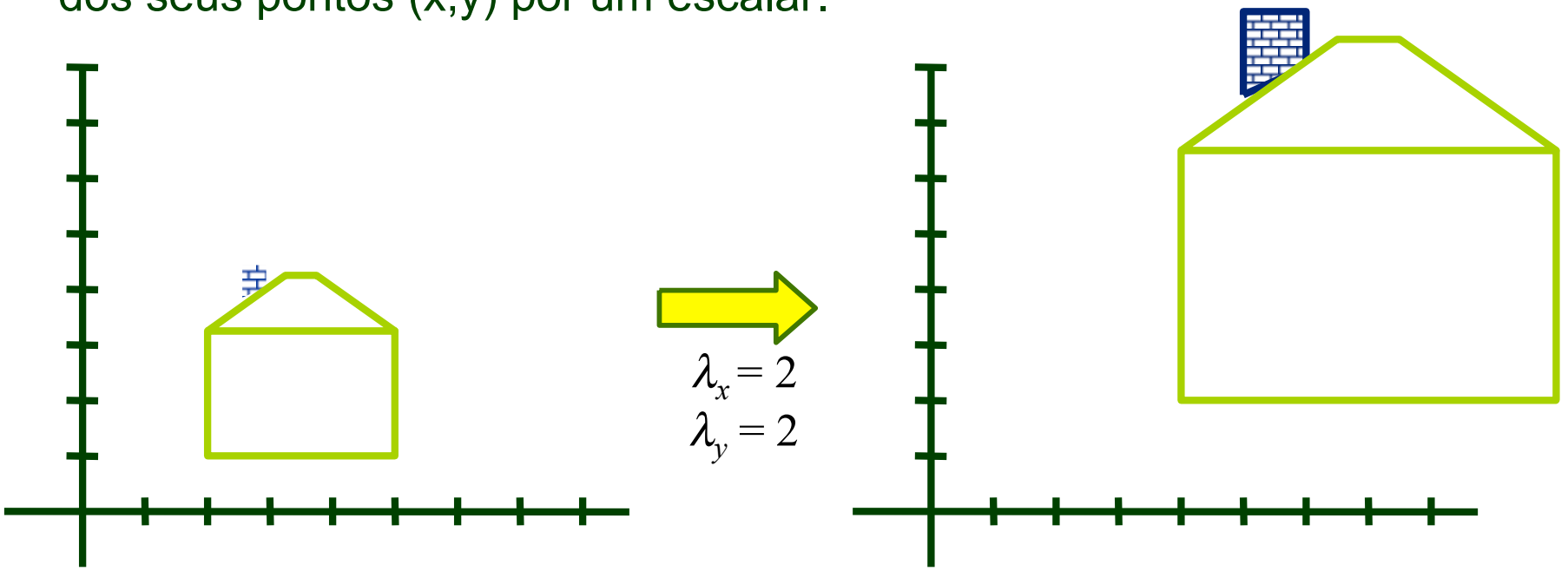
Geometria afim

- É uma generalização da geometria euclidiana.
- Conjunto de transformações afins (ou afinidades): translação, rotação, **variação de tamanho** (scaling) e **cisalhamento** (shearing).
- O conjunto **A**(n) de afinidades em $\mathbf{R}^n$ e o operador de concatenação • formam um grupo **GA**(n)=(**A**(n),•).
- Invariante fundamental:
  - paralelismo.
- Outros invariantes:
  - razão de distâncias entre quaisquer três pontos pertencentes a uma linha
  - colinearidade
- Exemplos:
  - um quadrado pode ser transformado num rectângulo
  - uma circunferência pode ser transformada numa elipse
- Geometria afim 2-dimensional: ($\mathbf{R}^2$,**GA**(2))

