Transformações Geométricas

Computação Gráfica

Professora: Lis Custódio

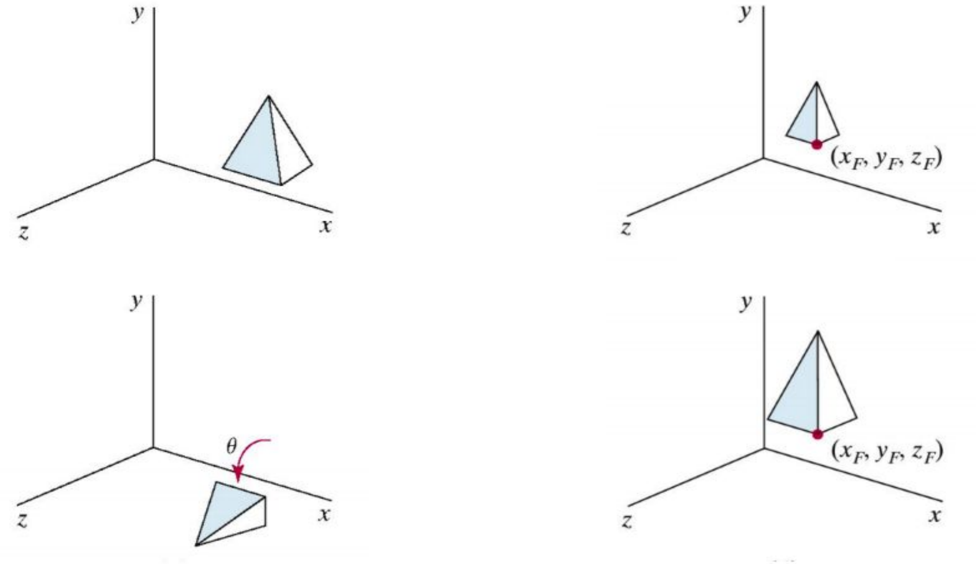
Tópicos da aula de hoje

- Por que transformações?
- Classificação das transformações
- Transformações
  - Translação
  - Escala
  - Rotação
- Coordenadas Homogêneas
- Combinando transformações

Por que transformações?

Por que transformações?

Em CG, uma vez tendo um objeto definido, “transformações” são utilizadas para mover este objeto, mudar seu tamanho (escala) ou rotacioná-lo.



Por que transformações?

Podemos utilizar uma matriz 2x2 para modificar os pontos de um objeto, consequentemente, para modificar a posição e estrutura desse objeto.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \end{bmatrix}$$

A tipo de operação, que pega um ponto em duas dimensões e leva em outro ponto, através de uma simples multiplicação de matrizes, chamamos **transformação linear**.

Classificação de transformações

Transformações Rígidas

Transformações Não-Rígidas

Classificação de transformações

Transformações Rígidas

São transformações geométricas que preservam a forma, o tamanho e os ângulos dos objetos. Elas são:

Transformações Não-Rígidas

Classificação de transformações

Transformações Rígidas

São transformações geométricas que preservam a forma, o tamanho e os ângulos dos objetos. Elas são:

- Rotação
- Translação
- Reflexão

Transformações Não-Rígidas

Classificação de transformações

Transformações Rígidas

São transformações geométricas que preservam a forma, o tamanho e os ângulos dos objetos. Elas são:

- Rotação
- Translação
- Reflexão

Transformações Não-Rígidas

São transformações geométricas que preservam o paralelismo entre as linhas mas não seus comprimentos e ângulos.

Classificação de transformações

Transformações Rígidas

São transformações geométricas que preservam a forma, o tamanho e os ângulos dos objetos. Elas são:

- Rotação
- Translação
- Reflexão

Transformações Não-Rígidas

São transformações geométricas que preservam o paralelismo entre as linhas mas não seus comprimentos e ângulos. Exemplo de transformações Não-Rígidas:

- Escala
- Cisalhamento

Classificação de transformações

Transformações Afins

- Basicamente as transformações afins são:

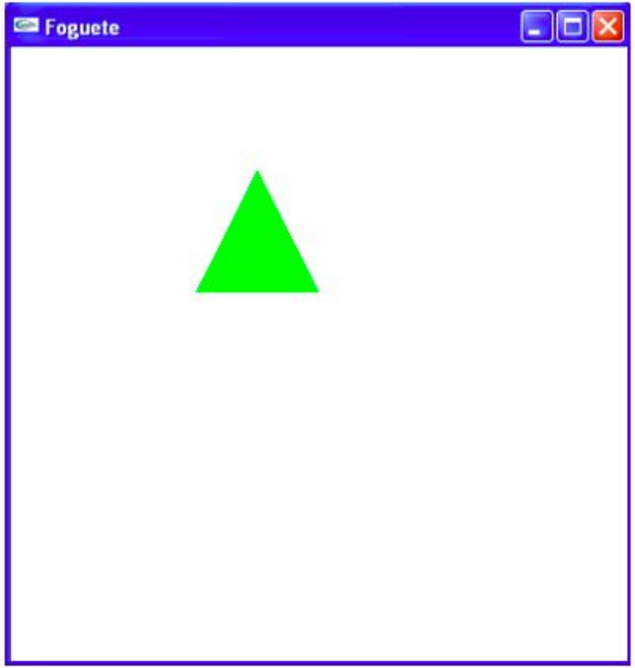
  $\boxed{\text{Transformações Lineares + Translações}}$

- Desta forma, podemos afirmar que toda transformação linear é também afim, mas o contrário não é verdade

Desenhando um foquete

Bico

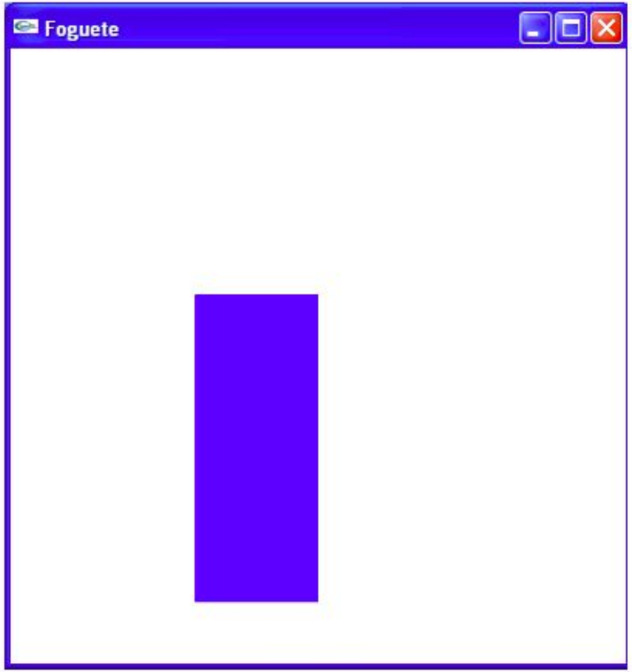
```
void Bico()
{
    glBegin(GL_TRIANGLES);
      glColor3f(0,1,0);
      glVertex3f(3.0,6.0,0);
      glVertex3f(4.0, 8.0,0);
      glVertex3f(5.0,6.0,0);
   glEnd();
}
```



Desenhando um foguete

Corpo

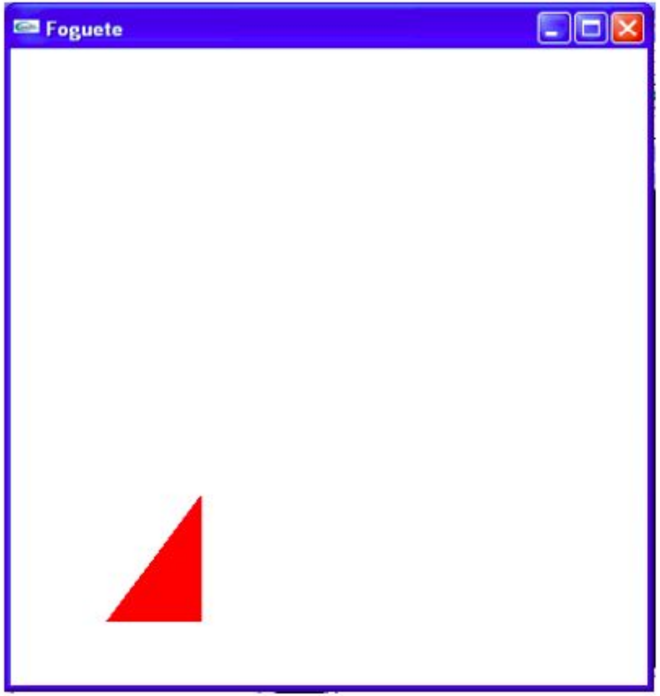
```
void Corpo()
{
    glBegin(GL_QUADS);
      glColor3f(0,0,1);
      glVertex3f(3.0,1.0,0);
      glVertex3f(5.0, 1.0,0);
      glVertex3f(5.0,6.0,0);
      glVertex3f(3.0,6.0,0);
   glEnd();
}
```



