

Curso de Jogos Digitais

Disciplina de Computação Gráfica

Transformações Geométricas

Aula 04

Professor: André Flores dos Santos

Santa Maria – RS 2025



SUMÁRIO



Transformações Geométricas

- Objetivo:
 - manipular o conteúdo de uma cena;
- Como?
 - pelo movimento da câmera ou dos objetos que estão presentes na cena;
- Tipos de transformações:
 - orientação, tamanho e formato de um objeto;
- Essas mudanças podem ser cumulativas!

Transformações Geométricas

- **Transformações afim:**
 - Mantém o paralelismo de retas.
 - Preservam os ângulos;
- **Principais:**
 - Translação, escala e rotação.
- **Como representar?**
 - Operações algébricas
 - Matrizes.

Operações Algébricas

- Podem se tornar caras computacionalmente;
- Porquê matrizes são mais interessantes?
 - Podem realizar as transformações e **combiná-las de forma mais eficiente.**
 - Mais eficientes no armazenamento de figuras presentes num sistema de coordenadas.

Translação

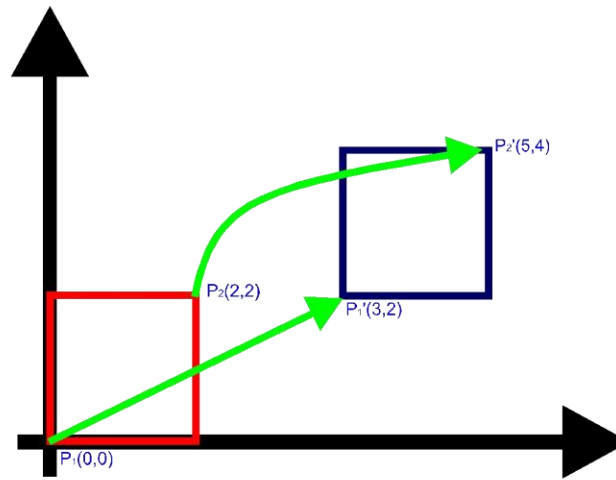
- Define a **posição** de um objeto no universo;
- A partir de coordenadas de um ponto (x_o, y_o) , soma-se a ele os valores de translação T_x e T_y , obtendo-se assim o ponto (x_u, y_u) , que representa o ponto posicionado no universo;
- Para tanto aplicam-se as seguinte equações:

$$x_u = x_o + T_x$$

$$y_u = y_o + T_y$$

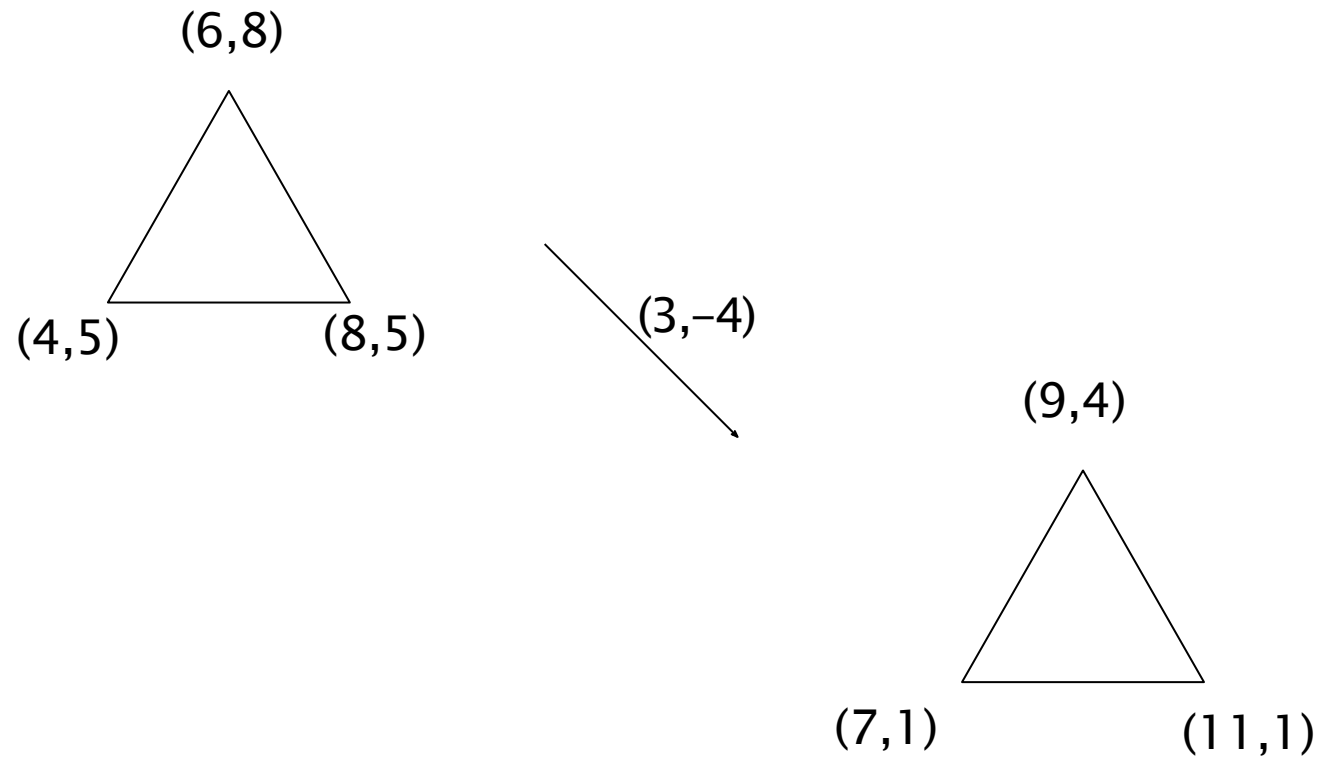
Translação

- Exemplo:
 - $P_1(0,0)$ e $P_2(2,2)$
 - $T_x = 3$ e $T_y = 2$
 - $P_1'(3,2)$ e $P_2'(5,4)$



Translação

□ Exemplo 2:



Translação

- ▢ Também representada na forma matricial através da soma de matrizes:

$$P_o = \begin{bmatrix} x_o \\ y_o \end{bmatrix}$$

$$P_u = \begin{bmatrix} x_u \\ y_u \end{bmatrix}$$

$$T = \begin{bmatrix} T_x \\ T_Y \end{bmatrix}$$

$$P_u = P_o + T$$

$$P_u = \begin{bmatrix} x_o \\ y_o \end{bmatrix} + \begin{bmatrix} T_x \\ T_Y \end{bmatrix}$$

$$P_u = \begin{bmatrix} x_o + T_x \\ y_o + T_Y \end{bmatrix}$$

Translação

Vamos aplicar essa teoria na prática, através de um algoritmo em Python para a operação de translação para os pontos dos exemplos anteriores, [slide 7](#). **Exercício 01.**

```
import numpy as np
import matplotlib.pyplot as plt

# Função para calcular a translação dos pontos
def translacao(pontos, Tx, Ty):
    pontos_translados = []
    for ponto in pontos:
        x_u = ponto[0] + Tx
        y_u = ponto[1] + Ty
        pontos_translados.append((x_u, y_u))
    return pontos_translados

# Pontos originais
p1 = (0, 0)
p2 = (2, 2)

# Vetor de translação
Tx = 3
Ty = 2

# Calcular a translação dos pontos
pontos_translados = translacao([p1, p2], Tx, Ty)
```

```
# Plotar os pontos originais e os pontos translados
plt.plot([p1[0], p2[0]], [p1[1], p2[1]], 'bo-', label='Pontos originais')
