Transformações Geométricas 2D: Parte 3

Disciplina: Computação Gráfica (BCC35F)
Curso: Ciência da Computação

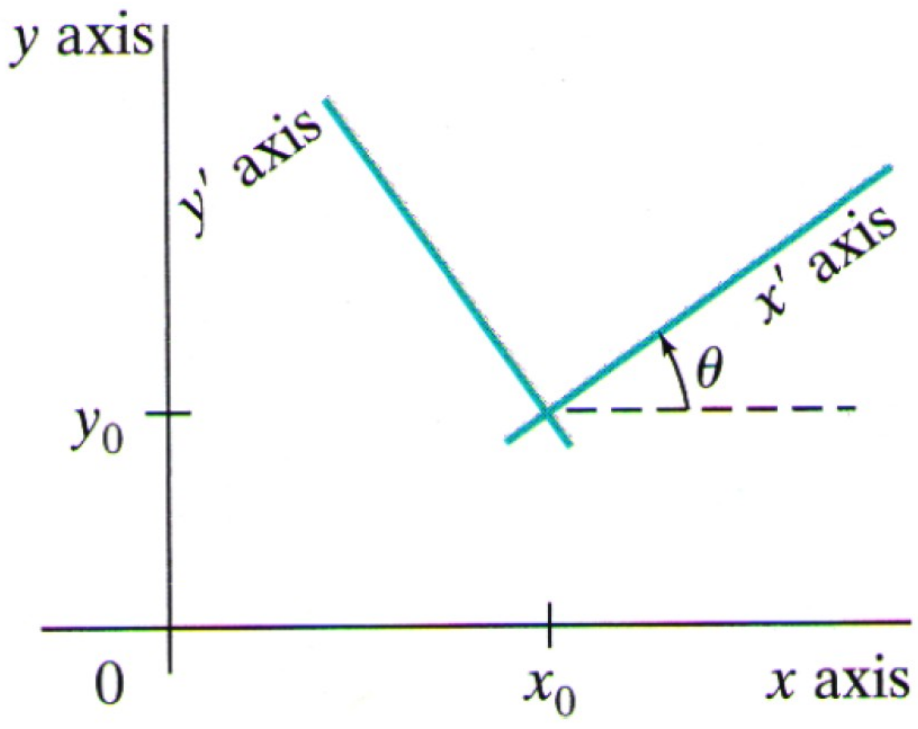
Prof. Walter T. Nakamura
waltertakashi@utfpr.edu.br

Campo Mourão - PR

Baseados nos materiais elaborados pelas professoras Aretha Alencar (UTFPR) e Rosane Minghim (USP)

Transformações entre Sistemas de Coordenadas

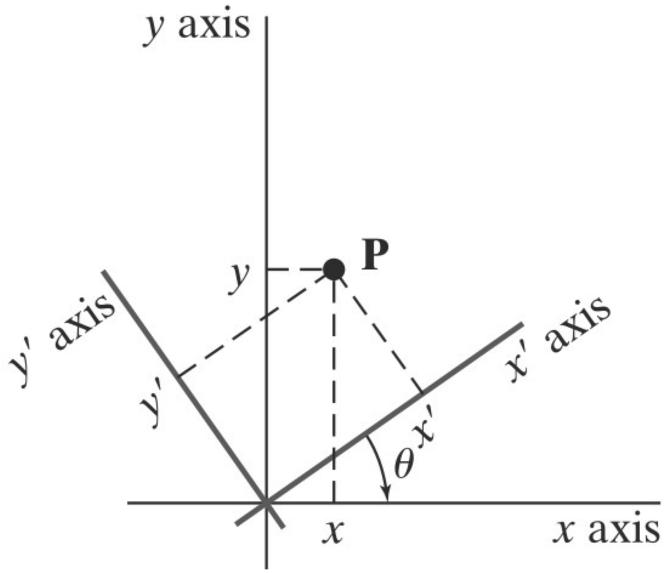
- Aplicações de computação gráfica envolvem a **transformação** de um sistema de coordenadas em outro em vários estágios do processamento da cena.



Transformações entre Sistemas de Coordenadas

- Para se transformar um sistema de coordenadas xy em outro $x'y'$:
  - Translade (x_0, y_0) para a origem $(0, 0)$
  - Rotacione em $-\theta$

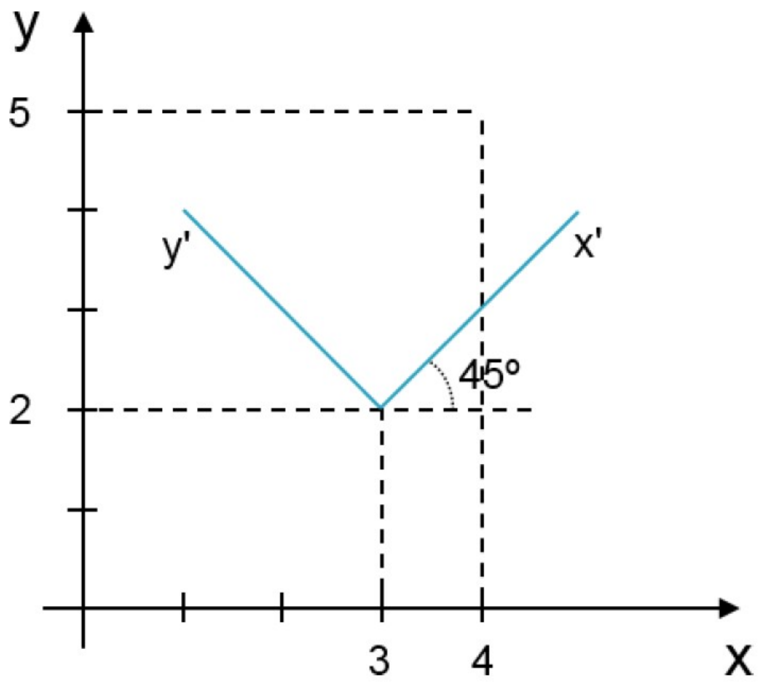
$$M_{xy,x'y'} = R(-\theta) \cdot T(-x_0, -y_0)$$



- Essa transformação nos dá a **geometria da cena** em relação ao <u>novo sistema de referência</u> x'y' (a cena é a mesma).