Desenhando um foquete

Asa Esquerda

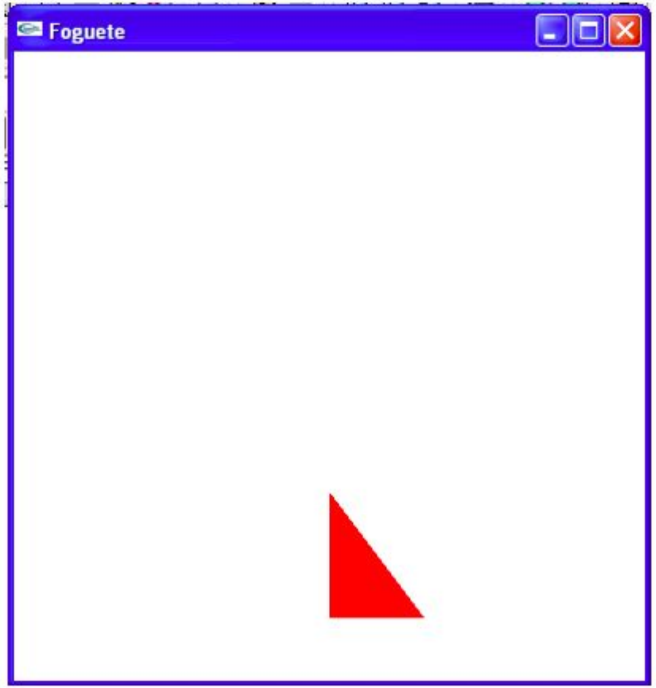
```
void AsaEsquerda()
{
    glBegin(GL_TRIANGLES);
      glColor3f(1,0,0);
      glVertex3f(1.5,1.0,0);
      glVertex3f(3.0, 1.0,0);
      glVertex3f(3.0,3.0,0);
   glEnd();
}
```



Desenhando um foquete

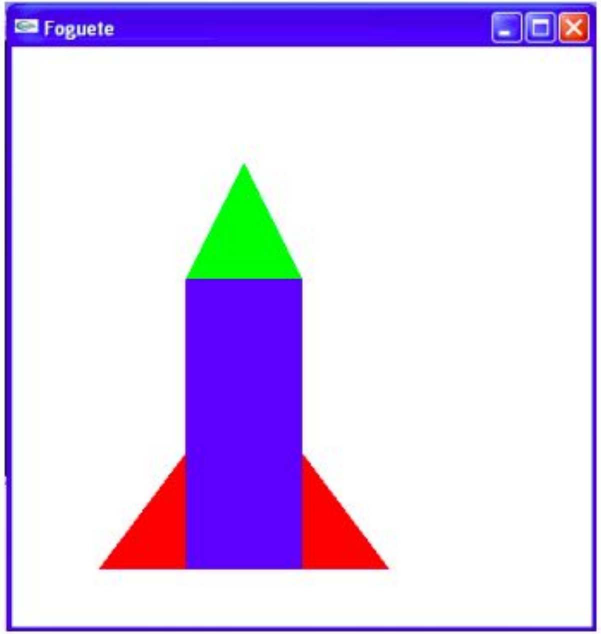
Asa Direita

```
void AsaDireita()
{
    glBegin(GL_TRIANGLES);
      glColor3f(1,0,0);
      glVertex3f(5.0,1.0,0);
      glVertex3f(6.5, 1.0,0);
      glVertex3f(5.0,3.0,0);
   glEnd();
}
```



Desenhando um foquete

```
void DesenhaFoguete(void)
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    Bico();
    Corpo();
    AsaEsquerda();
    AsaDireita();
    glFlush();
}
```



Escala

- Escalas multiplicam todas as coordenadas
- **Atenção:** Objetos podem mudar de lugar ao aplicarmos operações de escala

$$\text{scale}(s_x, s_y) = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

Escala

- Escalas multiplicam todas as coordenadas
- **Atenção:** Objetos podem mudar de lugar ao aplicarmos operações de escala.

$$\mathrm{scale}(s_x, s_y) = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

Dado um ponto (x,y), uma escala de parâmetros Sx e Sy, será da forma:

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

Escala

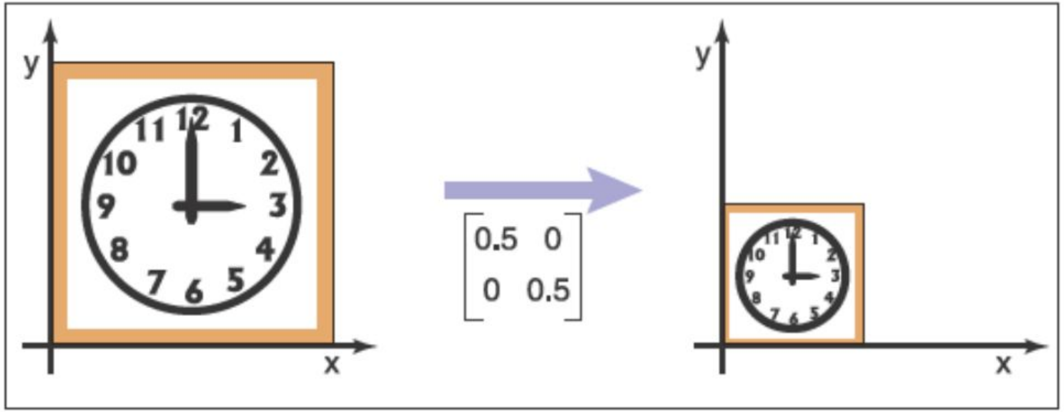
Exemplo:

$$\text{scale}(0.5, 0.5) = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$$

Escala

Exemplo:

$$\text{scale}(0.5, 0.5) = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$$



Escala

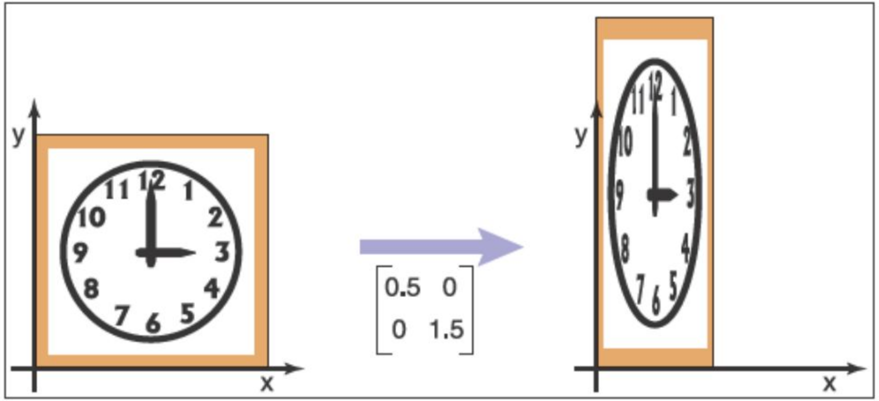
Exemplo:

$$\text{scale}(0.5, 1.5) = \begin{bmatrix} 0.5 & 0 \\ 0 & 1.5 \end{bmatrix}$$

Escala

Exemplo:

$$\text{scale}(0.5, 1.5) = \begin{bmatrix} 0.5 & 0 \\ 0 & 1.5 \end{bmatrix}$$



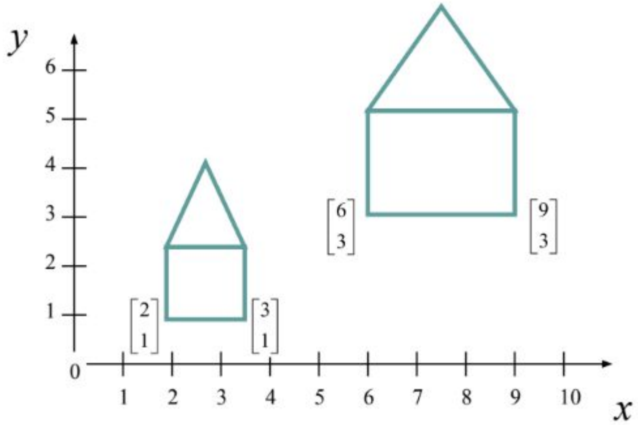
Escala

- Escalas multiplicam todas as coordenadas

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

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

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



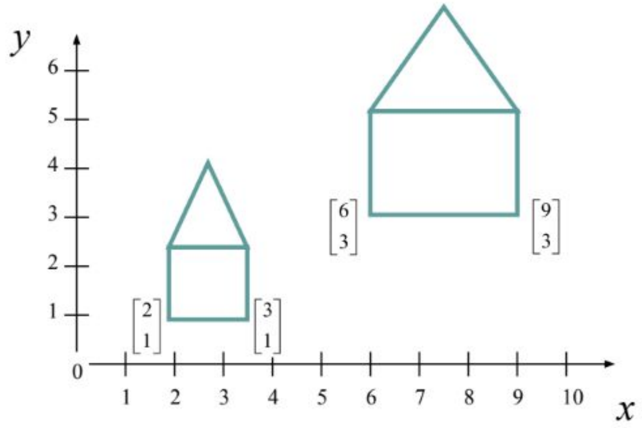
Escala

- Escalas multiplicam todas as coordenadas
- **Atenção:** Objetos podem mudar de lugar ao aplicarmos operações de escala

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

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

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

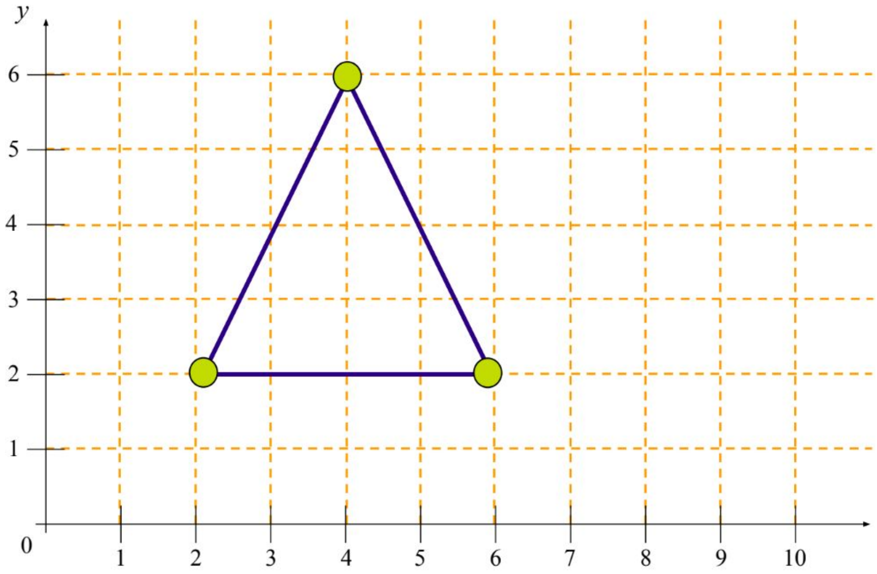


Escala

Exemplo: O que acontecerá ao objeto se aplicarmos uma escala com sx = 2 e sy = 2.

Escala

Exemplo: O que acontecerá ao objeto se aplicarmos uma escala com sx = 2 e sy = 2.

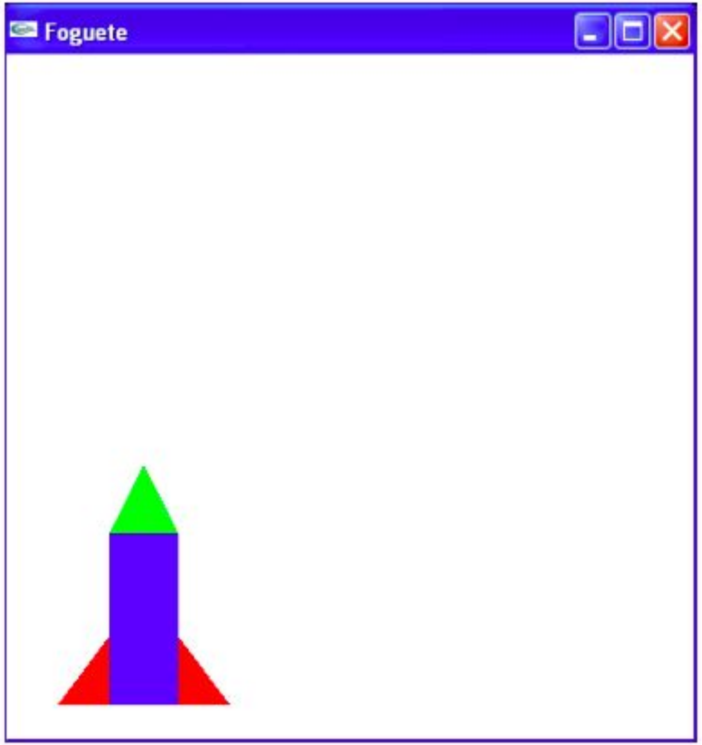


Escala

```
void DesenhaFoguete(void)
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    glScalef(0.5f, 0.5f, 1.0f);
    Bico();
    Corpo();
    AsaEsquerda();
    AsaDireita();
    glFlush();
}
```

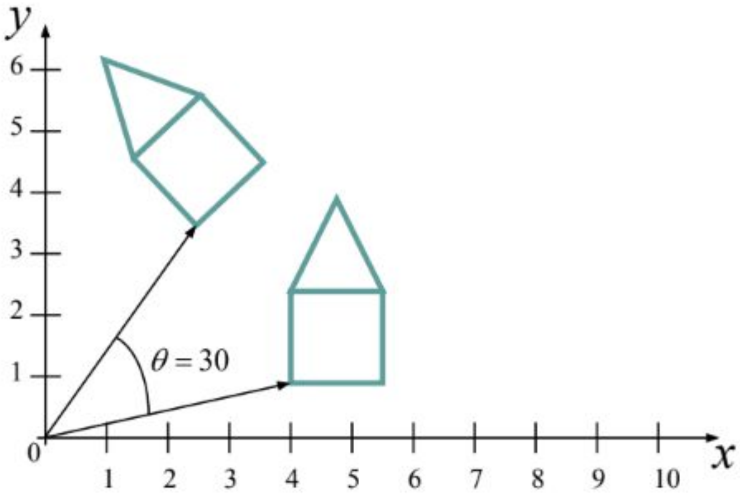
Escala

```
void DesenhaFoguete(void)
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    glScalef(0.5f, 0.5f, 1.0f);
    Bico();
    Corpo();
    AsaEsquerda();
    AsaDireita();
    glFlush();
}
```



Rotação

- Rotaciona todas as coordenadas por um ângulo específico
- Pontos sempre serão rotacionados em relação à origem



Rotação

- A rotação de um ponto em relação a origem teria a seguinte expressão:

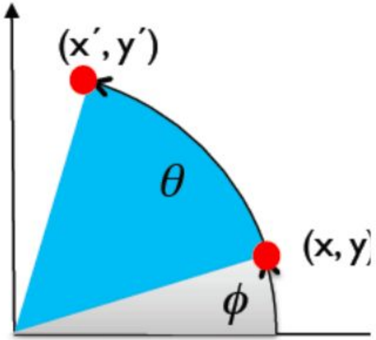
$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\theta \times x - \sin\theta \times y \\ \sin\theta \times x + \cos\theta \times y \end{bmatrix}$$

Rotação

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

Lembra como chegamos a essa fórmula?