plt.plot([ponto[0] for ponto in pontos_translados], [ponto[1] for ponto in pontos_translados],
         'ro-', label='Pontos translados')
#[ponto[0] for ponto in pontos_translados]:
#Isso é uma compreensão de lista, uma construção Python que permite criar listas de maneira mais
compacta.
#O que está entre os colchetes é uma expressão que define como cada elemento da lista será
construído.
#ponto[0] é a expressão usada para cada elemento da lista. Aqui, 'ponto' representa cada elemento
da lista 'pontos_translados', que são as coordenadas dos pontos translados.
#ponto[0] extrai o primeiro elemento de cada tupla dentro da lista pontos_translados, que
corresponde à coordenada x do ponto.
#Então, [ponto[0] for ponto in pontos_translados] cria uma lista contendo apenas as coordenadas
x dos pontos translados. Essa lista é usada para definir os valores do eixo x no gráfico.

#'bo-' especifica o estilo do gráfico: b para azul (blue), 'o' para marcador de círculo e '-' para linha
contínua.
#'ro-':#Isso define o estilo do gráfico para os pontos translados. Aqui, 'r' indica que os pontos
serão vermelhos,
# 'o' especifica que os marcadores serão círculos e '-' indica que uma linha contínua conectará os
pontos.

# Configurações do gráfico
plt.xlabel('X')
plt.ylabel('Y')
plt.title("Translação de pontos no plano cartesiano")
plt.grid(True)
plt.legend()

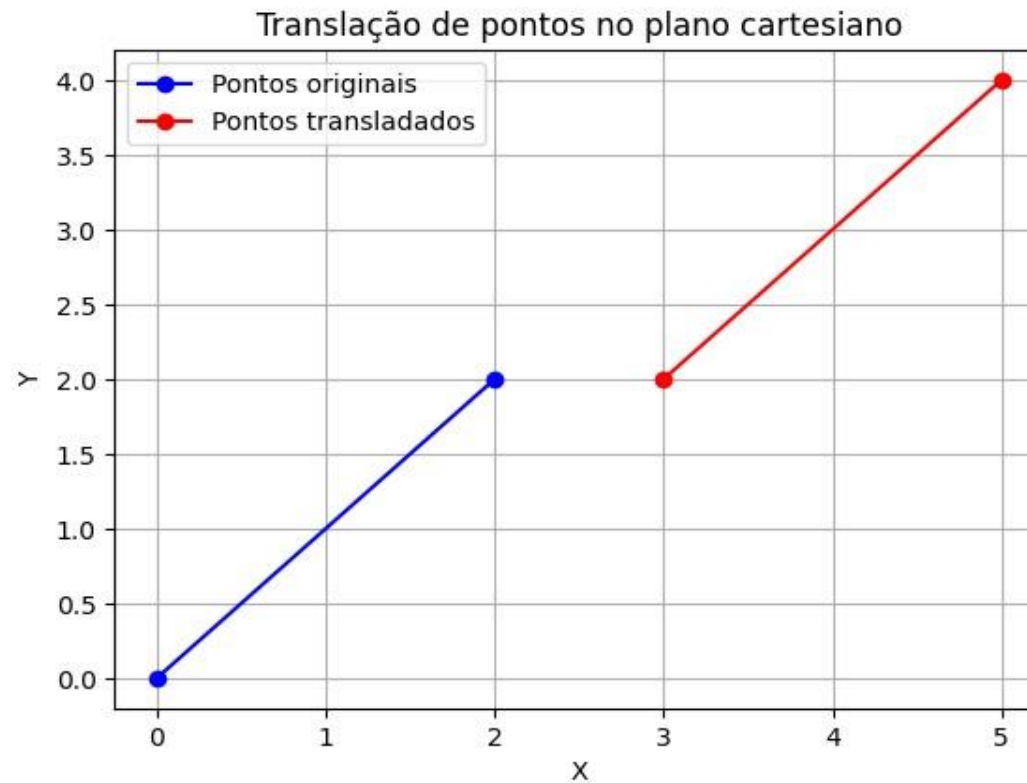
# Mostrar o gráfico
plt.show()
```