Variação de tamanho 2D

$$\begin{cases} x' = \lambda_x x \\ y' = \lambda_y y \end{cases}$$

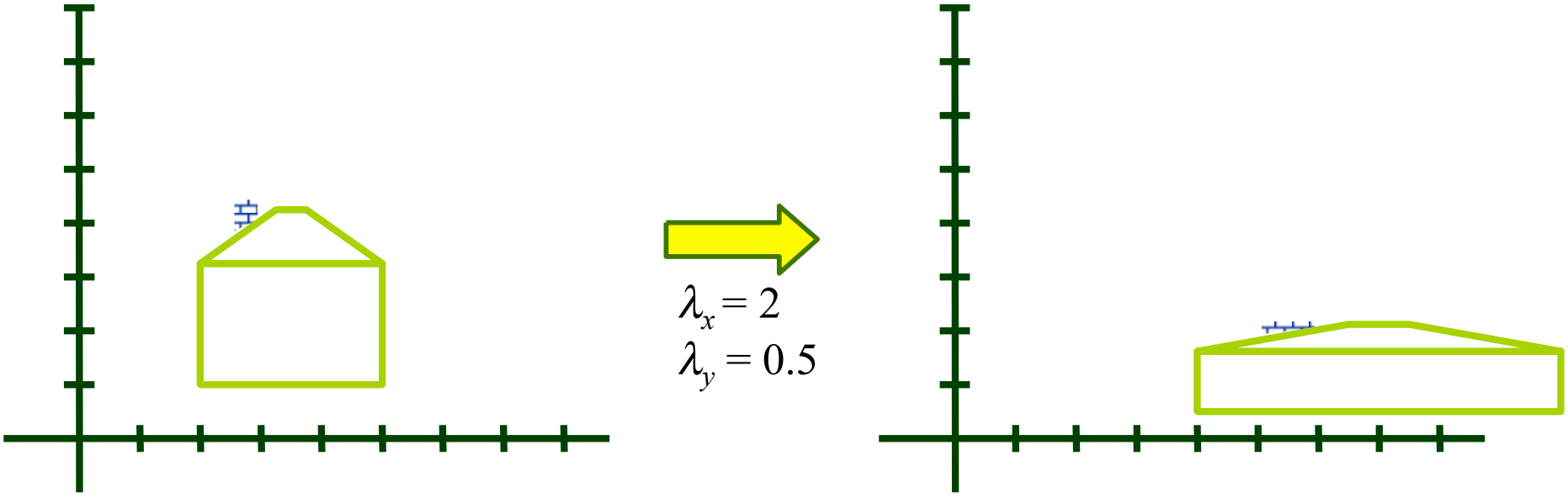
Variar o tamanho dum objecto é multiplicar cada componente de cada um dos seus pontos (x,y) por um escalar.



Variação de tamanho não-uniforme

$$\begin{cases} x' = \lambda_x x \\ y' = \lambda_y y \end{cases} \quad \text{com} \quad \lambda_x \neq \lambda_y$$

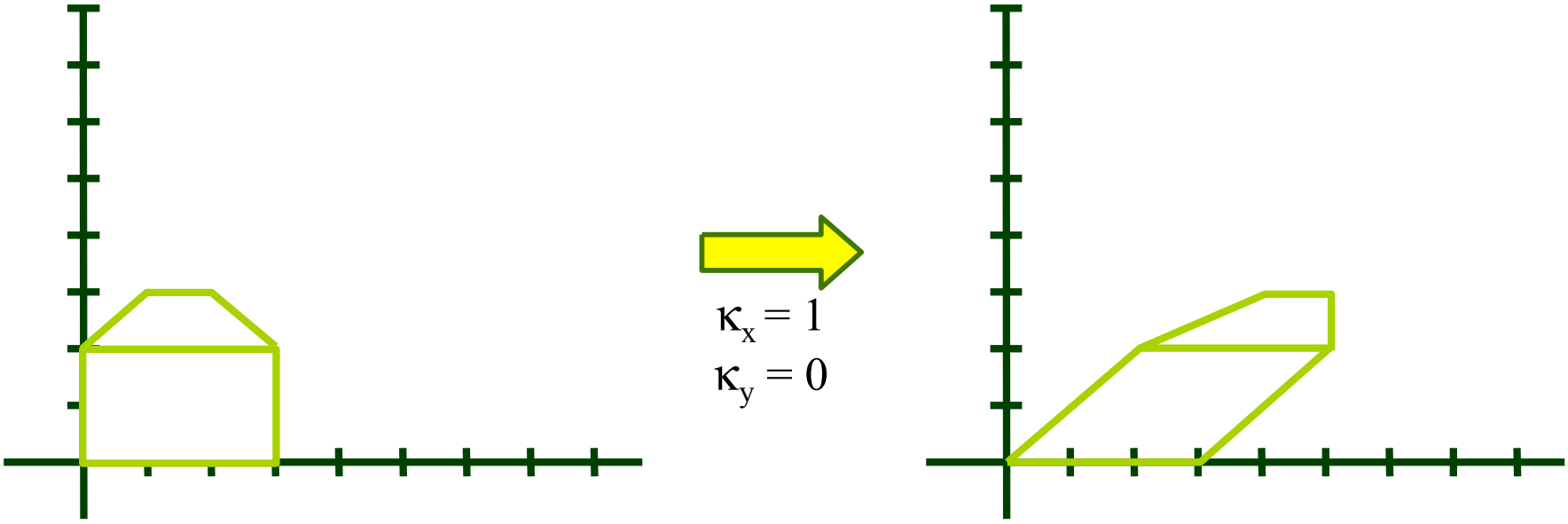
Variar o tamanho dum objecto é multiplicar cada componente de cada um dos seus pontos (x,y) por um escalar.