$$\begin{cases} x = r\cos\phi \\ y = r\sin\phi \end{cases}$$



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

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

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

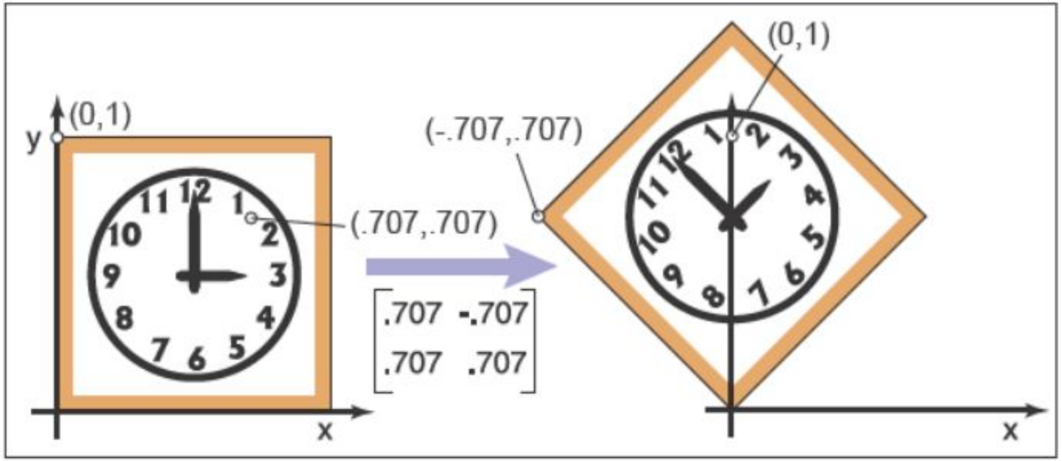
$$\begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \end{cases}$$

Rotação

$$\text{rotate}(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$$

Exemplo:

$$\begin{bmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{bmatrix} = \begin{bmatrix} 0.707 & -0.707 \\ 0.707 & 0.707 \end{bmatrix}$$

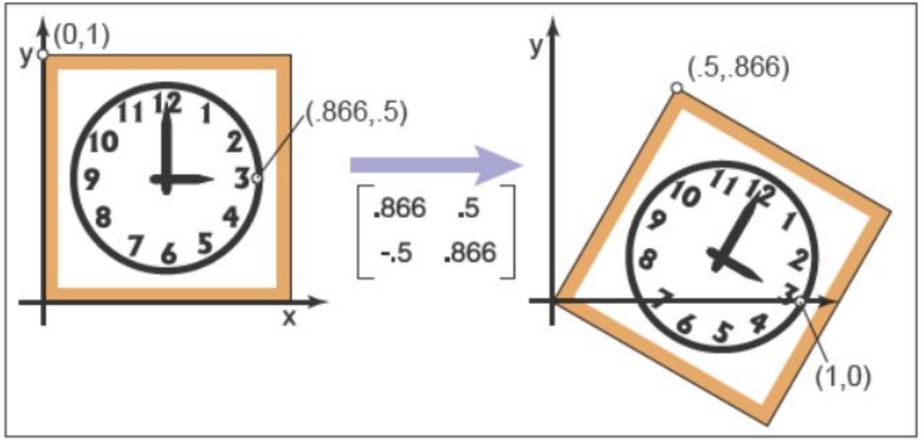


Rotação

$$\text{rotate}(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$$

Exemplo:

$$\begin{bmatrix} \cos\frac{-\pi}{6} & -\sin\frac{-\pi}{6} \\ \sin\frac{-\pi}{6} & \cos\frac{-\pi}{6} \end{bmatrix} = \begin{bmatrix} 0.866 & 0.5 \\ -0.5 & 0.866 \end{bmatrix}$$



Rotação

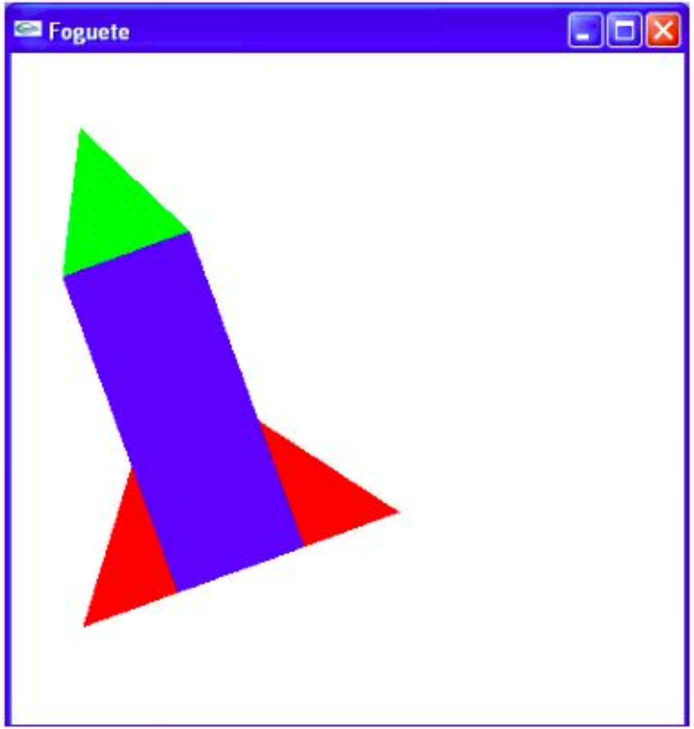
- Atenção! Muitas linguagens (como C e C++) já possuem funções trigonométricas implementadas. Nestas funções, normalmente, o ângulo a ser passado como parâmetro deve estar em radianos e não em graus. Se necessário, para converter de graus (g) para radianos (r), utilize a seguinte fórmula:

$$r = \frac{g \times \pi}{180}$$

- Em OpenGL as funções de rotação utilizam graus. Neste caso, não é necessário converter.

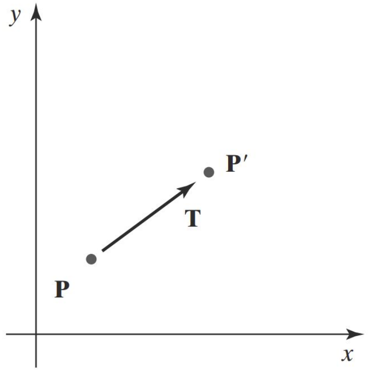
Rotação

```
void DesenhaFoguete(void)
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    glRotatef(20, 0.0f, 0.0f, 1.0f);
    Bico();
    Corpo();
    AsaEsquerda();
    AsaDireita();
    glFlush();
}
```



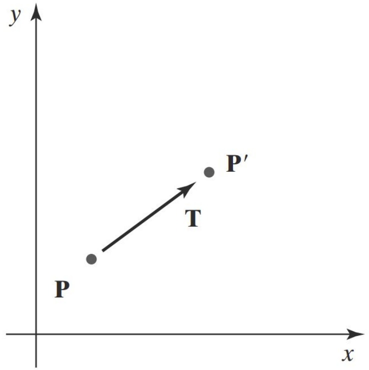
Translação

- Move um ponto de uma posição para outra.



Translação

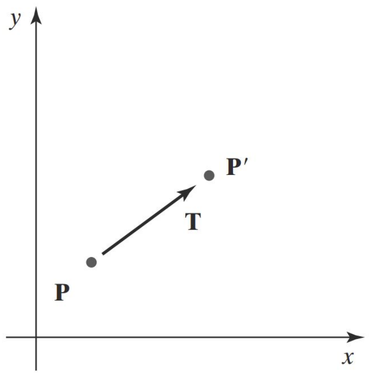
- Move um ponto de uma posição para outra.



$$\boxed{x' = x + t_x, \qquad y' = y + t_y}$$

Translação

- Move um ponto de uma posição para outra.