https://github.com/andreflores2009/ComputacaoGrfica_2025-01/blob/22f6b629f820397722da34d62ed6340f990e5ba0/Exercicios/Aula04/Ex01_Transformacoes_geometricas_Translacao.py

Translação

Vamos aplicar essa teoria na prática, através de um algoritmo em Python para a operação de translação para os pontos dos exemplos anteriores, [slide 7](#).

Gráfico gerado:



Translação

Vamos aplicar essa teoria na prática, através de um algoritmo em Python para a operação de translação para os pontos dos exemplos anteriores, slide 7 e [slide 8](#). Exercício 02.

Agora complete o algoritmo anterior para o exemplo do Slide 8, que tem 3 pontos, ao invés de 2 apenas.

$p1(6,8)$

$p2(4,5)$

$p3(8,5)$

$(tx, ty) = (3, -4)$

$p1'(? , ?) = (9, 4)$

$p2'(? , ?) = (7, 1)$

$p3'(? , ?) = (11, 1)$

OBS: Todos exercícios devem ser enviados no final da aula, através de um link do seu repositório no github

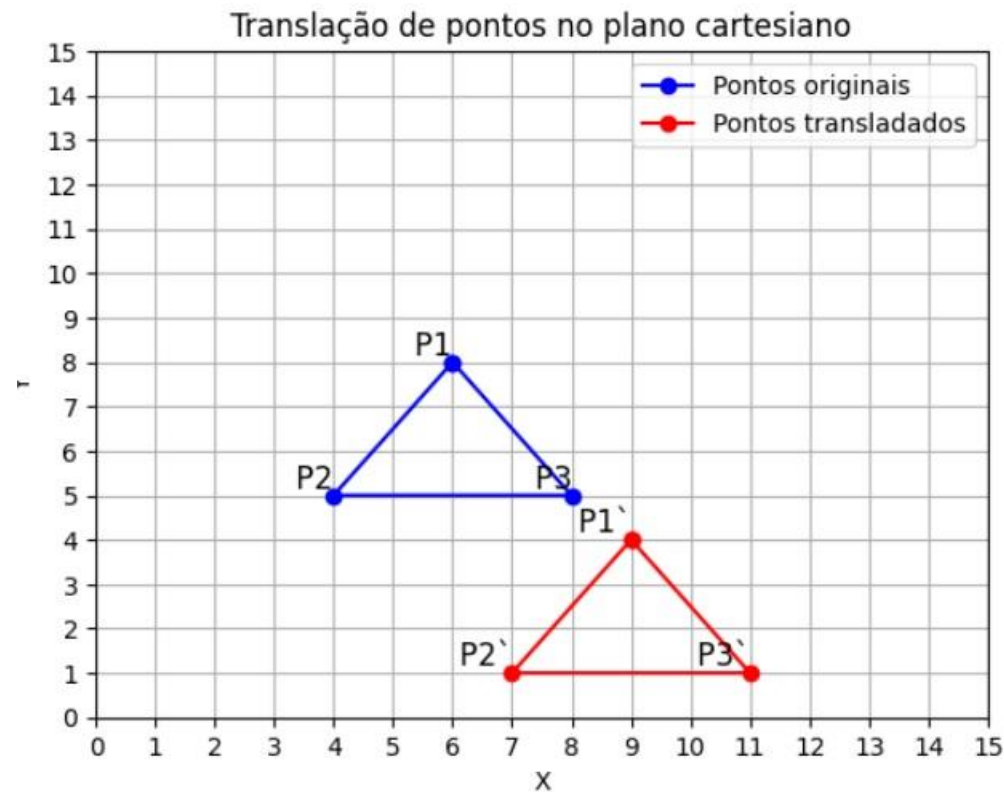
Translação

Vamos aplicar essa teoria na prática, através de um algoritmo em Python para a operação de translação para os pontos dos exemplos anteriores, slide 8 com 3 pontos.

Gráfico gerado:

```
# Define os pontos originais  
p1 = (6, 8) # Primeiro ponto  
p2 = (4, 5) # Segundo ponto  
p3 = (8, 5) # Terceiro ponto  
  
# Define o vetor de translação  
Tx = 3 # Translação no eixo X  
Ty = -4 # Translação no eixo Y
```

```
p1'(?,:) = (9,4)  
p2'(?,:) = (7,1)  
p3'(?,:) = (11,1)
```



Escala

- É a alteração do **tamanho do objeto**.
- Aplicam-se as seguintes equações sobre os pontos do objeto:

$$x_u = x_o * S_x$$

$$y_u = y_o * S_y$$

- Onde S_x e S_y são os fatores de escala em relação aos eixos X e Y ;
- A escala pode mudar a posição do objeto, pois ela sempre usa como referência a origem (0,0).

Escala

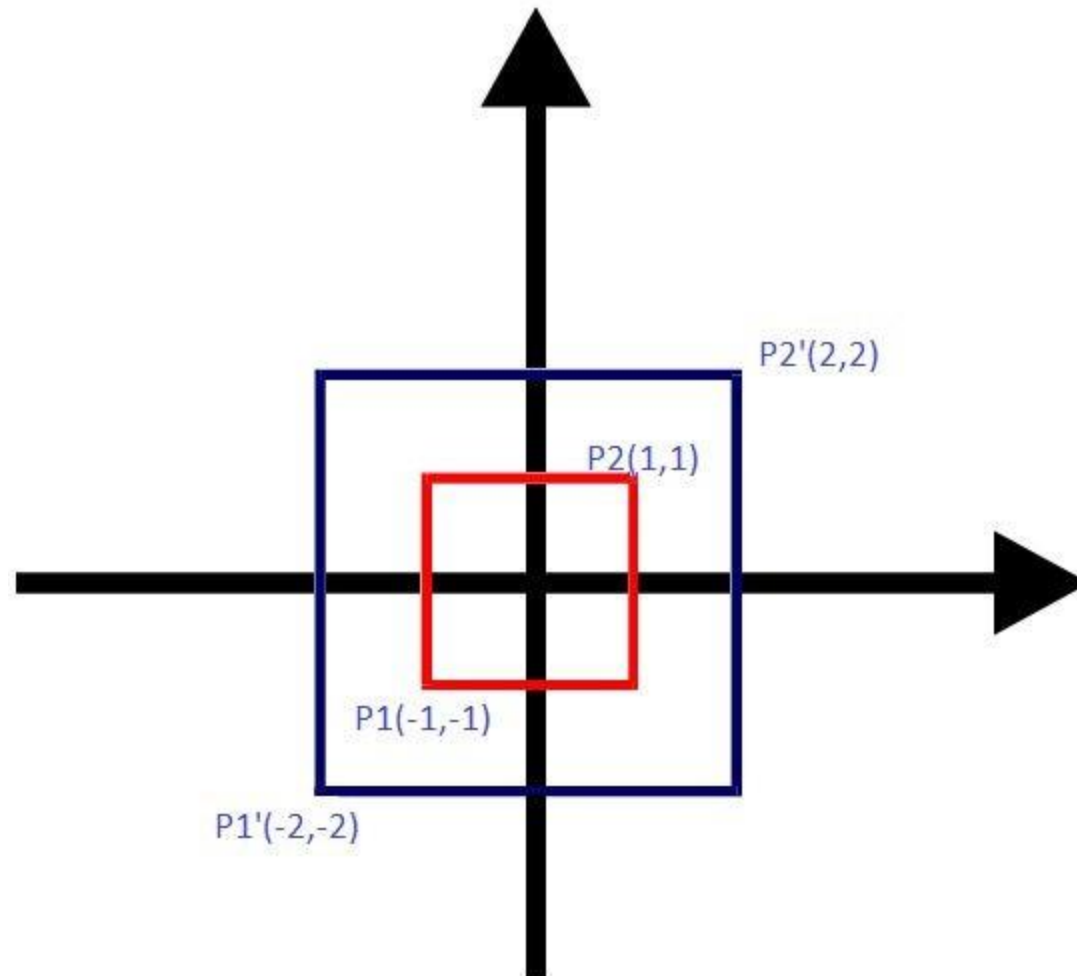
- Observações:

- Se $S > 1$ □ aumenta o tamanho do objeto;
- Se $0 < S < 1$ □ diminui o tamanho do objeto;
- Se $S < 0$ □ inverte/espelha o objeto.

