Transformações Geométricas 2D: Parte 1

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

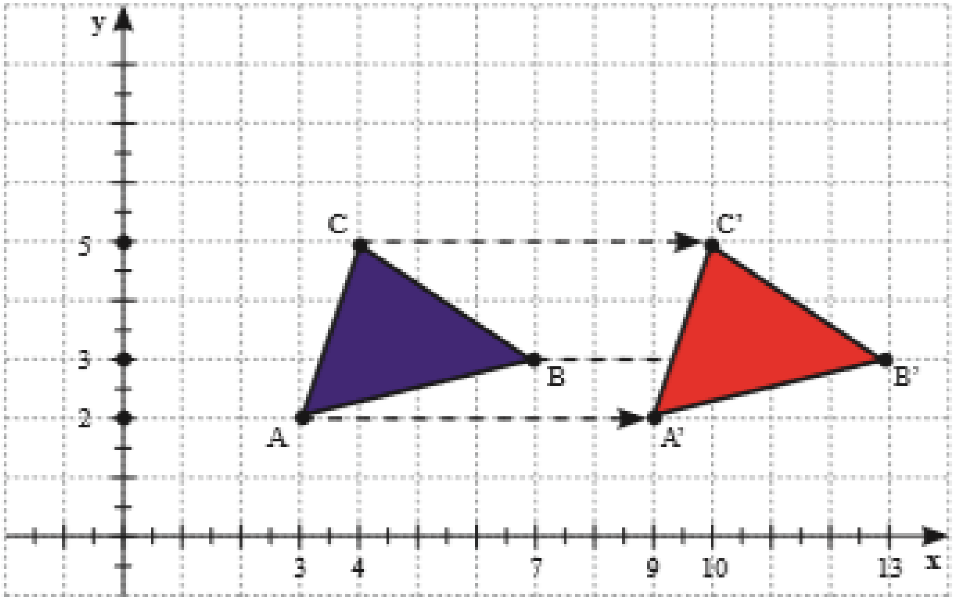
Prof. Walter T. Nakamura
waltertakashi@utfpr.edu.br

Campo Mourão - PR

Baseados nos materiais elaborados pelas professoras Aretha Alencar (UTFPR), Rosane Minghim (USP) e Pelo professor Aldo Von Wangenheim (UFSC)

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



Translação

Translação

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

$$x' = x + t_x$$

$$y' = y + t_y$$

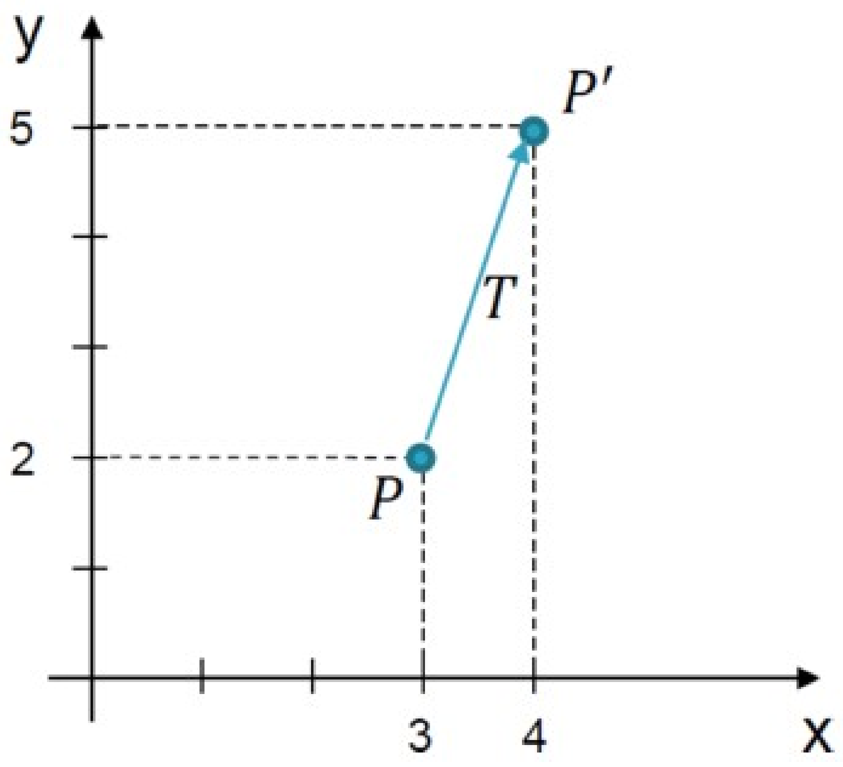
- Usando notação matricial, uma translação 2D pode ser descrita como:

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

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

Translação

- Translação do ponto $P = (3, 2)$ com offsets $t_x = 1$ e $t_y = 3$:



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \end{bmatrix}$$

$$P' = (4, 5)$$

Translação em OpenGL

- Gere a função `translatePolygon()` para fazer a translação de um objeto qualquer de 'n' vértices considerando a assinatura abaixo:

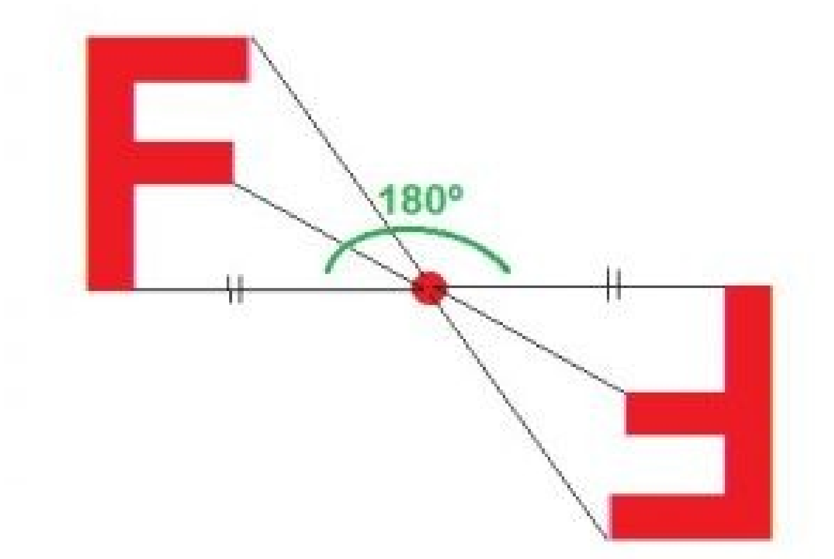
```
class wcPt2D {
    public: GLfloat x, y;
};
```

```
void translatePolygon(wcPt2D * verts, GLint nVerts,
                      GLfloat tx, GLfloat ty)
```

Translação em OpenGL

```
void translatePolygon (wcPt2D * verts, GLint nVerts, GLfloat tx, GLfloat ty) {
    GLint k;

    for (k = 0; k < nVerts; k++) {
        verts[k].x += tx;
        verts[k].y += ty;
        }

    glBegin (GL_POLYGON);
        for (k = 0; k < nVerts; k++)
            glVertex2f(verts[k].x, verts[k].y);
    glEnd();
}
```

```
void display(void){
    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_TRIANGLES);
        glVertex2i(50, 10);
        glVertex2i(100, 10);
        glVertex2i(75, 60);
    glEnd();

    wcPt2D verts[3] = {{50,10}, {100,10}, {75,60}};
    translatePolygon(verts, 3, 20.0, 20.0);
    glFlush();
}
```