Exemplo - Solução

- Calcular a matriz de transformação de xy para $x'y'$ e as coordenadas finais do ponto P no sistema destino (P')
  - **Dicas:** $\theta = 45^o$; $(x_0, y_0) = (3, 2)$; $P = (4, 5)$; e $\text{sen}\ 45^o = \cos 45^o = \sqrt{2}/2$



$$M_{xy,x'y'} = R(-45^\circ) \cdot T(-3,-2) =$$

$$\begin{bmatrix} \cos 45^\circ & \text{sen}\,45^\circ & 0 \\ -\,\text{sen}\,45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 & 0 \\ -\sqrt{2}/2 & \sqrt{2}/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

Exemplo - Solução

- Calcular a matriz de transformação de xy para x'y' e as coordenadas finais do ponto P no sistema destino (P')
  - **Dicas:** $\theta = 45^o$; $(x_0, y_0) = (3, 2)$; $P = (4, 5)$; e $\text{sen } 45^o = \cos 45^o = \sqrt{2}/2$

$$M_{xy,x'y'} = R(-45^\circ)\cdot T(-3,-2) = \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 & -5\sqrt{2}/2 \\ -\sqrt{2}/2 & \sqrt{2}/2 & \sqrt{2}/2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P' = M_{xy,x'y'}\cdot P = \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 & -5\sqrt{2}/2 \\ -\sqrt{2}/2 & \sqrt{2}/2 & \sqrt{2}/2 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 4 \\ 5 \\ 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} 4\sqrt{2}/2 \\ 2\sqrt{2}/2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2.8284 \\ 1.4142 \\ 1 \end{bmatrix} = (2.8284, 1.4142)$$

Transformações entre Sistemas de Coordenadas

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

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

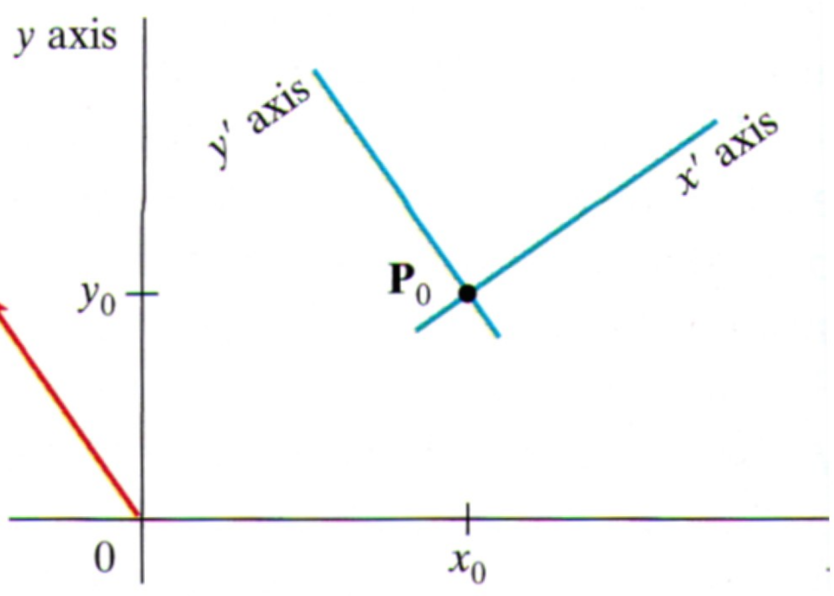
- Isto é, cada linha (r_{xx}, r_{xy}) e (r_{yx}, r_{yy}) forma um conjunto de vetores unitários ortogonais (ortonormais)

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

$$(r_{xx}, r_{xy}) \cdot (r_{yx}, r_{yy}) = r_{xx} \cdot r_{yx} + r_{xy} \cdot r_{yy} = 0 = \cos 90^\circ$$

Transformações entre Sistemas de Coordenadas

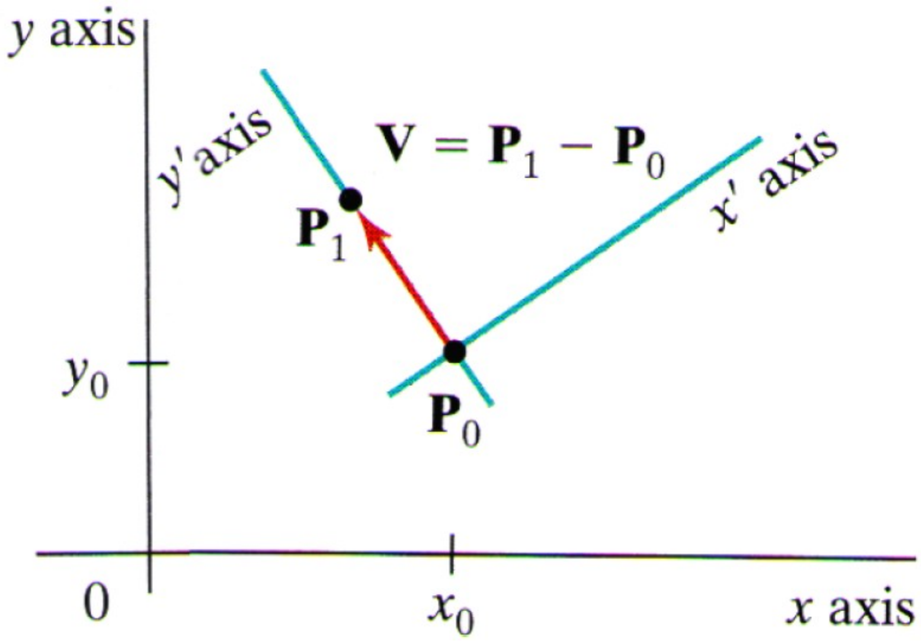
- Usando essa propriedade, outro método para transformar xy em x'y' pode ser derivado:
  - Para isso, inicialmente descrevemos a orientação do sistema de coordenadas x'y' por meio de um vetor V indicando a direção positiva do eixo y'



Transformações entre Sistemas de Coordenadas

- É possível especificar v com base em P_0 e P_1:

$$v = P1 - P0 = (V_x,V_y)$$



Transformações entre Sistemas de Coordenadas

- Para tornar o vetor V unitário, aplicamos:

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

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

$$u = (v_y, -v_x) = (u_x, u_y)$$

Transformações entre Sistemas de Coordenadas

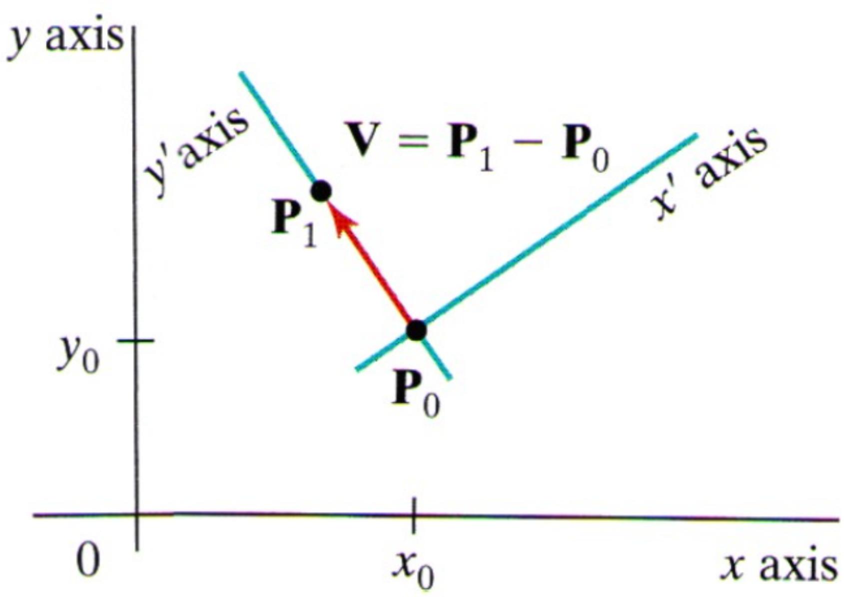
- Como qualquer matriz de rotação pode ser expressa por um **conjunto de vetores ortonormais**, então podemos escrever a matriz de rotação que faz x'y' coincidir com xy como:

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

- Observe, que a **translação** para alinhar a origem dos dois sistemas de coordenadas <u>ainda é necessária</u>

Exemplo - Solução

- Calcular a matriz de transformação de xy para x'y' e as coordenadas finais do ponto P no sistema destino (P').
  - **Dicas:** $(x_0, y_0) = (4, 3)$; $(x_1, y_1) = (3, 4)$; e $P = (-1, 5)$;



$$v = \frac{P_1 - P_0}{|P_1 - P_0|} = \frac{(x_1, y_1) - (x_0, y_0)}{|(x_1, y_1) - (x_0, y_0)|}$$

$$v = \frac{(3,4) - (4,3)}{|(3,4) - (4,3)|} = \frac{(-1,1)}{|(-1,1)|}$$

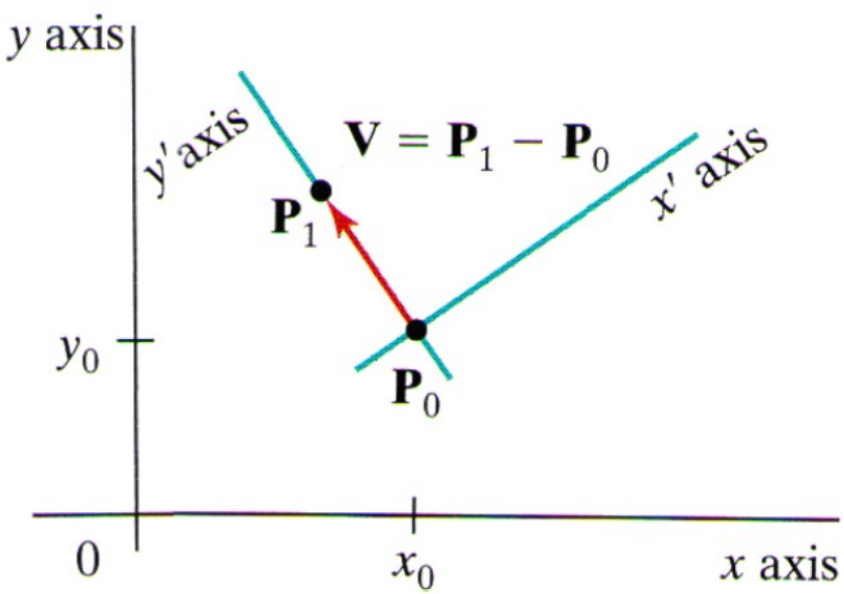
$$v = \frac{(-1,1)}{\sqrt{(-1)^2 + (1^2)}} = \frac{(-1,1)}{\sqrt{2}}$$

$$v = \left(\frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$$

$$v_x = \frac{-1}{\sqrt{2}} \qquad v_y = \frac{1}{\sqrt{2}}$$

Exemplo - Solução

- Calcular a matriz de transformação de xy para x'y' e as coordenadas finais do ponto P no sistema destino (P').
  - **Dicas:** θ = 45º; $(x_0, y_0) = (4, 3)$; $(x_1, y_1) = (3, 4)$; e $P = (-1, 5)$;



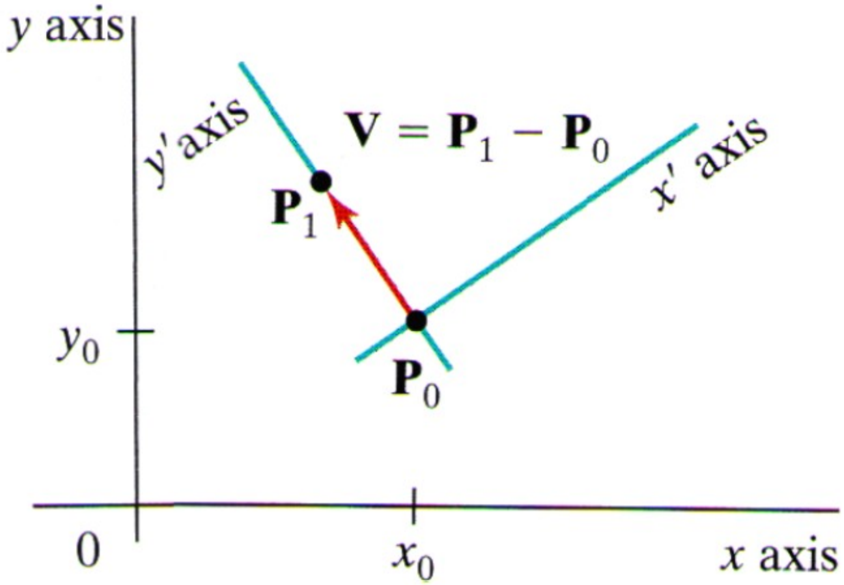
$$R = \begin{bmatrix} v_y & -v_x & 0 \\ v_x & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{-1}{\sqrt{2}} & 0 \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