Escala

□ Exemplo 1:

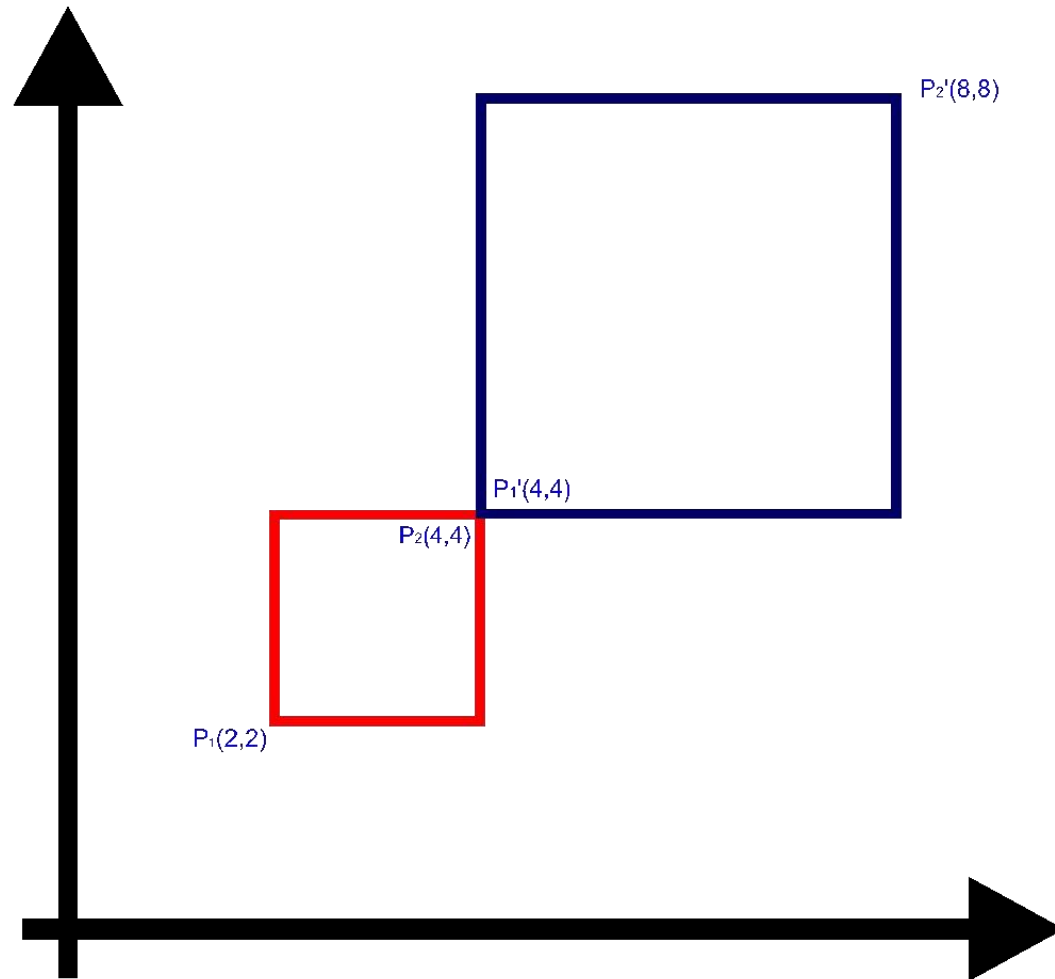
- $P_1(-1,-1)$ e $P_2(1,1)$
- $S_x = 2$ e $S_y = 2$
- $P_1'(-2,-2)$ e $P_2'(2,2)$



Escala

□ Exemplo 2:

- $P_1(2,2)$ e $P_2(4,4)$
- $S_x = 2$ e $S_y = 2$
- $P_1'(4,4)$ e $P_2'(8,8)$



Escala

- ▶ A escala também pode ser representada na forma matricial através de uma multiplicação de matrizes:

$$P_o = \begin{bmatrix} x_o \\ y_o \end{bmatrix}$$

$$P_u = \begin{bmatrix} x_u \\ y_u \end{bmatrix}$$

$$S = \begin{bmatrix} S_x & 0 \\ 0 & S_Y \end{bmatrix}$$

$$P_u = S * P_o$$

$$P_u = \begin{bmatrix} S_x & 0 \\ 0 & S_Y \end{bmatrix} * \begin{bmatrix} x_o \\ y_o \end{bmatrix}$$

$$P_u = \begin{bmatrix} x_o S_x \\ y_o S_Y \end{bmatrix}$$

Escala

Vamos aplicar essa teoria na prática, através de um algoritmo em Python para a operação de escala para os pontos dos exemplos anteriores, [slide 16](#). **Exercício 03.**