Cisalhamento

$$\begin{cases} x' = x + \kappa_x y \\ y' = y + \kappa_y x \end{cases}$$

Cisalhar um objecto é deformá-lo linearmente ao longo do eixo x ou do eixo y ou de ambos.



Só transformações lineares podem ser representadas por uma matriz 2x2

Resumo: representação matricial 3x3 de transformações afins 2D

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

Translação

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda_x & 0 & 0 \\ 0 & \lambda_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Variação de Tamanho

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

Rotação

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \kappa_x & 0 \\ \kappa_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

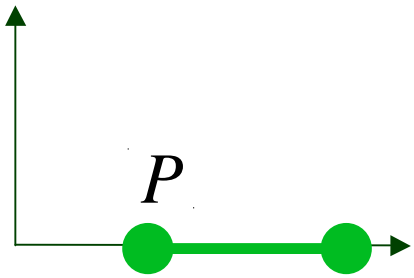
Cisalhamento

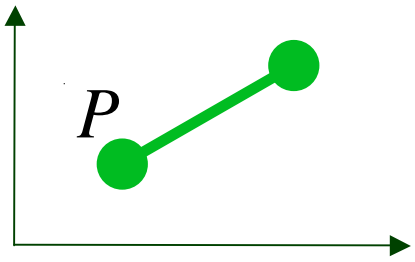
Composição de transformações afins 2D

- O operador de composição é o produto de matrizes.
- É uma consequência do Axioma da Associatividade da geometria afim e da dimensão 3x3 das matrizes associadas às transformações afins 2D.
- ATENÇÃO:
  - A ordem de composição de transformações afins é relevante.
  - O produto de matrizes não é uma operação comutativa.
  - A geometria afim não satisfaz o Axioma da Comutatividade.
- Exemplo:

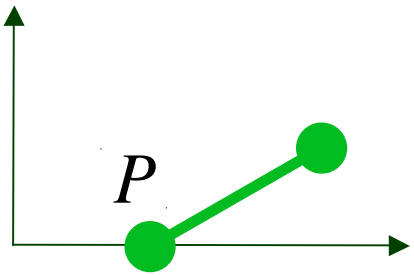
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \lambda_x & 0 & 0 \\ 0 & \lambda_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Exemplo: rotação $\theta = 30^\circ$ dum segmento $\overline{PQ}$ em torno de $P(2,0)$



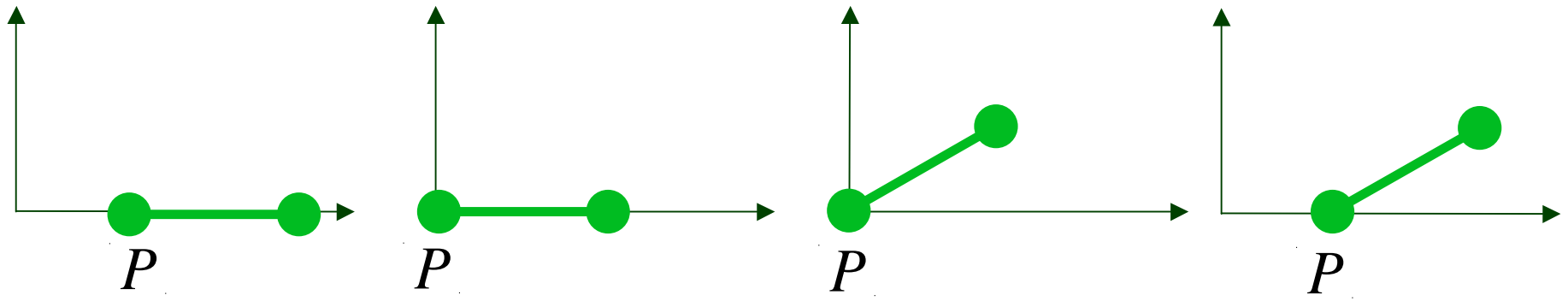


Incorrecto
Rot(30)