Exemplo - Solução

- Calcular a matriz de transformação de xy para x'y' e as coordenadas finais do ponto P no sistema destino (P').
  - **Dicas:** θ = 45º; $(x_0, y_0) = (4, 3)$; $(x_1, y_1) = (3, 4)$; e $P = (-1, 5)$;



$$M_{xy,x'y'} = R \cdot T$$

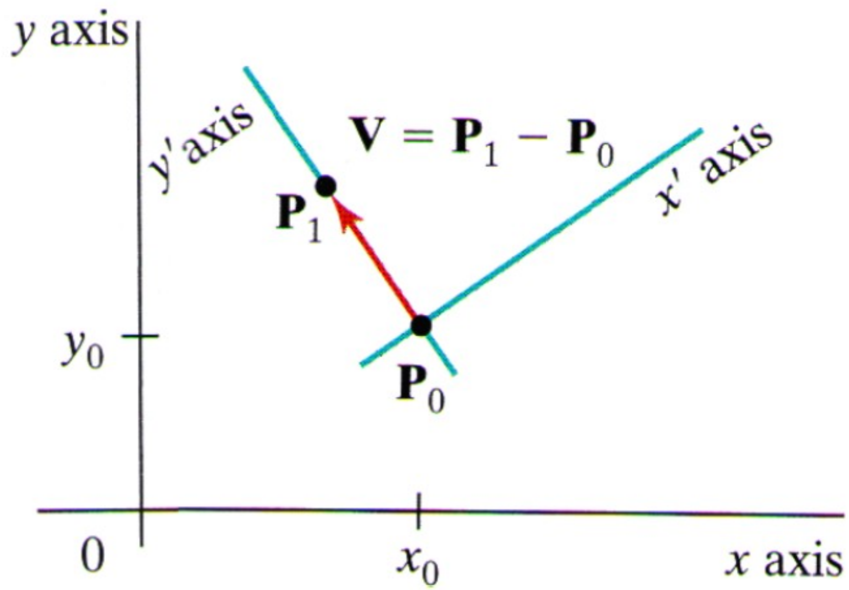
$$M_{xy,x'y'} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_{xy,x'y'} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{7}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & 1 \end{bmatrix}$$

$$P' = M_{xy,x'y'} \cdot P$$

Exemplo - Solução

- Calcular a matriz de transformação de xy para x'y' e as coordenadas finais do ponto P no sistema destino (P').
  - **Dicas:** $\theta = 45^{o}$; $(x_0, y_0) = (4, 3)$; $(x_1, y_1) = (3, 4)$; e $P = (-1, 5)$;

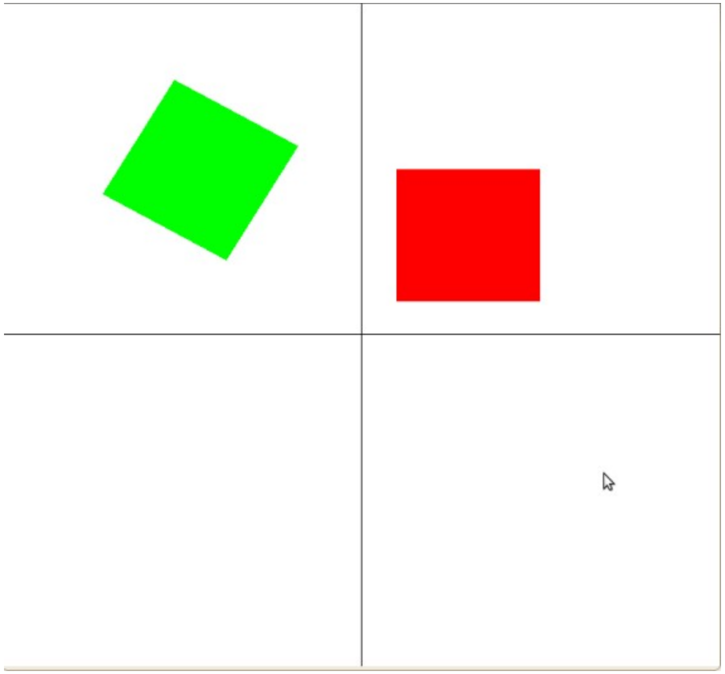


$$P' = M_{xy,x'y'} \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{7}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 5 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{3}{\sqrt{2}} \\ \frac{7}{\sqrt{2}} \\ 1 \end{bmatrix}$$

$$(x', y') = (-2.1213, 4.9497)$$



Transformações 2D em OpenGL usando matrizes

Desafio

- Desenvolva as funções para realizar a transformação de objetos 2D por meio da multiplicação de matrizes, considerando as matrizes, classes, tipos e assinaturas abaixo:

Matriz de rotação:

$$\begin{bmatrix} \cos(\theta) & -sen(\theta) & -x_r \cdot \cos(\theta) + y_r \cdot sen(\theta) + x_r \\ sen(\theta) & \cos(\theta) & -x_r \cdot sen(\theta) - y_r \cdot \cos(\theta) + y_r \\ 0 & 0 & 1 \end{bmatrix}$$

Matriz de escala:

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

Matriz de translação:

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Desafio

```
class wcPt2D {
    public: GLfloat x, y;
};


typedef GLfloat Matrix3x3[3][3];

// Matriz composta global
Matrix3x3 matComposite;

// Inicializa a matriz como uma matriz identidade
void matrix3x3SetIdentity(Matrix3x3 matIdent3x3);

// Realiza a multiplicação de duas matrizes
void matrix3x3PreMultiply(Matrix3x3 m1, Matrix3x3 m2);

void translate2D(GLflo0at tx, GLfloat ty);

void rotate2D(wcPt2D pivotPt, GLfloat theta);

void scale2D(GLfloat sx, GLfloat sy, wcPt2D fixedPt);

// Multiplica a matriz de rotação com a matriz composta
void transformVerts2D(GLint nVerts, wcPt2D * verts);
```