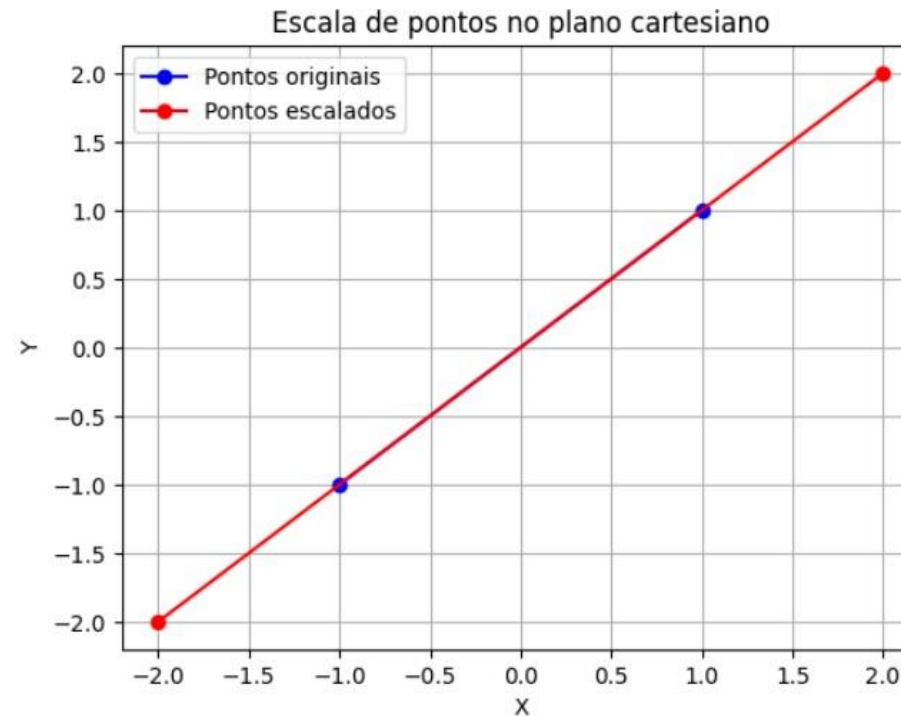
Agora adapte o algoritmo utilizado na translação para a operação de escala! Eles são bem parecidos, serão necessários algumas alterações! Só é preciso plotar os pontos, não precisa aquele quadrado em volta que esta descrito no slide 16.

OBS: Todos exercícios devem ser enviados no final da aula, através de um link do seu repositório no github

Escala

Vamos aplicar essa teoria na prática, através de um algoritmo em Python para a operação de escala para os pontos dos exemplos anteriores, [slide 16](#). **Exercício 03.**

Gráfico gerado:



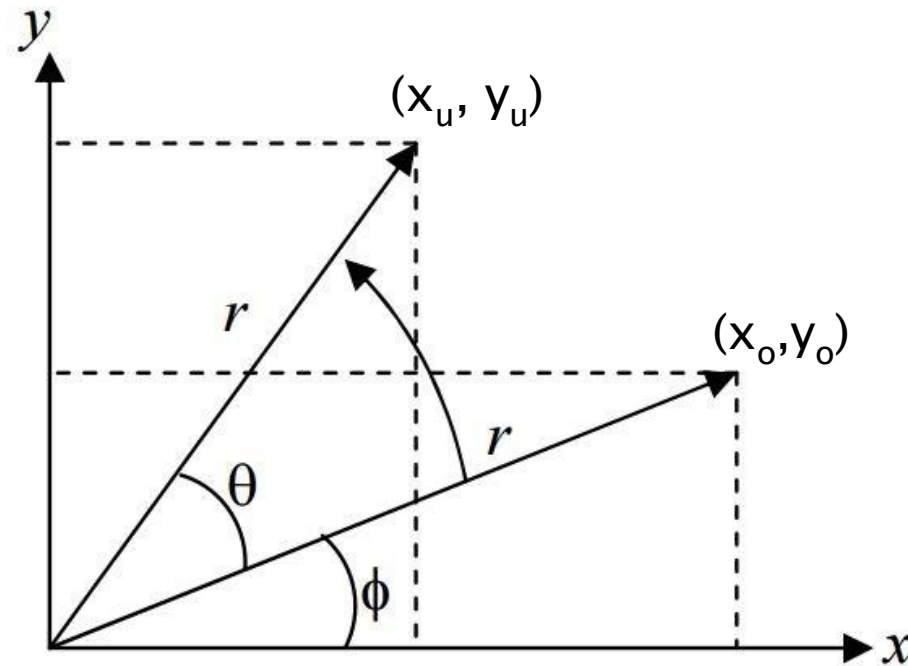
Escala

Rotação

- Modifica a **direção de um vetor** sobre algum eixo de rotação;
- No plano 2D, rotacionar um objeto significa **reposicionar** o mesmo em um **caminho circular**, semelhante aos ponteiros de um relógio;
 - O eixo de rotação neste caso é um vetor perpendicular ao plano xy e passa pelo centro de rotação □ como se fosse o eixo z;
- A rotação é especificada por um ângulo θ :
 - Valores positivos de θ rotacionam no sentido anti-horário;
 - O centro de rotação é sempre a origem do sistema de coordenadas.