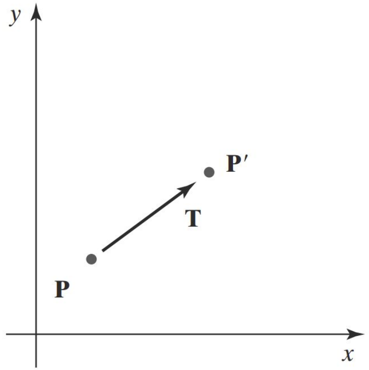
$$x' = x + t_x, \qquad y' = y + t_y$$

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

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

Translação

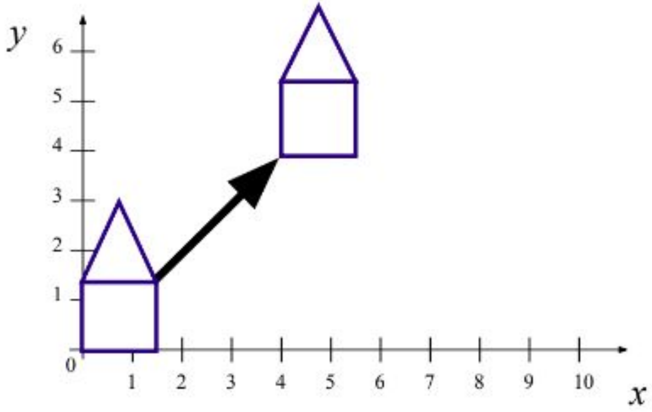
- Move um ponto de uma posição para outra.



$$x' = x + t_x, \qquad y' = y + t_y$$

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

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



Translação

Exemplo: O que acontecerá ao objeto se aplicarmos uma translação com tx = 5 e ty = 3?

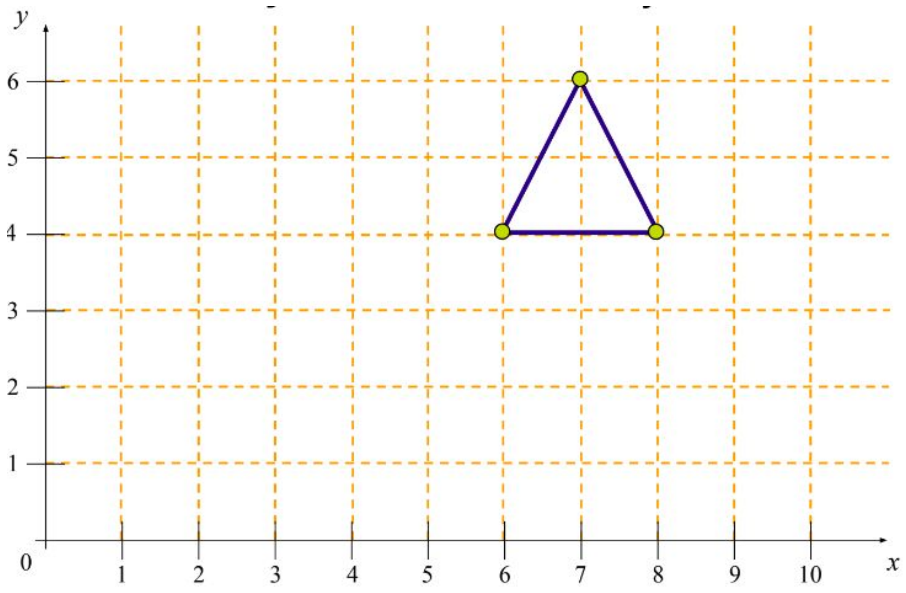
Translação

Exemplo: O que acontecerá ao objeto se aplicarmos uma translação com tx = 5 e ty = 3?

$$x' = x + t_x, \qquad y' = y + t_y$$

Translação

Exemplo: O que acontecerá ao objeto se aplicarmos uma translação com tx = 5 e ty = 3?



$$x' = x + t_x, \qquad y' = y + t_y$$

Translação

- Encontre uma matriz 2x2 com a qual seja possível representar a operação de translação

$$\begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x + dx \\ y + dy \end{bmatrix}$$

Translação

- Encontre uma matriz 2x2 com a qual seja possível representar a operação de translação

$$\begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x + dx \\ y + dy \end{bmatrix}$$

Não é possível representar matrizes 2 x 2 para resolver esse problema.

Translação

- Encontre uma matriz 2x2 com a qual seja possível representar a operação de translação

$$\begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x + dx \\ y + dy \end{bmatrix}$$

Não é possível representar matrizes 2 x 2 para resolver esse problema.

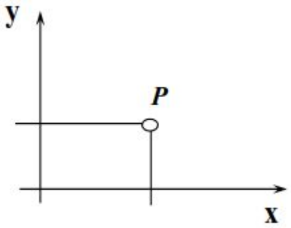
Solução: Coordenadas Homogêneas

Coordenadas Homogeneas

- Um ponto (x, y) pode ser reescrito em coordenadas homogêneas como (x_h, y_h, w)
- O parâmetro h é um valor diferente de zero tal que:

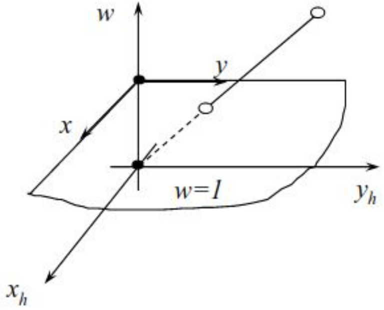
$$x = x_h/w \qquad w>0$$
$$y = y_h/w$$

- Deste modo, o ponto cartesiano (x, y) corresponde à uma infinidade de triplas (x_h, y_h, w), incluindo o caso particular de (x, y, 1)



$$P = \begin{pmatrix} x \\ y \end{pmatrix} \triangleq \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \triangleq \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} x_h \\ y_h \\ w \end{bmatrix}$$

$$\boxed{\begin{array}{ll} x = x_h/w & \\ & w>0 \\ y = y_h/w & \end{array}}$$



Ex.:

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix} \triangleq \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} \triangleq \begin{bmatrix} 6 \\ 4 \\ 2 \end{bmatrix} = \begin{bmatrix} 9 \\ 6 \\ 3 \end{bmatrix}$$

Representando transformações

- Podemos representar as transformações através de matrizes.
- Esta forma de representação tem como vantagem básica permitir que em uma única matriz represente a combinação de várias transformações.

Transformações Geométricas - Coordenadas Homogeneas

Escala

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

Translação

$$\begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + dx \\ y + dy \\ 1 \end{bmatrix}$$