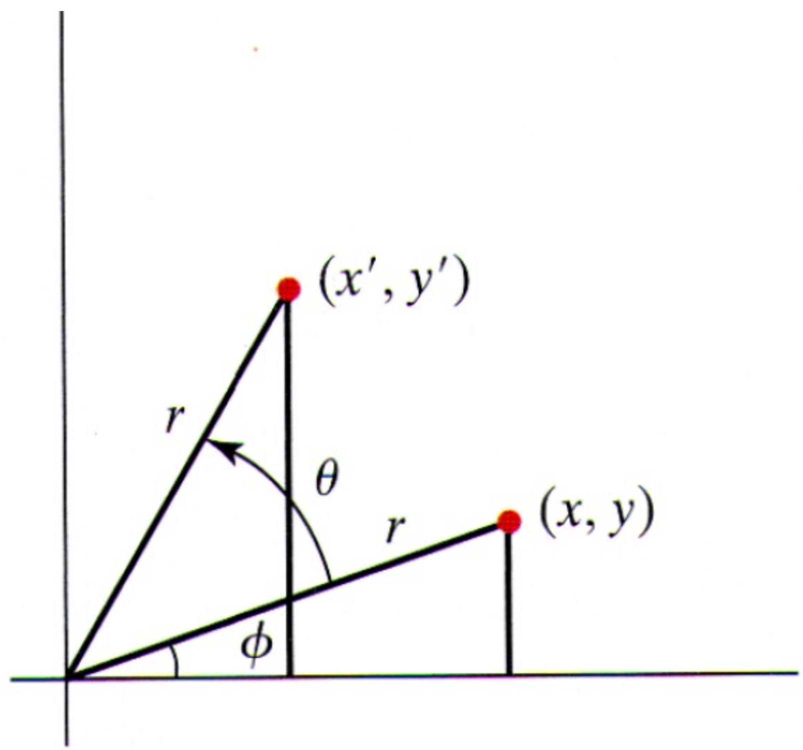
Rotação

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 - Dedução

- Para simplificar considera-se que o **ponto de rotação** está na <u>origem</u> do sistema de coordenadas:
  - O raio r é constante, φ é o ângulo original de $P = (x, y)$ e θ é o ângulo de rotação.



Rotação - Dedução

- Sabendo que

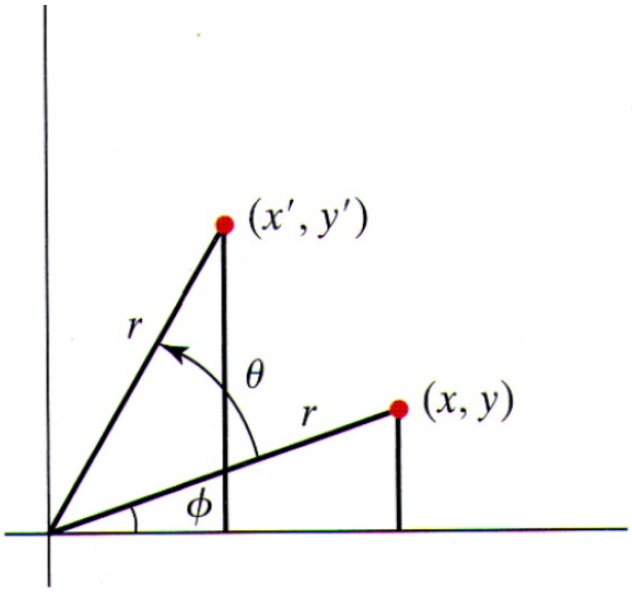
$$\cos(\varphi+\theta)=\frac{cateto\ adjacente}{hipotenusa}=\frac{x'}{r}\Rightarrow x'=r\cdot\cos(\varphi+\theta)$$

$$sen(\varphi+\theta)=\frac{cateto\ oposto}{hipotenusa}=\frac{y'}{r}\Rightarrow y'=r\cdot sen(\varphi+\theta)$$

- como (identidade trigonométrica)