Rotação

- Rotação de um vetor da posição (x_o, y_o) para a posição (x_u, y_u) segundo um ângulo θ relativo à origem do sistema de coordenadas



Rotação

- ▢ Define a orientação do objeto no universo;
- As equações de rotação de um ponto (x_o, y_o) de um ângulo são:

$$x_u = x_o * \cos(\theta) - y_o * \text{sen}(\theta)$$

$$y_u = y_o * \cos(\theta) + x_o * \text{sen}(\theta)$$

- A rotação sempre acontece na origem, assim o centro de rotação é o ponto 0,0.

Rotação

Exemplo 1:

- $P_1(2,2)$ e $P_2(4,4)$
- Ângulo 45 graus

- P_1 :

$$x_u = 2 * \cos(45) - 2 * \text{sen}(45) = 2 * 0.7071 - 2 * 0.7071 = 0$$

$$y_u = 2 * \cos(45) + 2 * \text{sen}(45) = 2 * 0.7071 + 2 * 0.7071 = 2.8284$$

- P_2 :

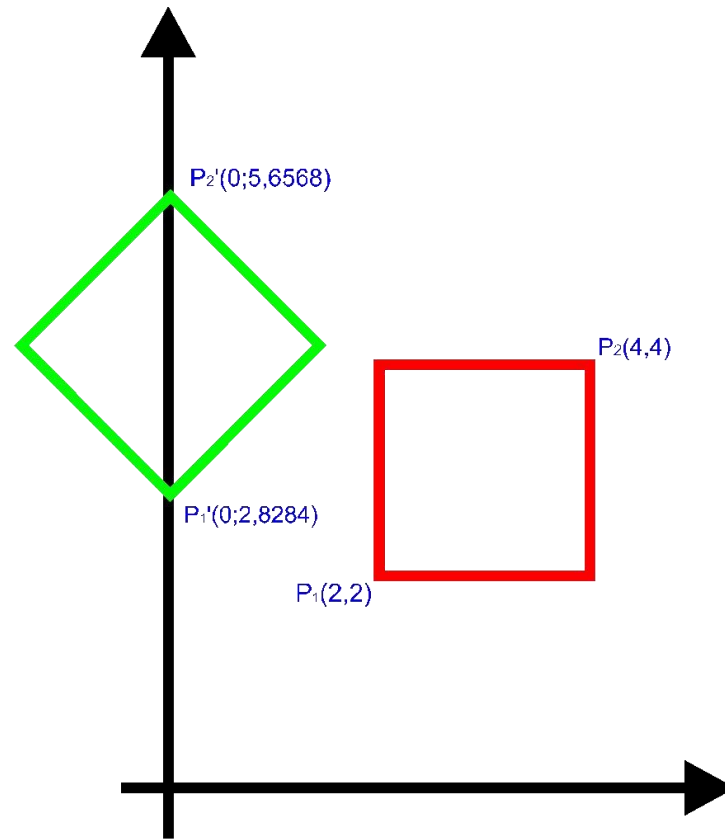
$$x_u = 4 * \cos(45) - 4 * \text{sen}(45) = 4 * 0.7071 - 4 * 0.7071 = 0$$

$$y_u = 4 * \cos(45) + 4 * \text{sen}(45) = 4 * 0.7071 + 4 * 0.7071 = 5.6568$$

$P_1'(0;2.8284)$ e $P_2'(0;5.6568)$

Rotação

□ Exemplo 1:



Rotação

- ▢ A rotação também pode ser representada na forma matricial através de uma multiplicação de matrizes:

$$P_o = \begin{bmatrix} x_o \\ y_o \end{bmatrix}$$

$$P_u = \begin{bmatrix} x_u \\ y_u \end{bmatrix}$$

$$R = \begin{bmatrix} \cos \theta & -\text{sen} \theta \\ \text{sen} \theta & \cos \theta \end{bmatrix}$$

$$P_u = R * P_o$$

$$P_u = \begin{bmatrix} \cos \theta & -\text{sen} \theta \\ \text{sen} \theta & \cos \theta \end{bmatrix} * \begin{bmatrix} x_o \\ y_o \end{bmatrix}$$

$$P_u = \begin{bmatrix} x_o \cos \theta - y_o \text{sen} \theta \\ x_o \text{sen} \theta + y_o \cos \theta \end{bmatrix}$$

Rotação

Exercício 04.