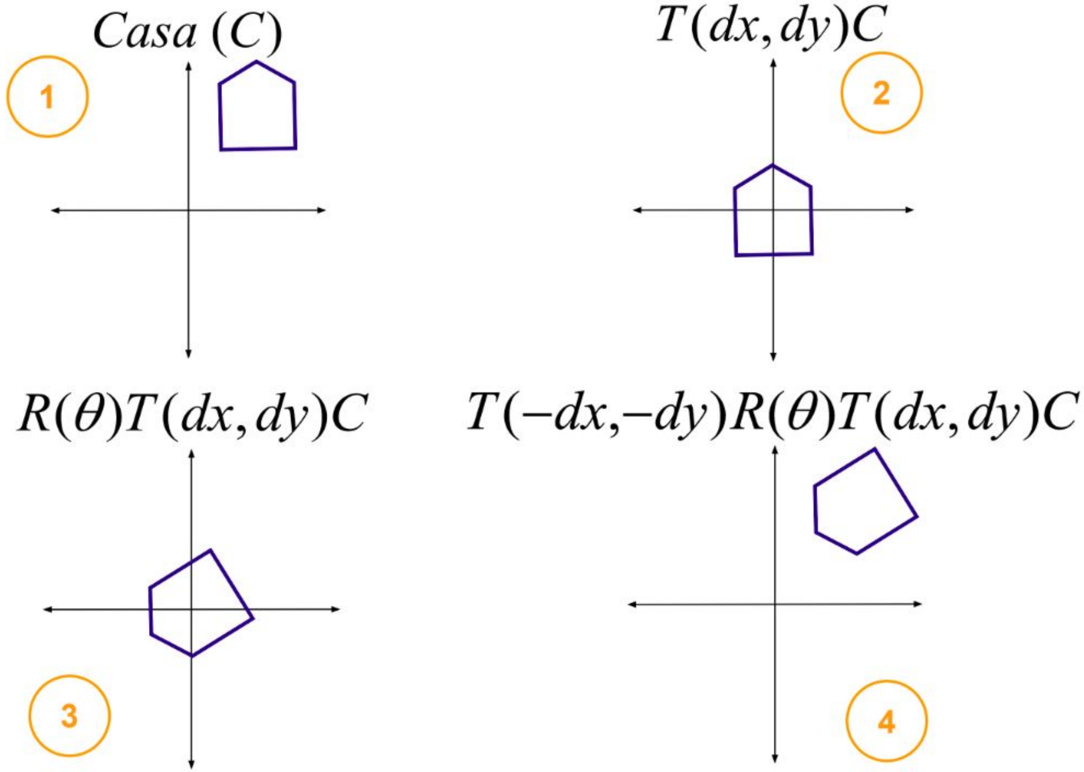
Rotação

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

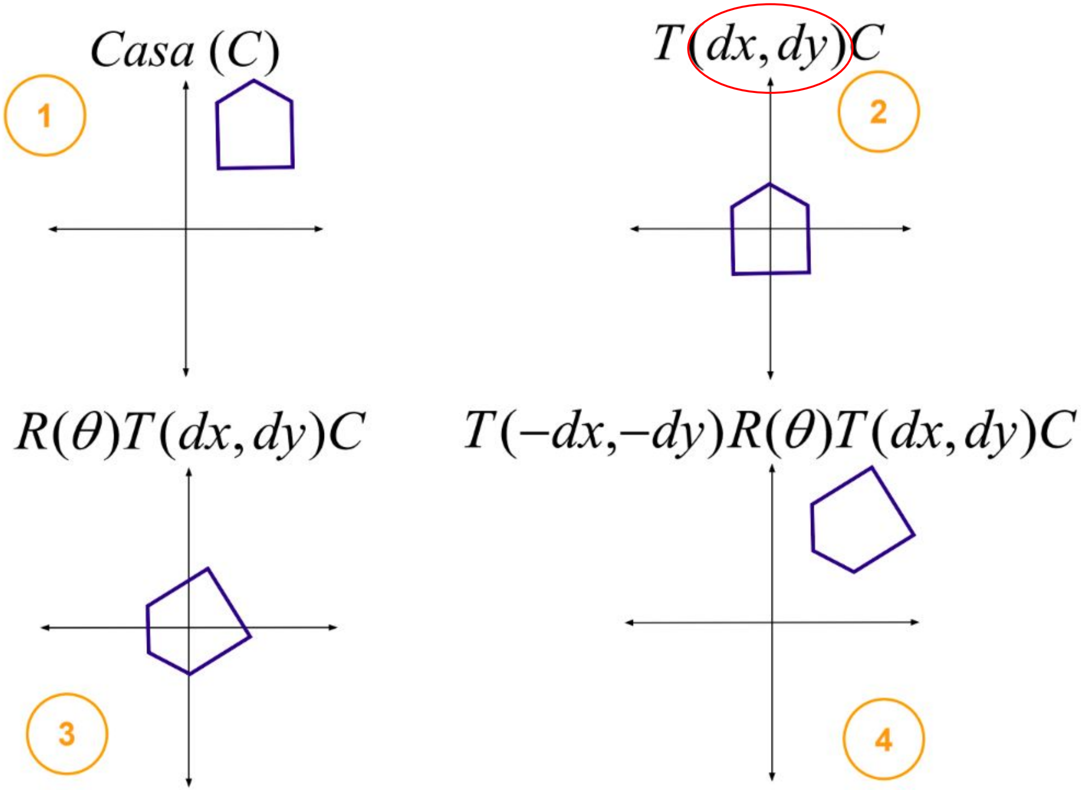
Combinando Transformações

- Rotacionar um polígono em volta de um ponto que não seja a origem:
  - Translade o centro do polígono para a origem
  - Rotacione normalmente em relação à origem
  - Faça a translação inversa

Combinando Transformações

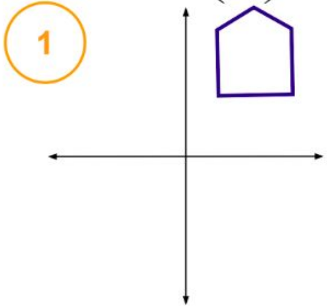


Combinando Transformações



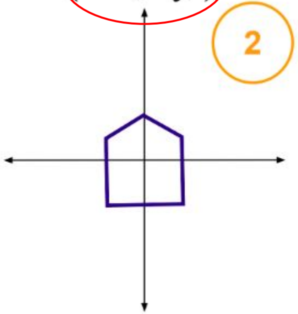
Combinando Transformações

$Casa\ (C)$

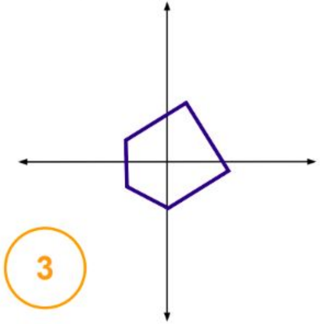


T(-2,-4)