Correcto
Tr(-2) Rot(30) Tr(2)

Exemplo: rotação $\theta = 30^\circ$ dum segmento $\overline{PQ}$ em torno de $P(2,0)$



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Transformações afins 3D

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

Identidade

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda_x & 0 & 0 & 0 \\ 0 & \lambda_y & 0 & 0 \\ 0 & 0 & \lambda_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Variação de Tamanho

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Translação

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

Reflexão
relativamente ao plano YZ

Outras transformações afins 3D

Rotação em torno do eixo Z

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

Rotação em torno do eixo Y

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

Rotação em torno do eixo X

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

Transformações afins em OpenGL

- Há duas formas de especificar uma transformação :
  - Transformações geométricas pré-definidas: **`glTranslate`**, **`glRotate`** e **`glScale`**.
  - Transformações geométricas arbitrárias através de especificação directa de matrizes: **`glLoadMatrix`**, **`glMultMatrix`**
- Transformações geométricas são efectuadas pela matriz de modelação e visualização (*modelview*).
- Para operacionalizar estas transformações geométricas, há que designar a matriz *modelview* como a matriz corrente através da instrução **`glMatrixMode(GL_MODELVIEW)`**.
- A OpenGL tem uma pilha para cada um dos **quatro tipos de matrizes**: modelação e visualização (*modelview*), projecção *(projection)*, textura (*projection*) e cor (*color*).
- A pilha *modelview* é inicializada com a matriz identidade.

Transformações afins pré-definidas em OpenGL

- `glTranslate{f,d}(dx, dy, dz)`
  - Especifica uma translação segundo o vector `(dx,dy,dz)`, em que as componentes são números reais de precisão simples `f` ou dupla `d`.
- `glRotate{f,d}(angle, vx, vy, vz)`
  - Especifica uma rotação de `angle` graus em torno do eixo definido pelo vector `(vx,vy,vz)` na origem.
  - Note-se que esta função define uma rotação geral relativamente a um eixo que passa pela origem.
- `glScale{f,d}(sx, sy, sz)`
  - Especifica uma variação de tamanho segundo cada um dos eixos de coordenadas.
  - Esta função pode ser usada para fazer reflexões usando valores negativos nos factores de escala sx, sy ou sz.
  - ATENÇÃO: se algum factor de escala for zero, pode ocorrer um erro de processamento.

Operações com matrizes

em OpenGL

- `glLoadIdentity()`
  - Especifica a matriz identidade como a matriz de topo da pilha corrente. A matriz de topo duma pilha é sempre a matriz corrente.
- `glLoadMatrix{f,d}(<array>)`
  - Especifica a matriz corrente através dum array 1-dimensional com 16 elementos dados na ordem de coluna-a-coluna.
- `glMultMatrix{f,d}(<array>)`
  - Multiplica a matriz corrente *M* com a matriz *N* dada pelos elementos fornecidos pelo array 1-dimensional: $M=M \cdot N$

Exemplo: produção da matriz *modelview* $M=M_2 \cdot M_1$

- A sequência de instruções é a seguinte

```
glLoadIdentity();
glMultMatrixf(<array of M2>);
glMultMatrixf(<array of M1>);
```

- Note-se que a primeira transformação efectuada é a última que é especificada.

Exemplos em OpenGL

- Transformações afins com acumulação
- Transformações afins sem acumulação
- Transformações afins com acumulação controlada pela stack

Transformações 2D c/ acumulação

```
/* * quad.cc - Cumulative 2D transformations * Abel Gomes  */
#include <OpenGL/gl.h>         // Header File For The OpenGL Library
#include <OpenGL/glu.h>        // Header File For The GLu Library
#include <GLUT/glut.h>         // Header File For The GLut Library
#include <stdlib.h>

void draw(){
          // Make background colour yellow
     glClearColor( 100, 100, 0, 0 );
     glClear ( GL_COLOR_BUFFER_BIT );
          // modelview matrix for modeling transformations
     glMatrixMode(GL_MODELVIEW);
          // x-axis
     glColor3f(0,0,0);
     glBegin(GL_LINES);
          glVertex2f(0.0,0.0);
          glVertex2f(0.5,0.0);
     glEnd();
     // y-axis
     glColor3f(0,0,0);
     glBegin(GL_LINES);
          glVertex2f(0.0,0.0);
          glVertex2f(0.0,0.5);
     glEnd();
```