Translação

- O seguinte comando cria uma matriz 4 × 4 de translação:

```
glTranslate*(tx, ty, tz);
```

- **Parâmetros:**
  - O * pode ser substituído por `f` (float) ou `d` (double)
  - `tx, ty` e `tz` são os offsets de translação nos eixos `x, y` e `z` respectivamente
  - Para translação em 2D, coloque `tz = 0`
- **Exemplo:**

```
glTranslatef(25.0, -10.0, 0.0);
```

Rotação

- O seguinte comando cria uma matriz 4 × 4 de rotação:

```
glRotate*(theta, vx, vy, vz);
```

- **Parâmetros:**
  - O * pode ser substituído por `f` (float) ou `d` (double)
  - `theta` é o ângulo de rotação em graus
  - `vx, vy` e `vz` definem a orientação do eixo de rotação, que passa pelo ponto de origem (0, 0, 0)
  - Para a rotação em 2D, coloque `vx=0, vy=0` e `vz=1`
- **Exemplo:**

```
glRotate(90.0, 0.0, 0.0, 1.0);
```

Escala

- O seguinte comando cria uma matriz 4 × 4 de escala em relação à origem do sistema de coordenadas:

```
glScale*(sx, sy, sz);
```

- **Parâmetros:**
  - O * pode ser substituído por `f` (float) ou `d` (double)
  - `sx, sy` e `sz` são fatores de escalonamento
  - Para a escala em 2D, coloque `sz=1`
  - Valores negativos geram reflexão
- **Exemplo:**

```
glScalef(2.0, 0.2, 1.0);
```

Operações sobre Matrizes em OpenGL

- O seguinte comando concatena a matriz especificada com a matriz atual:

```
glMultMatrix*(matrix_param);
```

- **Parâmetros:**
  - O * pode ser substituído por `f` (float) ou `d` (double)
  - `matrix_param` é um vetor com 16 elementos que representa a transformação desejada
- **Exemplo:**

```
float cisalhamento_x[] = {1,shx,0,0,
                          0,1,0,0,
                          0,0,1,0,
                          0,0,0,1};
glMultMatrixf(cisalhamento_x);
```

Operações sobre Matrizes em OpenGL

- Antes de usar qualquer comando para uma transformação geométrica, é necessário informar que estaremos **utilizando matrizes** para <u>transformações geométricas</u>
- Especifique isso, usando:

```
glMatrixMode(GL_MODELVIEW);
```

Exemplo 1 – Escala com Ponto Fixo

```
#include <windows.h>
#include <stdio.h>
#include <GL/glut.h>

int init(void);
void display(void);
void desenhaCasa(void);

int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(200,0);   // posição da janela
    glutInitWindowSize(400,300);     // largura e altura da janela
    glutCreateWindow("Escala com Ponto Fixo");  // cria a janela

    init();                          // executa função de inicialização
    glutDisplayFunc(display);  // função "display" como a função de
                                 // callback de exibição
    glutMainLoop();              // mostre tudo e espere
    return 0;
}
// continua...
```

Exemplo 1 – Escala com Ponto Fixo

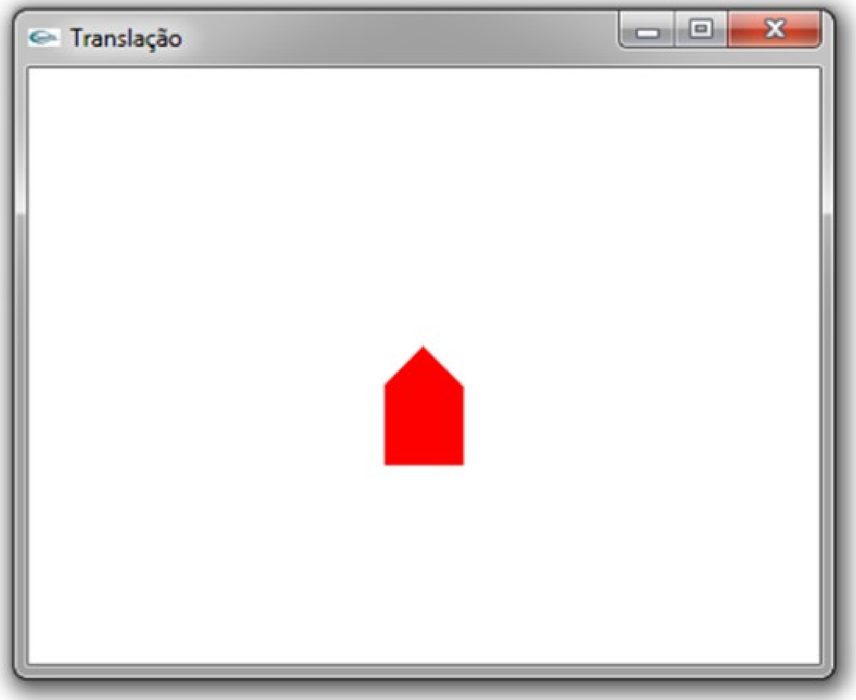
```
int init(void){
    glClearColor(1.0, 1.0, 1.0, 0.0);    // define a cor de fundo

    glMatrixMode(GL_PROJECTION); // carrega a matriz de projeção
    gluOrtho2D(0, 200, 0, 150);  // define projeção ortogonal 2D que
                                 // mapeia objetos da coordenada do
                                 // mundo para coordenadas da tela
}

void desenhaCasa(void){
    glBegin(GL_POLYGON);         // desenha uma casa
        glVertex2f(110, 50);
        glVertex2f(110, 70);
        glVertex2f(100, 80);
        glVertex2f(90, 70);
        glVertex2f(90, 50);
    glEnd();
}
```