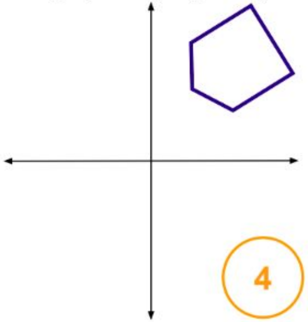
$T(dx,dy)C$



$R(\theta)T(dx,dy)C$

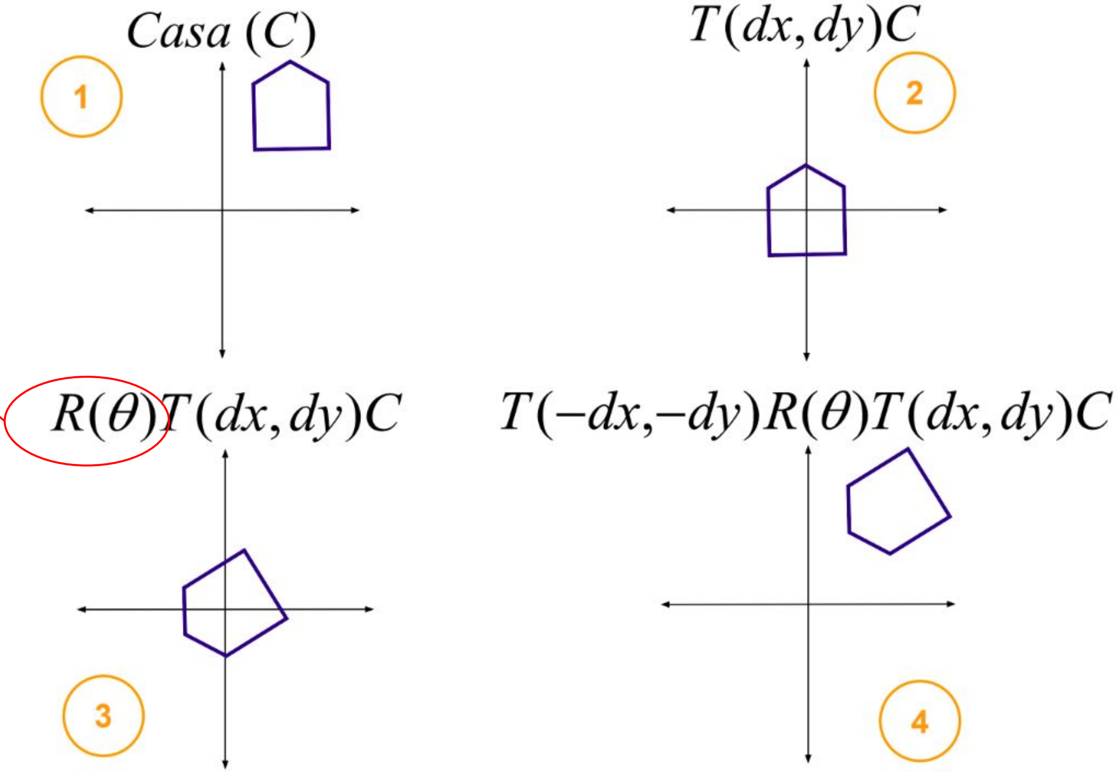


$T(-dx,-dy)R(\theta)T(dx,dy)C$

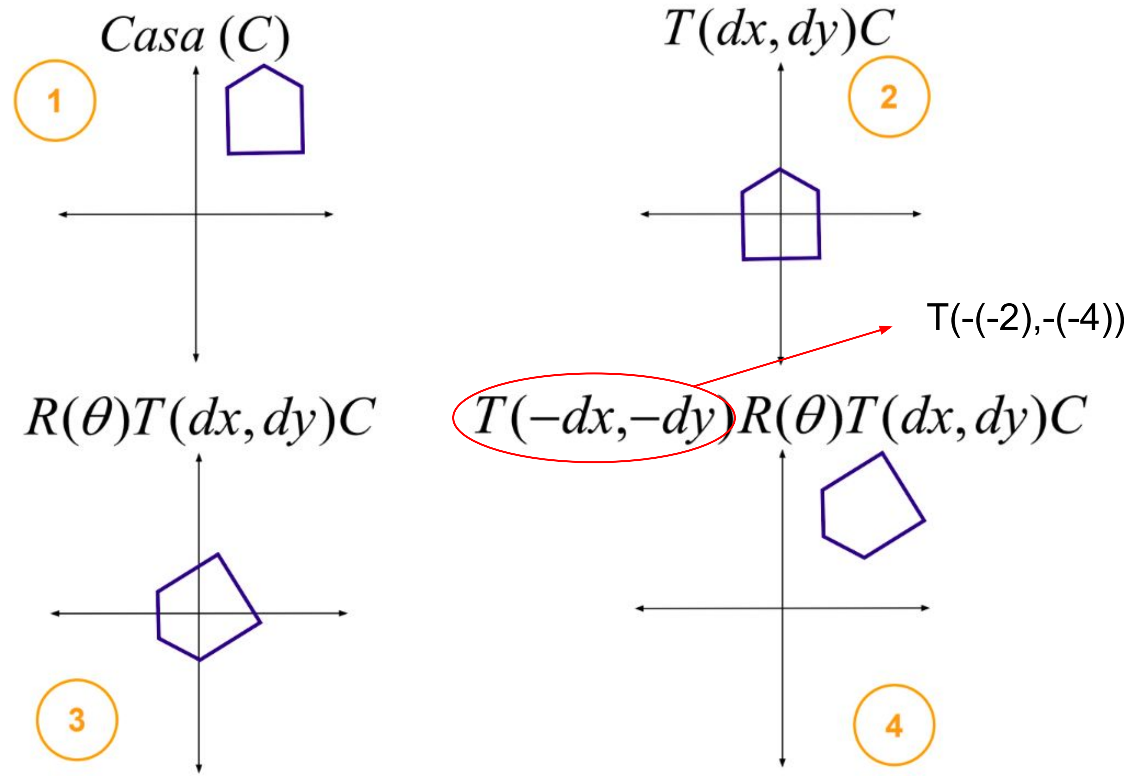


Combinando Transformações

R(135)



Combinando Transformações



Combinando Transformações

As três transformações seriam combinadas da seguinte forma

$$\begin{bmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$v' = T(-dx, -dy)R(\theta)T(dx, dy)v$$

Transformações Geométricas com OpenGL

Os principais comandos em OpenGL para realizar as transformações geométricas são:

```
glTranslatef(dx, dy, dz);

glScalef(sx, sy, sz);

glRotatef(ângulo, x, y, z);
```

Transformações Geométricas com OpenGL

Os principais comandos em OpenGL para realizar as transformações geométricas são:

```
glTranslatef(dx, dy, dz);

glScalef(sx, sy, sz);

glRotatef(ângulo, x, y, z);
```

- Todas as operações da OpenGL são 3D, sendo o 2D um caso particular delas. Por exemplo, para as transformações os valores referentes à coordenada z devem ser considerados como listado abaixo.

```
glTranslated(tx, ty, 0);
glRotated(theta, 0.0, 0.0, 1.0);
glScaled(sx, sy, 1.0);
```

Transformações Geométricas com OpenGL

- Todas as transformações da OpenGL consideram a origem como referência. Antes de fazer rotação ou escala é necessário transladar para a origem, efetuar as transformações e transladar de volta para a posição inicial.

```
...
glMatrixMode(GL_MODELVIEW);
...
glTranslated(tx, ty, 0);
glRotated(theta, 0.0, 0.0, 1.0);
glTranslated(-tx, -ty, 0);
...
```

Transformações Geométricas com OpenGL

Matrizes no OpenGL

Escala

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

Translação

$$\begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + dx \\ y + dy \\ 1 \end{bmatrix}$$

Rotação

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

Transformações Geométricas com OpenGL

Matrizes do OpenGL

- Todos os comandos de transformação são compostos em uma matriz de transformação.

Transformações Geométricas com OpenGL

Matrizes do OpenGL

- Todos os comandos de transformação são compostos em uma matriz de transformação. **A OpenGL possui 3 tipos de matrizes GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE.**

Transformações Geométricas com OpenGL

Matrizes do OpenGL

- Todos os comandos de transformação são compostos em uma matriz de transformação. A OpenGL possui 3 tipos de matrizes GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE.

A matriz modelview controla as transformações dos vértices dos objetos da cena.

Transformações Geométricas com OpenGL

Matrizes do OpenGL

- Todos os comandos de transformação são compostos em uma matriz de transformação. A OpenGL possui 3 tipos de matrizes GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE.

A matriz de projeção controla como a cena 3-D é projetada em 2-D.

Transformações Geométricas com OpenGL

Matrizes do OpenGL

- Todos os comandos de transformação são compostos em uma matriz de transformação. A OpenGL possui 3 tipos de matrizes GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE.

A matriz de texturas transforma as coordenadas das textura para obter efeitos como projetar e deslocar texturas.

Transformações Geométricas com OpenGL

Matrizes do OpenGL

- Todos os comandos de transformação são compostos em uma matriz de transformação. A OpenGL possui 3 tipos de matrizes GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE.

- O modo de matriz atual é alterado através do comando **glMatrixMode**. Cada novo comando é acumulado, alterando a configuração da matriz atual.

Transformações Geométricas com OpenGL

Matrizes do OpenGL

- Todos os comandos de transformação são compostos em uma matriz de transformação. A OpenGL possui 3 tipos de matrizes GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE.

- O modo de matriz atual é alterado através do comando **glMatrixMode**. Cada novo comando é acumulado, alterando a configuração da matriz atual.

```
...
glMatrixMode(GL_MODELVIEW);
...
glTranslated(tx, ty, 0);
glRotated(theta, 0.0, 0.0, 1.0);
glTranslated(-tx, -ty, 0);
...
```

Transformações Geométricas com OpenGL

Matrizes do OpenGL

- Todos os comandos de transformação são compostos em uma matriz de transformação. A OpenGL possui 3 tipos de matrizes GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE.

- O modo de matriz atual é alterado através do comando **glMatrixMode**. Cada novo comando é acumulado, alterando a configuração da matriz atual.

```
...
glMatrixMode(GL_MODELVIEW);
...
glTranslated(tx, ty, 0);
glRotated(theta, 0.0, 0.0, 1.0);
glTranslated(-tx, -ty, 0);
...
```

$$\begin{bmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$v' = T(-dx,-dy)R(\theta)T(dx,dy)v$$

Transformações Geométricas com OpenGL

Matrizes do OpenGL

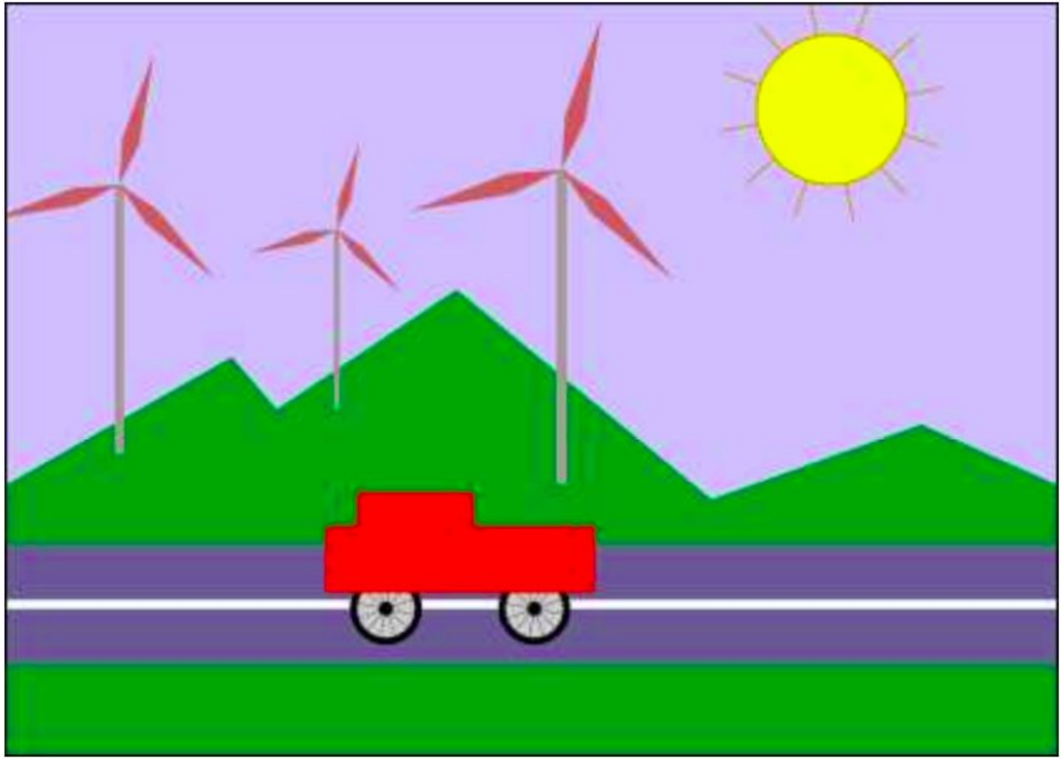
- Todos os comandos de transformação são compostos em uma matriz de transformação. A OpenGL possui 3 tipos de matrizes GL_MODELVIEW, GL_PROJECTION, **GL_TEXTURE**.