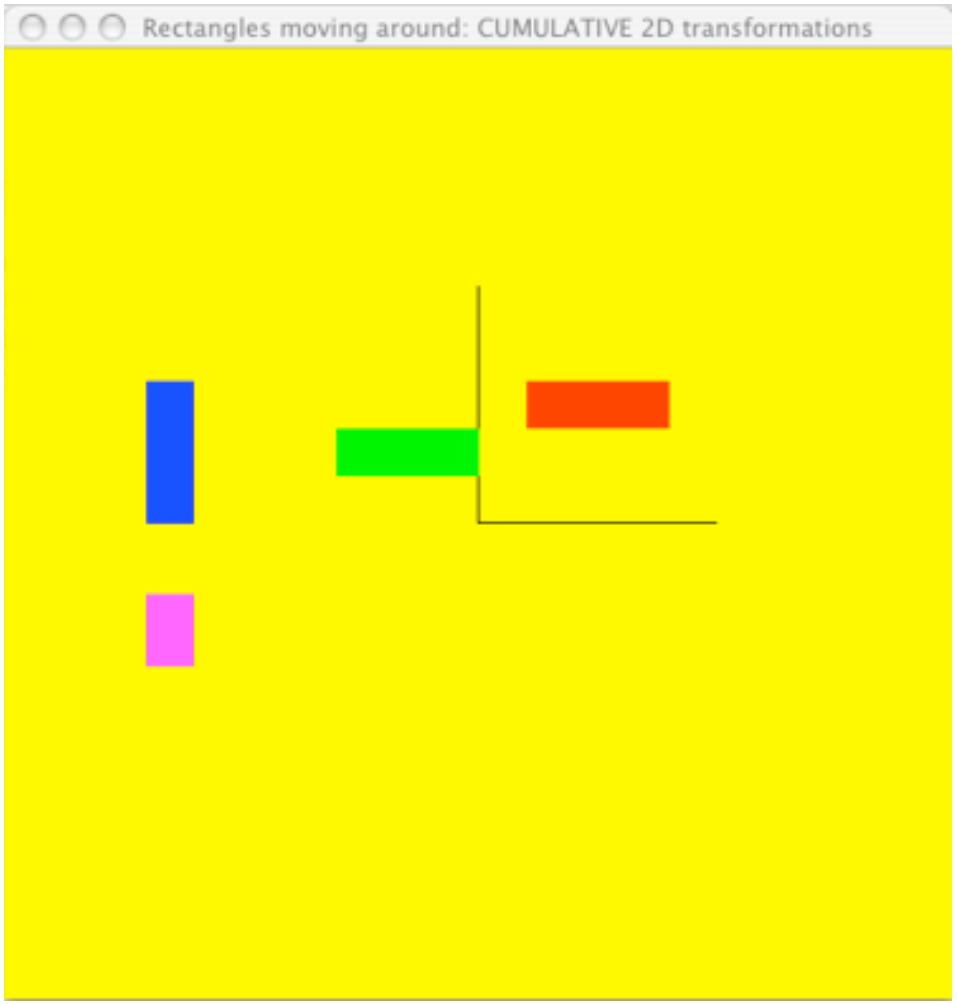
Transformações 2D c/ acumulação (cont.)

```
        // RED rectangle
    glColor3f( 1, 0, 0 );
    glRectf(0.1,0.2,0.4,0.3);
        // Translate GREEN rectangle
    glColor3f( 0, 1, 0 );
    glTranslatef(-0.4, -0.1, 0.0);
    glRectf(0.1,0.2,0.4,0.3);
        // Rotate and translate BLUE rectangle
    glColor3f( 0, 0, 1 );
        //glLoadIdentity();// reset the modelview matrix
    glRotatef(90, 0.0, 0.0,1.0);
    glRectf(0.1,0.2,0.4,0.3);
        // Scale, rotate and translate MAGENTA rectangle
    glColor3f( 1, 0, 1 );
        //glLoadIdentity();// reset the modelview matrix
    glScalef(-0.5, 1.0, 1.0);
    glRectf(0.1,0.2,0.4,0.3);
        // display rectangles
    glutSwapBuffers();
}                                                    // end of draw()
```