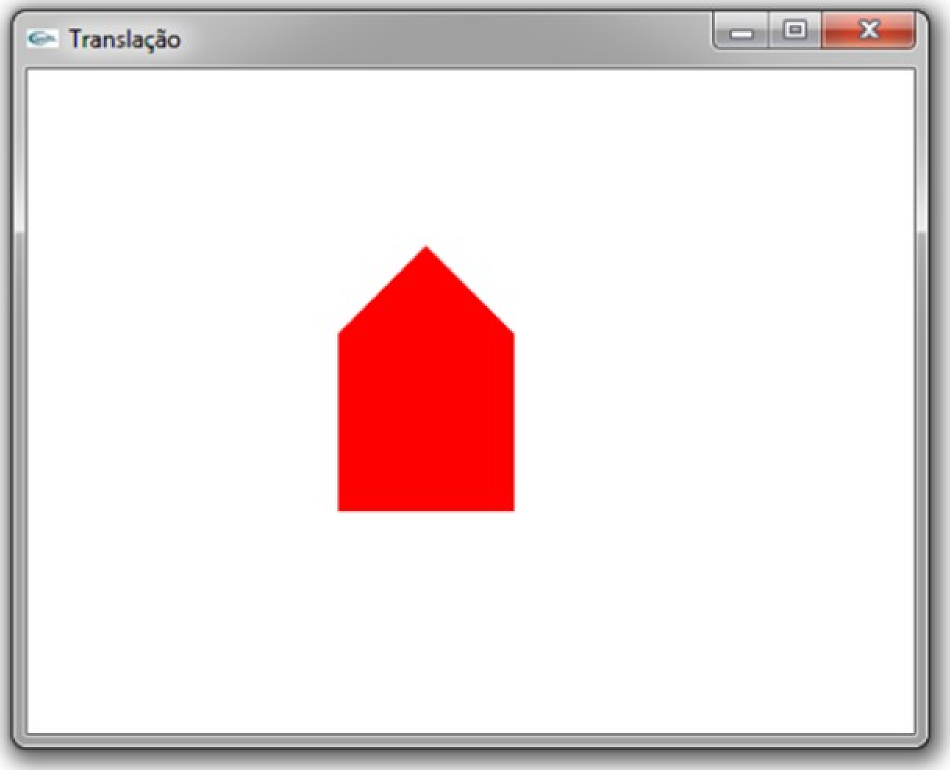
Exemplo 1 – Escala com Ponto Fixo

```
void display(void){
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0f, 0.0f, 0.0f); // desenha objetos com a cor vermelha
    glMatrixMode(GL_MODELVIEW);   // carrega a matriz de modelo

    // utiliza o primeiro vértice da lista como ponto fixo
    glTranslatef(110, 50, 0);     // ponto fixo para a posição original
    glScalef(2.0, 2.0, 1.0);      // faz a escala
    glTranslatef(-110, -50, 0);   // ponto fixo para a origem

    desenhaCasa();

    glFlush();     // desenha os comandos não executados
}
```

OpenGL: Cumulativo

- O método `display()` é chamado mais de uma vez (modificação do tamanho da janela) – o objeto é escalado duas vezes!





Exemplo 2 – OpenGL: Cumulativo

- **Solução:** carregar a matriz identidade
  - `glLoadIdentity();`

```
void display(void){
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0f, 0.0f, 0.0f);  // desenha objetos com a cor vermelha
    glMatrixMode(GL_MODELVIEW);   // carrega a matriz de modelo

    glLoadIdentity();             // carrega a matriz identidade

    // utiliza o primeiro vértice da lista como ponto fixo
    glTranslatef(110, 50, 0);     // ponto fixo para a posição original
    glScalef(2.0, 2.0, 1.0);      // faz a escala
    glTranslate(-110, -50, 0);    // ponto fixo para a origem

    desenhaCasa();

    glFlush();     // desenha os comandos não executados
}
```

Exemplo 3 – OpenGL: Ordem das Transformações

- Primeiro **rotaciono** usando um ponto fixo, depois faço a **translação**

```
void display(void){
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0f, 0.0f, 0.0f); // desenha objetos com a cor vermelha
    glMatrixMode(GL_MODELVIEW);   // carrega a matriz de modelo

    glLoadIdentity();             // carrega a matriz identidade

    glTranslatef(50, 0, 0);       // faço a translação
    glTranslatef(110, 50, 0);     // ponto fixo para a posição original
    glRotatef(90, 0, 0, 1);       // rotaciono
    glTranslate(-110, -50, 0);    // ponto fixo para a origem

    desenhaCasa();

    glFlush();     // desenha os comandos não executados
}
```

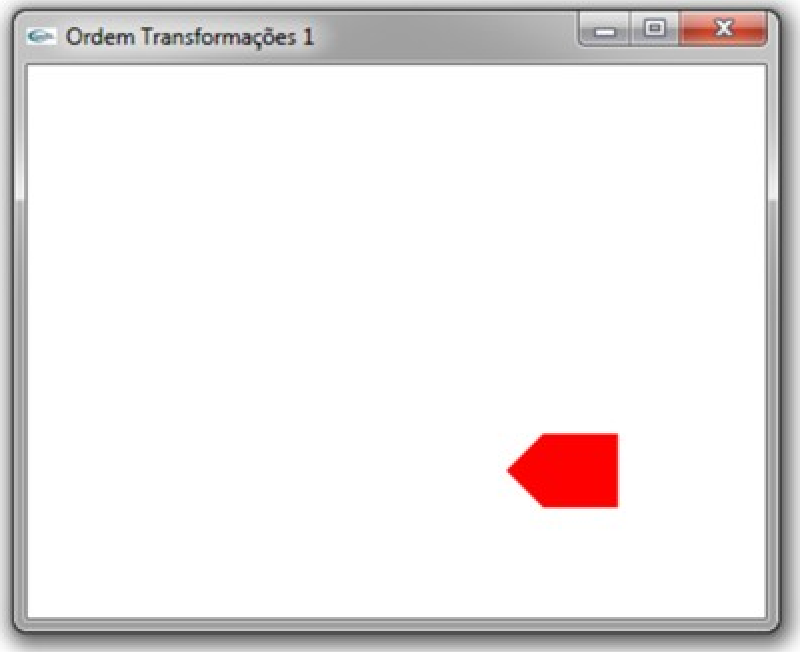
Exemplo 3 – OpenGL: Ordem das Transformações