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

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



- então por substituição

$$x'=r\cdot\cos(\varphi)\cdot\cos(\theta)-r\cdot sen(\varphi)\cdot sen(\theta)$$

$$y'=r\cdot\cos(\varphi)\cdot sen(\theta)+r\cdot sen(\varphi)\cdot\cos(\theta)$$

Rotação - Dedução

- Lembrando que o ponto P = (x,y) no sistema cartesiano também pode ser descrito por meio de coordenadas polares como:

$$x = r \cdot \cos(\varphi) \qquad\qquad y = r \cdot sen(\varphi)$$

- então por substituição temos a **equação final da rotação**:

$$x' = x \cdot \cos(\theta) - y \cdot sen(\theta)$$
$$y' = x \cdot sen(\theta) + y \cdot \cos(\theta)$$

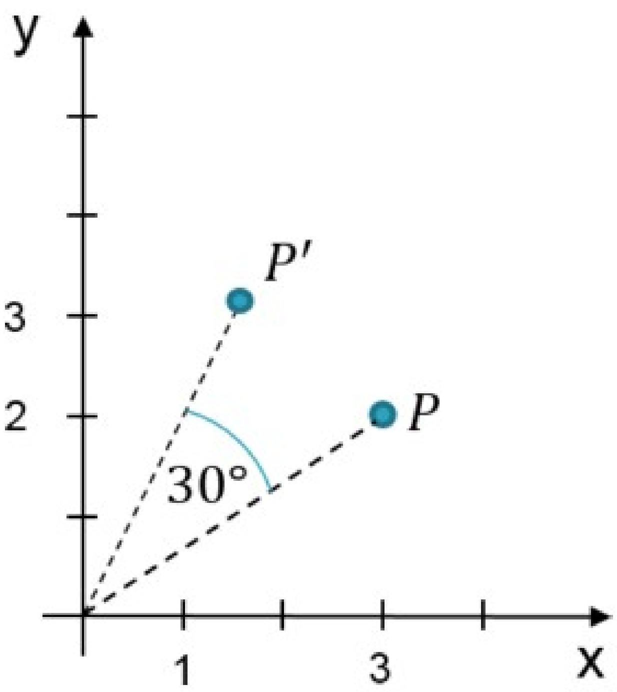
- Escrevendo na **forma matricial** temos:

$$P' = R \cdot P$$

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

Rotação - Exemplo

- Rotação do ponto $P = (3, 2)$ com ângulo de rotação $\theta = 30^{o}$ e ponto de rotação na origem do sistema de coordenadas:



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(30°) & -\operatorname{sen}(30°) \\ \operatorname{sen}(30°) & \cos(30°) \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 3 \cdot \cos(30°) - 2 \cdot \operatorname{sen}(30°) \\ 3 \cdot \operatorname{sen}(30°) + 2 \cdot \cos(30°) \end{bmatrix}$$

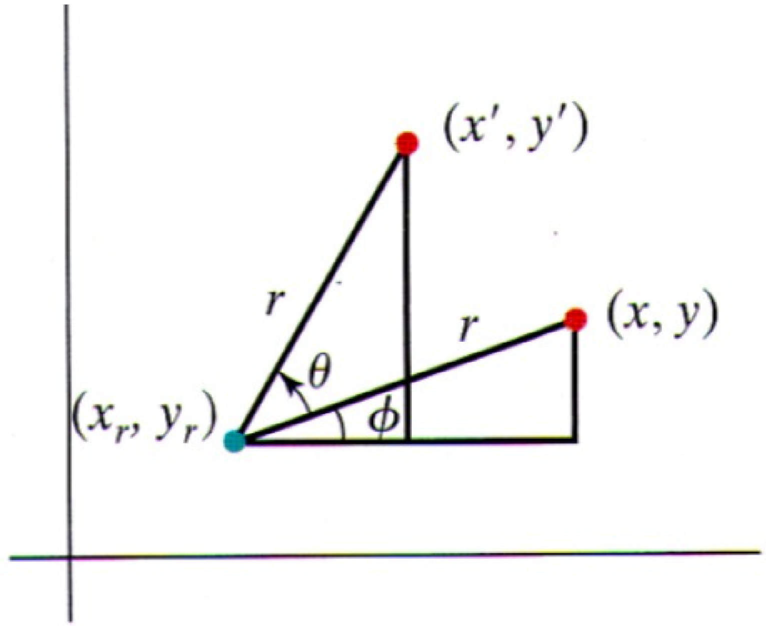
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 3 \cdot \sqrt{3}/2 - 2 \cdot 1/2 \\ 3 \cdot 1/2 + 2 \cdot \sqrt{3}/2 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1.598076 \\ 3.232051 \end{bmatrix}$$