- O modo de matriz atual é alterado através do comando **glMatrixMode**. Cada novo comando é acumulado, alterando a configuração da matriz atual.

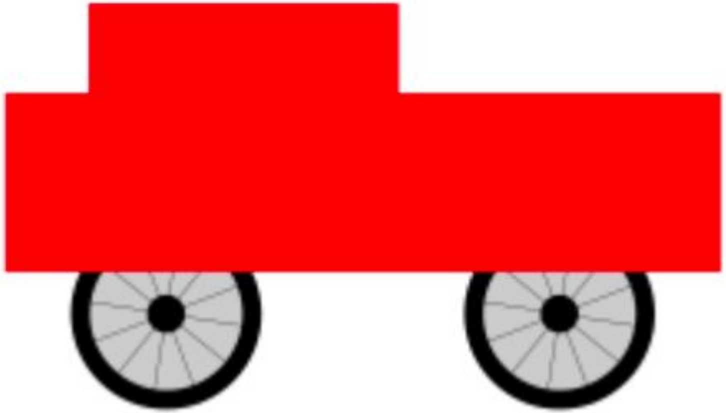
- Ao especificar um novo vértice, a sua posição é calculada aplicando-se a matriz de transformação corrente às suas coordenadas.

Modelagem Hierárquicas

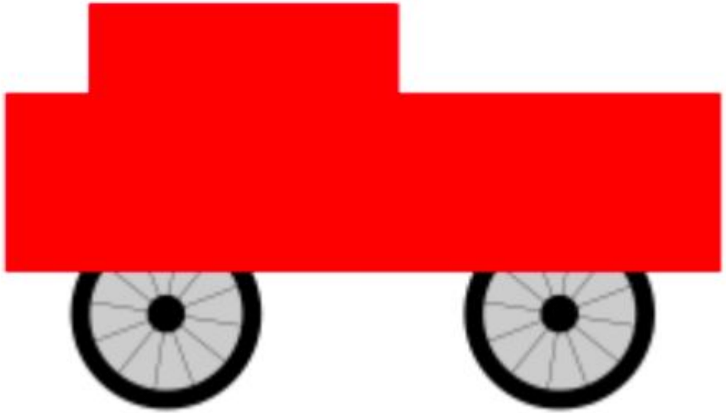
Modelagem Hierárquicas



Modelagem Hierárquicas

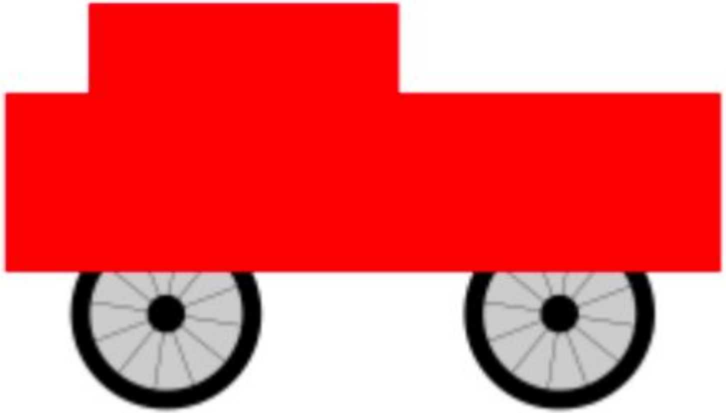


Modelagem Hierárquicas

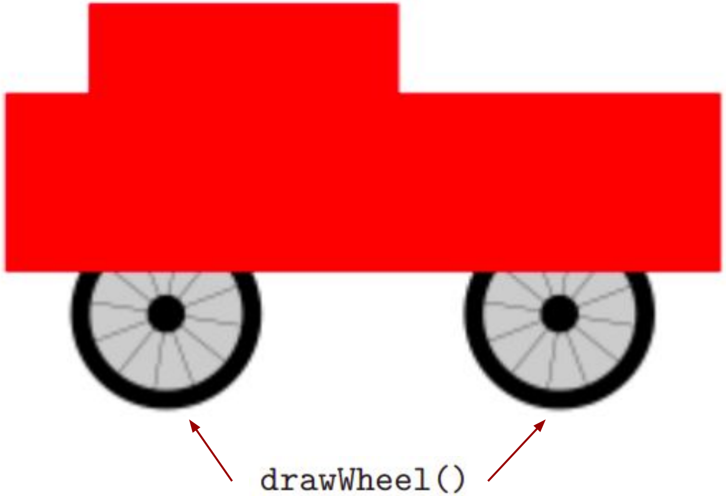


```
saveTransform()
translate(dx,dy) // move object into position
rotate(r)        // set the orientation of the object
scale(sx,sy)     // set the size of the object
     .
     .  // draw the object, using its natural coordinates
     .
restoreTransform()
```

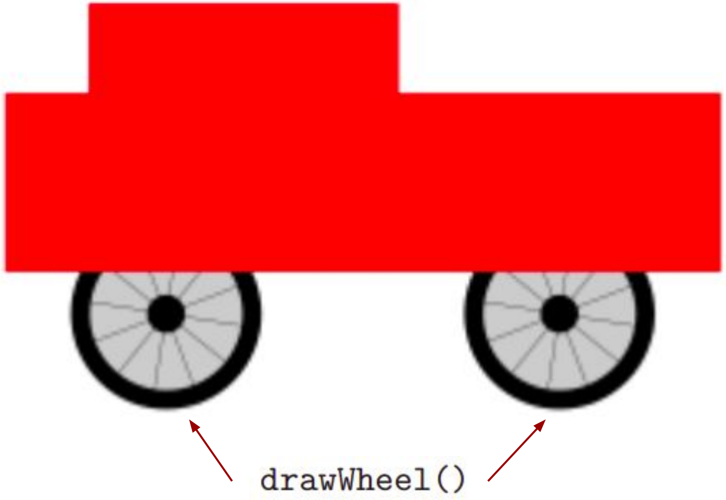
Modelagem Hierárquicas



Modelagem Hierárquicas



Modelagem Hierárquicas




```
subroutine drawCart() :
    saveTransform()          // save the current transform
    translate(-1.65,-0.1)    // center of first wheel will be at (-1.65,-0.1)
    scale(0.8,0.8)           // scale to reduce radius from 1 to 0.8
    drawWheel()              // draw the first wheel
    restoreTransform()       // restore the saved transform
    saveTransform()          // save it again
    translate(1.5,-0.1)      // center of second wheel will be at (1.5,-0.1)
    scale(0.8,0.8)           // scale to reduce radius from 1 to 0.8
    drawWheel                // draw the second wheel
    restoreTransform()       // restore the transform
    setDrawingColor(RED)     // use red color to draw the rectangles
    fillRectangle(-3, 0, 6, 2)      // draw the body of the cart
    fillRectangle(-2.3, 1, 2.6, 1)  // draw the top of the cart
```

Transformações Geométricas com OpenGL

Pilhas 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 atribuí o valor da matriz identidade à matriz corrente.

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

Transformações Geométricas com OpenGL

Pilhas 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
...
```