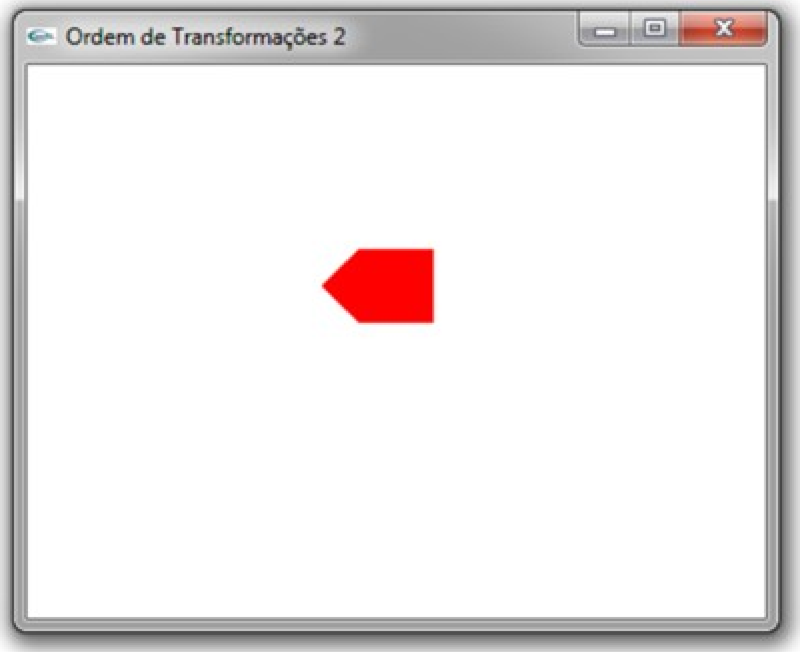
- Primeiro faço a **translação**, depois **rotaciono** usando um ponto fixo

```
void display(void){
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0f, 0.0f, 0.0f);  // desenha objetos com a cor vermelha
    glMatrixMode(GL_MODELVIEW);   // carrega a matriz de modelo

    glLoadIdentity();             // carrega a matriz identidade

    glTranslatef(110, 50, 0);     // ponto fixo para a posição original
    glRotatef(90, 0, 0, 1);       // rotaciono
    glTranslatef(-110, -50, 0);   // ponto fixo para a origem
    glTranslate(50, 0, 0);        // faço a translação

    desenhaCasa();

    glFlush();      // desenha os comandos não executados
}
```

OpenGL – Ordem das Transformações

- A ordem das transformações leva a resultados completamente diferentes!



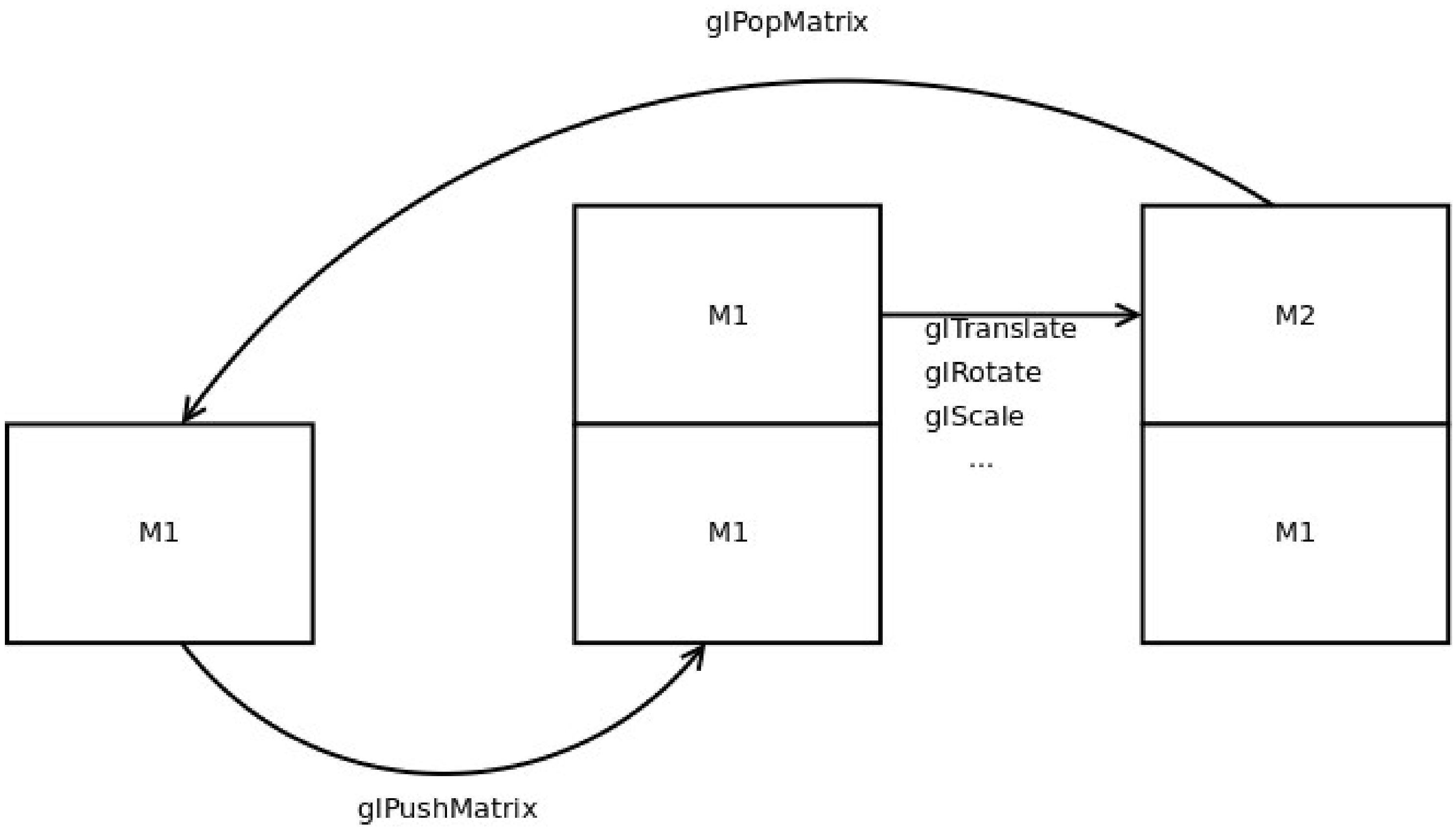


Pilha de Transformações

- Cada modo definido por `glMatrixMode` possui uma pilha de matrizes. A matriz corrente de cada modo é a matriz do topo da sua respectiva pilha.
- A função `glPushMatrix()` duplica a matriz do topo da pilha e essa cópia se torna o novo topo da pilha
- A função `glPopMatrix()` desempilha a matriz atual do respectivo modo ativo.
- A função `glLoadIdentity()` atribui o valor da matriz identidade à matriz do topo da pilha corrente.

```
//Empilha uma copia da matriz atual
void glPushMatrix();
//Desempilha a matriz atual
void glPopMatrix();
//Carrega valores da matriz identidade
void glLoadIdentity();
```