- **Dicas:** $\cos(30^{o}) = \sqrt{3/2}$ e $sen(30^{o}) = 1/2$

Rotação

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



- Poderíamos deduzir a rotação em torno de um **ponto arbitrário** da mesma forma que deduzimos a rotação em torno da origem do sistema de coordenadas
- No entanto, existe uma forma **melhor e mais fácil** de fazer isso que será apresentada mais adiante.

Rotação em OpenGL

- Gere a função `rotatePolygon()` para fazer a rotação de um objeto qualquer de 'n' vértices considerando a assinatura abaixo:

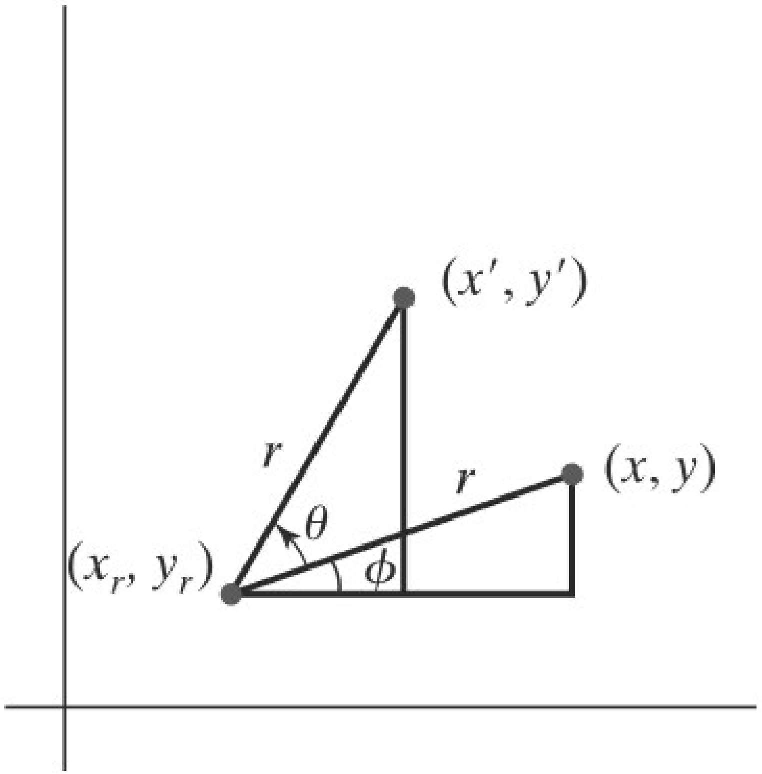
```
class wcPt2D {
    public: GLfloat x, y;
};
```

```
void rotatePolygon(wcPt2D * verts, GLint nVerts, GLdouble theta)
```

Rotação 2D com Ponto de Rotação

- A rotação de um objeto pode ser feita definindo-se um **ponto de rotação** (x_r, y_r) também chamada de pivô
  - Usando as relações trigonométricas apresentadas na figura ao lado é possível gerar as equações de rotação de um ponto sob uma determinada posição x_r e y_r