Transformações 2D c/ acumulação (cont.)

```
// Keyboard method to allow ESC key to quit
void keyboard(unsigned char key,int x,int y)
{
     if(key==27) exit(0);
}

int main(int argc,  char ** argv)
{
     glutInit(&argc, argv);
          // Double Buffered RGB display
     glutInitDisplayMode( GLUT_RGB | GLUT_DOUBLE);
          // Set window size
     glutInitWindowSize( 500,500 );
     glutCreateWindow("Rectangles moving around: CUMULATIVE 2D transformations");
          // Declare the display and keyboard functions
     glutDisplayFunc(draw);
     glutKeyboardFunc(keyboard);
          // Start the Main Loop
     glutMainLoop();
     return 0;
}
```

Transformações 2D
c/ acumulação (cont.): output



Transformações 2D s/ acumulação

```
/* * quad.cc - Non-cumulative 2D transformations * Abel Gomes  */
#include <OpenGL/gl.h>          // Header File For The OpenGL Library
#include <OpenGL/glu.h>         // Header File For The GLu Library
#include <GLUT/glut.h>          // Header File For The GLut Library
#include <stdlib.h>

void draw(){
          // Make background colour yellow
     glClearColor( 100, 100, 0, 0 );
     glClear ( GL_COLOR_BUFFER_BIT );
          // modelview matrix for modeling transformations
     glMatrixMode(GL_MODELVIEW);
          // x-axis
     glColor3f(0,0,0);
     glBegin(GL_LINES);
          glVertex2f(0.0,0.0);
          glVertex2f(0.5,0.0);
     glEnd();
     // y-axis
     glColor3f(0,0,0);
     glBegin(GL_LINES);
          glVertex2f(0.0,0.0);
          glVertex2f(0.0,0.5);
     glEnd();
```

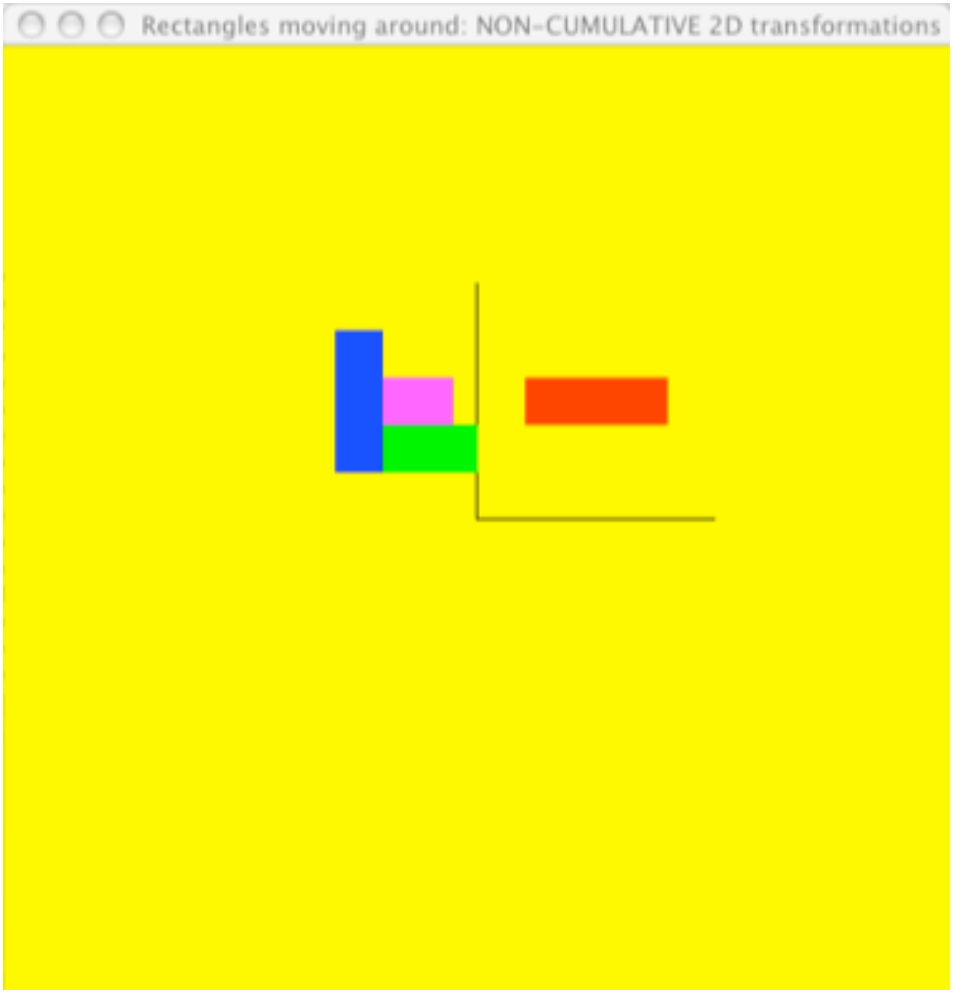
Transformações 2D s/ acumulação (cont.)

```
        // RED rectangle
    glColor3f( 1, 0, 0 );
    glRectf(0.1,0.2,0.4,0.3);
        // Translate GREEN rectangle
    glColor3f( 0, 1, 0 );
    glTranslatef(-0.4, -0.1, 0.0);
    glRectf(0.1,0.2,0.4,0.3);
        // Rotate BLUE rectangle
    glColor3f( 0, 0, 1 );
    glLoadIdentity();        // reset the modelview matrix
    glRotatef(90, 0.0, 0.0,1.0);
    glRectf(0.1,0.2,0.4,0.3);
        // Scale MAGENTA rectangle
    glColor3f( 1, 0, 1 );
    glLoadIdentity();        // reset the modelview matrix
    glScalef(-0.5, 1.0, 1.0);
    glRectf(0.1,0.2,0.4,0.3);
        // display rectangles
    glutSwapBuffers();
}                                                  // end of draw()
```