Pilha de Transformações



Pilha de Transformações

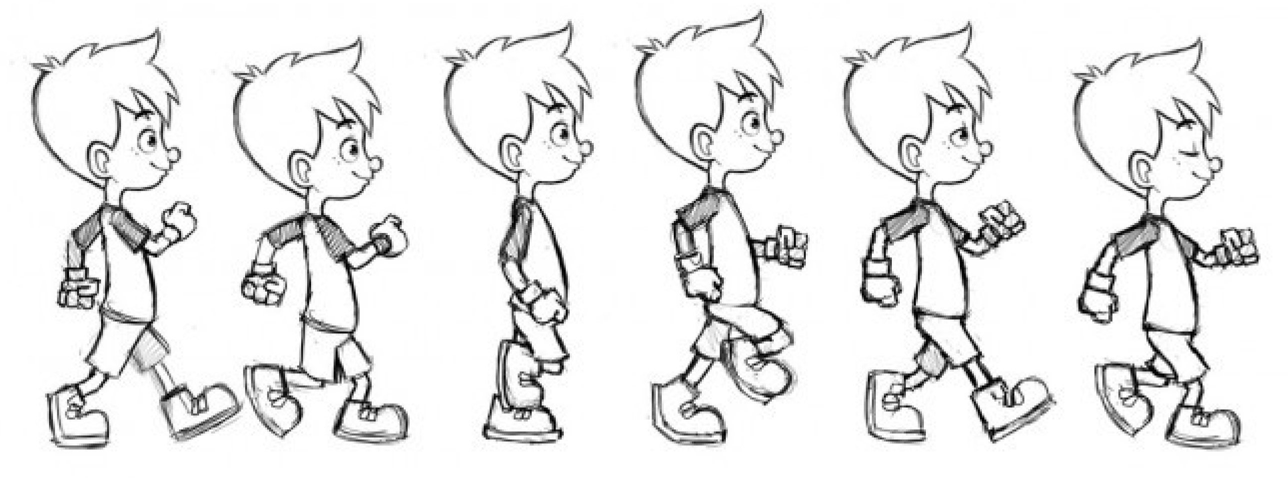
```
...
glMatrixMode(GL_MODELVIEW);
...
glPushMatrix();
glTranslated(tx, ty, 0);
glRotated(theta, 0.0, 0.0, 1.0);
glScale(sx, sy, 1.0);
...
//DESENHA ALGUMA COISA
...
glPopMatrix();
...
//DESENHA OUTRA COISA SEM CONSIDERAR AS
//TRANSFORMACOES ANTERIORES
...
```

Pilha de Transformações

- As matrizes também podem ser **salvas** ou **recarregadas**, possibilitando que o programador utilize qualquer tipo de combinação de matrizes para compor sua imagem.

```
void glGetDoublev(GLenum pname, GLdouble *params);
void glGetFloatv(GLenum pname, GLfloat *params);
void glLoadMatrixd(const GLdouble *m);
void glLoadMatrixf(const GLfloat *m);
```

```
GLfloat m[16];
...
glGetFloatv(GL_MODELVIEW_MATRIX, m);
...
glLoadMatrixf(m);
...
```



Animações 2D em OpenGL

Animações

- Animação tradicional envolve uma **sequência de imagens** em alta velocidade
- Velocidade de exibição (frame rate) varia de acordo com a mídia utilizada
- Cada imagem (quadro, cena) deve possuir uma **ligeira diferença** em relação às outras, criando a ilusão de movimento
- As diferenças podem ser na movimentação dos objetos, suas cores, formas etc. Também é possível modificar a posição do observador quando a imagem for 3D.

Animações

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

- Como a imagem precisa ser **modificada continuamente** a tela precisa ser atualizada constantemente
  - É necessário evitar que a imagem fique "piscando" quando a tela é redesenhada
  - Para evitar esse problema o OpenGL utiliza dois buffers para exibição
    - Enquanto um está sendo preenchido, o outro está sendo exibido
  - O parâmetro `GLUT_DOUBLE` deve ser utilizado na função `glutInitDisplayMode` para que a OpenGL utilize os dois buffers

Animações

```
//Executa o parametro quando nenhum evento esta ocorrendo
void glutIdleFunc(void (*func)(void));
//Executa a funcao parametro a cada msecs
void glutTimerFunc(unsigned int msecs,void (*func)(int value), value);
//Alterna os buffers da tela
void glutSwapBuffers();
```

Exercícios

- 1) Faça uma animação que gire a "casinha" apresentada no exemplo no seu centróide (100, 65). **Dicas:** utilize as funções abaixo:
  - `glutIdleFunc();`
  - `glutPostRedisplay();`
  - `glutSwapBuffers(); // utilize no lugar de glFlush()`

- 2) Complemente a animação anterior para que a "casinha" gire e aumente/diminua de tamanho constantemente.

Atividades Práticas

- 1) Crie uma animação para que um quadrado movimente-se para a esquerda e para a direita da tela. Quando o quadrado "bater" no final da tela, este deve voltar, fazendo o caminho inverso.
- 2) Modifique a animação anterior (crie um novo arquivo) para que o quadrado se movimente em um determinado ângulo e "quique" ao bater em uma das extremidades da tela. Ao bater na tela, o fundo deve trocar de cor de forma aleatória.
- 3) Modifique a animação anterior (crie um novo arquivo) para que o objeto mude aleatoriamente de formato cada vez que "quicar" nas extremidades da tela. Utilize pelo menos 3 formas (ex: quadrado, círculo e triângulo).
- 4) Crie uma animação de um relógio analógico com as marcações de tempo e os 3 ponteiros (horas, minutos, segundos).
- 5) Modifique o relógio anterior para que reflita o horário atual do sistema.