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

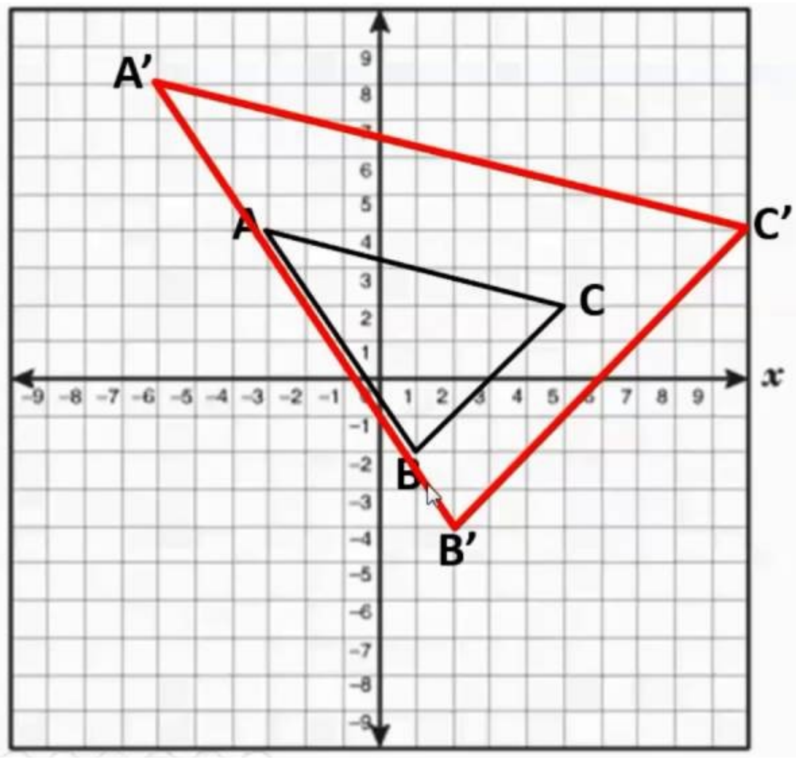


Rotação 2D com Ponto de Rotação

- Adapte a função `rotatePolygon()` criada anteriormente para fazer a rotação de um objeto qualquer de 'n' vértices sob um ponto de rotação (x,y) arbitrário

Transformação de Corpo Rígido

- A rotação e a translação são **transformações de corpo rígido** pois direcionam ou movem um objeto <u>sem deformá-lo</u>
  - Mantém **ângulos** e **distâncias** entre as coordenadas do objeto



Escala

Escala

- Para **alterar o tamanho** de um objeto aplica-se a <u>transformação de escala</u>
  - Multiplica-se as coordenadas de um objeto por fatores de escala:

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

- Na forma matricial:

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

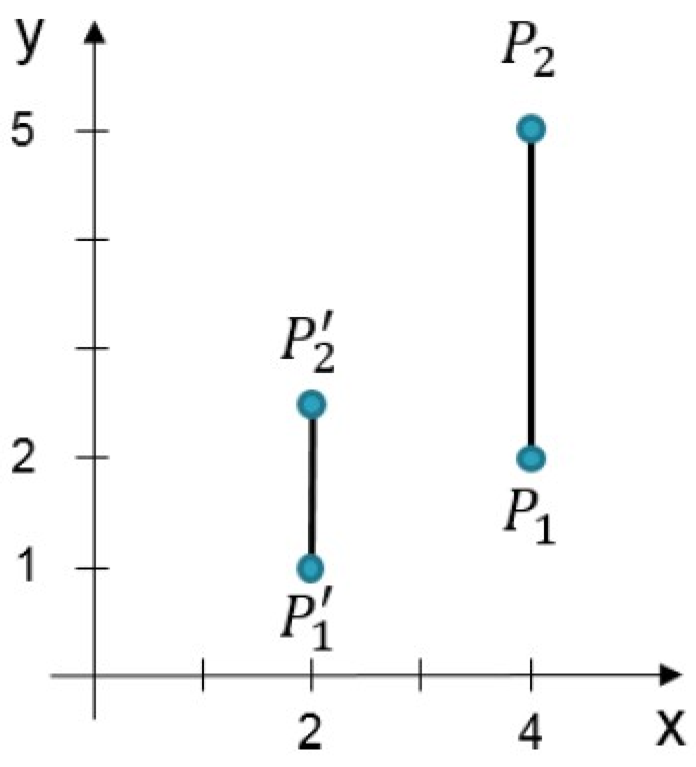
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Escala