Transformações 2D s/ acumulação (cont.)

```
// Keyboard method to allow ESC key to quit
void keyboard(unsigned char key,int x,int y)
{
     if(key==27) exit(0);
}

int main(int argc,  char ** argv)
{
     glutInit(&argc, argv);
          // Double Buffered RGB display
     glutInitDisplayMode( GLUT_RGB | GLUT_DOUBLE);
          // Set window size
     glutInitWindowSize( 500,500 );
     glutCreateWindow("Rectangles moving around: NON-CUMULATIVE 2D transformations");
          // Declare the display and keyboard functions
     glutDisplayFunc(draw);
     glutKeyboardFunc(keyboard);
          // Start the Main Loop
     glutMainLoop();
     return 0;
}
```

Transformações 2D
s/ acumulação (cont.): output



Transf. 2D c/ acum. controlada pela stack

```
/* * quad.cc - Stack-cumulative 2D transformations * Abel Gomes  */
#include <OpenGL/gl.h>        // Header File For The OpenGL Library
#include <OpenGL/glu.h>       // Header File For The GLu Library
#include <GLUT/glut.h>        // Header File For The GLut Library
#include <stdlib.h>

void draw(){
          // Make background colour yellow
     glClearColor( 100, 100, 0, 0 );
     glClear ( GL_COLOR_BUFFER_BIT );
          // modelview matrix for modeling transformations
     glMatrixMode(GL_MODELVIEW);
          // x-axis
     glColor3f(0,0,0);
     glBegin(GL_LINES);
          glVertex2f(0.0,0.0);
          glVertex2f(0.5,0.0);
     glEnd();
     // y-axis
     glColor3f(0,0,0);
     glBegin(GL_LINES);
          glVertex2f(0.0,0.0);
          glVertex2f(0.0,0.5);
     glEnd();
```