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

Escala - Exemplo

- Fazer a escala da reta definida por $P_1 = (x_1, y_1) = (4, 2)$ e $P_2 = (x_2, y_2) = (4, 5)$ com fatores de escala $s_x = s_y = 0.5$:



$$\begin{bmatrix} x1' \\ y1' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x1 \\ y1 \end{bmatrix}$$

$$\begin{bmatrix} x1' \\ y1' \end{bmatrix} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} x1' \\ y1' \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \Rightarrow P1' = (2, 1)$$

$$\begin{bmatrix} x2' \\ y2' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x2 \\ y2 \end{bmatrix}$$

$$\begin{bmatrix} x2' \\ y2' \end{bmatrix} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 4 \\ 5 \end{bmatrix}$$

$$\begin{bmatrix} x2' \\ y2' \end{bmatrix} = \begin{bmatrix} 2 \\ 2.5 \end{bmatrix} \Rightarrow P2' = (2, 2.5)$$

Escala

- Pela formulação definida, o objeto é escalado <u>e</u> movido

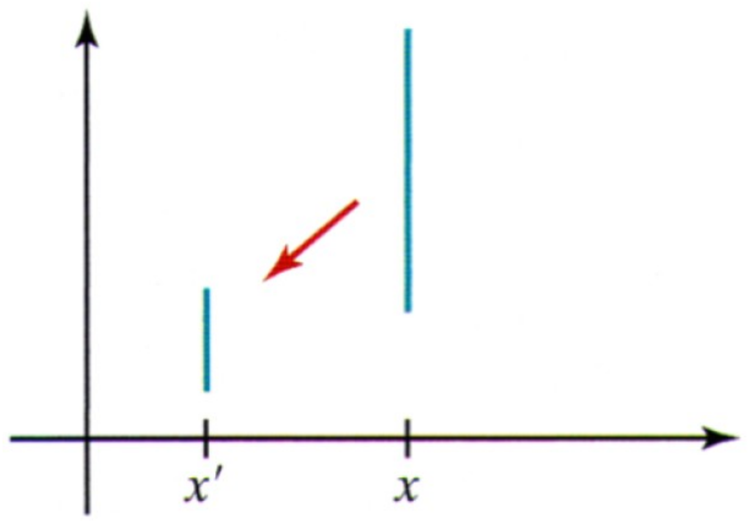


Figura: Escala de uma linha usando sx = sy = 0.5.

- É possível calcular uma forma matricial para fazer a escala com um ponto fixo
  - No entanto, existe uma forma melhor e mais fácil de fazer isso que será apresentada mais adiante.

Escala em OpenGL

- Gere a função `scalePolygon()` para fazer a escala de um objeto qualquer de 'n' vértices considerando a assinatura abaixo:

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

```
void scalePolygon(wcPt2D * verts, GLint nVerts, GLfloat sx,
                        GLfloat sy)
```

Escala 2D com Ponto Fixo

- Podemos controlar a localização de um objeto dimensionado escolhendo uma posição, chamada **ponto fixo** (x_f, y_f)
  - Permanece inalterado após a transformação de dimensionamento
  - Frequentemente é escolhido o centróide, mas qualquer outra posição espacial pode ser selecionada
- Os objetos são **redimensionados** escalando as distâncias entre os pontos do objeto e o ponto fixo
  - Para uma posição de coordenada (x, y), as coordenadas em escala (x', y') são calculadas a partir das seguintes relações:

$$x' - x_f = (x - x_f)s_x, \qquad y' - y_f = (y - y_f)s_y$$

  - Equação final:

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

Escala 2D com Ponto Fixo

- Adapte a função `scalePolygon()` para fazer a escala de um objeto qualquer de 'n' vértices sob um ponto fixo (x_f, y_f)