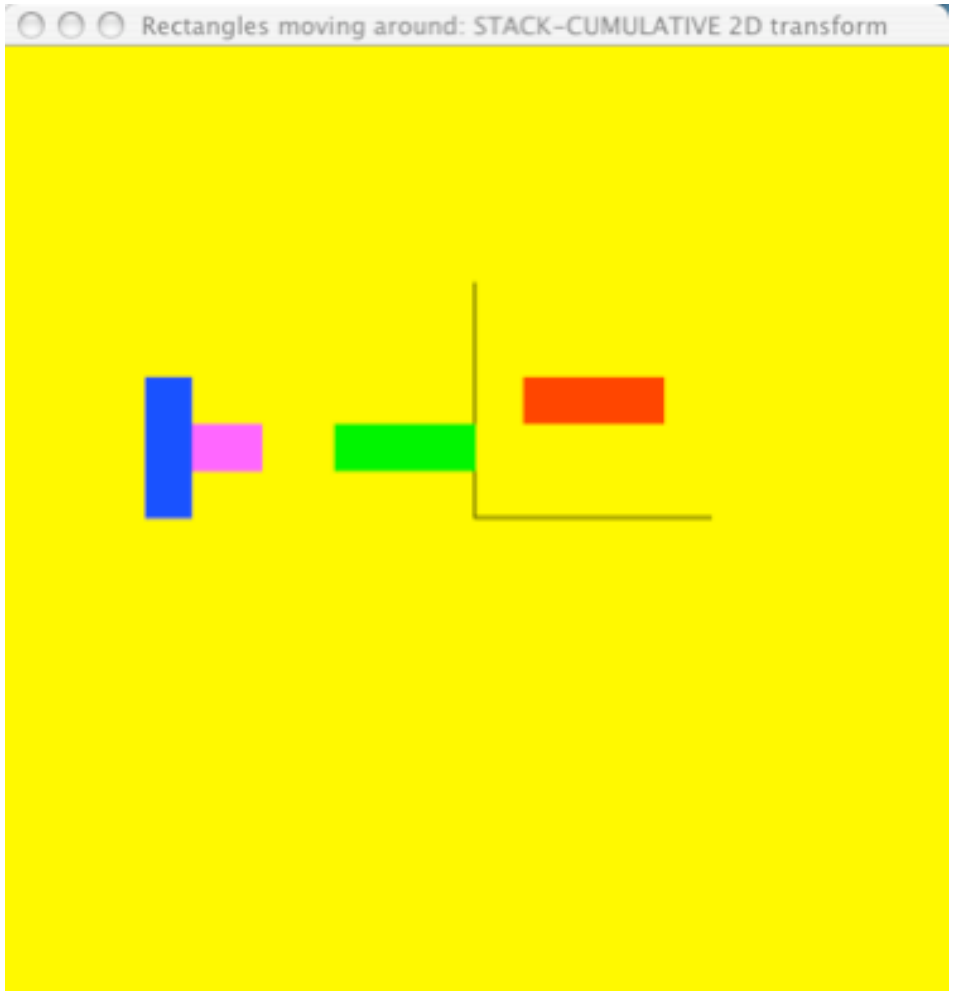
Transf. 2D c/ acum. controlada pela stack (cont.)

```
        // RED rectangle
    glColor3f( 1, 0, 0 );
    glRectf(0.1,0.2,0.4,0.3);
        // Translate GREEN rectangle
    glColor3f( 0, 1, 0 );
    glTranslatef(-0.4, -0.1, 0.0);
    glRectf(0.1,0.2,0.4,0.3);
        // save modelview matrix on the stack
    glPushMatrix();
        // Rotate and translate BLUE rectangle
    glColor3f( 0, 0, 1 );
    glRotatef(90, 0.0, 0.0,1.0);
    glRectf(0.1,0.2,0.4,0.3);
        // restore modelview matrix from the stack
    glPopMatrix();

        // Scale and translate MAGENTA rectangle
    glColor3f( 1, 0, 1 );
    glScalef(-0.5, 1.0, 1.0);
    glRectf(0.1,0.2,0.4,0.3);
        // display rectangles
    glutSwapBuffers();
}                                                   // end of draw()
```

Transf. 2D c/ acum. controlada pela stack (cont.):

```
// Keyboard method to allow ESC key to quit
void keyboard(unsigned char key,int x,int y)
{
     if(key==27) exit(0);
}

int main(int argc,  char ** argv)
{
     glutInit(&argc, argv);
          // Double Buffered RGB display
     glutInitDisplayMode( GLUT_RGB | GLUT_DOUBLE);
          // Set window size
     glutInitWindowSize( 500,500 );
     glutCreateWindow("Rectangles moving around: STACK-CUMULATIVE 2D transformations");
          // Declare the display and keyboard functions
     glutDisplayFunc(draw);
     glutKeyboardFunc(keyboard);
          // Start the Main Loop
     glutMainLoop();
     return 0;
}
```

Transf. 2D c/ acum. controlada pela stack (cont.): output



FIM