Agora adapte o algoritmo utilizado na translação para a operação de rotação! Eles são bem parecidos, serão necessários algumas alterações! Uma dica é que devemos receber (pontos, ângulo) como parâmetros. Use o exemplo do slide 25 para rotacionar $P1(2,2)$ e $P2(4,4)$.

Dica:

Converter ângulo de graus para radianos (funções python usam radianos)

angulo_rad = np.radians(angulo)

Calcular as coordenadas após a rotação

xu = x0 * np.cos(angulo_rad) - y0 * np.sin(angulo_rad)

.....

OBS: Todos exercícios devem ser enviados no final da aula, através de um link do seu repositório no github

Matrizes de Transformação

- ▶ Matriz de Translação (soma):

$$T = \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$


- ▶ Matriz de Escala (multiplicação):

$$S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

- ▶ Matriz de Rotação (multiplicação):

$$R = \begin{bmatrix} \cos \theta & -\operatorname{sen} \theta \\ \operatorname{sen} \theta & \cos \theta \end{bmatrix}$$

Coordenadas homogêneas

- Todas as transformações anteriores, **exceto a translação**, podem ser executadas por meio de **multiplicação de matrizes**;
 - Em muitos casos precisaremos representar **combinações de várias transformações** em uma única matriz;
 - Pelo fato de que a translação estará presente em quase todas as transformações, o ideal seria que ela também pudesse ser representada por uma **multiplicação de matrizes**;
 - Para isso introduziu-se o conceito de **coordenadas homogêneas**.
- 

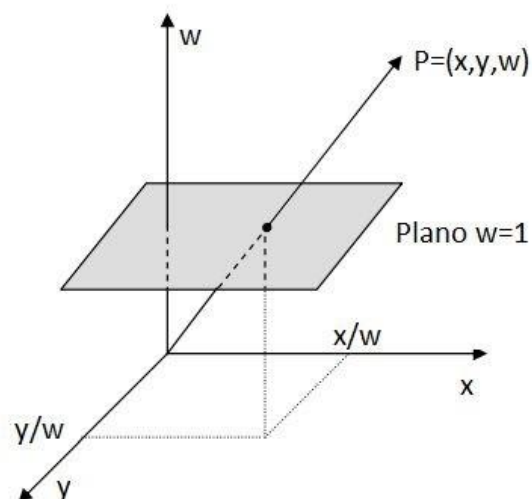
Coordenadas homogêneas

- ▣ O conceito de coordenadas homogêneas consiste em **adicionar uma dimensão a mais na matriz de transformação**;
- Desta forma, um ponto 2D é representado por (x, y, w) . Caso o valor de w não seja 1, deve-se dividir os componentes x e y por w :
 - Desta forma, os pontos $(2, 3, 6)$ e $(4, 6, 12)$ representariam a mesma informação:
 - Processo denominado homogeneização;
 - Neste caso, $\frac{x}{w}$ e $\frac{y}{w}$ são as coordenadas cartesianas do ponto homogêneo.

$$(x \quad y \quad w) = \left(\frac{x}{w} \quad \frac{y}{w} \quad 1 \right)$$

Coordenadas homogêneas

- Esta terceira coordenada do ponto pode ser interpretada geometricamente como um plano $h=1$ no \mathbb{R}^3 , conforme figura abaixo:



- O ponto $P'=(\frac{x}{w}, \frac{y}{w})$ é a projeção do ponto $P(x, y, w)$ no plano $w=1$.
- Desta forma, qualquer múltiplo do ponto P homogêneo representa o mesmo ponto.

Coordenadas Homogêneas

Com o uso de coordenadas homogêneas, as transformações são definidas como:

- Translação: $P_u = TP_o$

- $P_u = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_o \\ y_o \\ 1 \end{bmatrix}$

- Escala: $P_u = SP_o$

- $P_u = \begin{bmatrix} S_X & 0 & 0 \\ 0 & S_Y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_o \\ y_o \\ 1 \end{bmatrix}$

- Rotação: $P_u = RP_o$

- $P_u = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_o \\ y_o \\ 1 \end{bmatrix}$

Combinações de transformações

- Quando desejamos aplicar **várias transformações** a um ou mais pontos, ao invés de aplicarmos uma transformação por vez, podemos **combiná-las em uma única matriz**, e então aplicar esta matriz resultante aos pontos de um objeto:
 - Isso reduz muito o custo computacional!
- O processo de **concatenação de matrizes** é feito por meio da multiplicação das matrizes que representam cada transformação:
 - É importante lembrar que a ordem em que as multiplicações são feitas faz diferença □ é associativa, porém não comutativa!
 - $AB \neq BA$ (não comutativa)
 - $A(BC) = (AB)C$ (associativa)
 - $(FG)^T = G^T F^T$ $T=(\text{Transposta})$

Combinações de transformações

- ▶ Exemplo: concatenação de uma translação seguida de uma rotação:
 - Primeiro obtemos a matriz de transformação (M_T), multiplicando a matriz de rotação (R) pela de translação (T);
 - $M_T = RT \rightarrow$ a ordem importa!

- $$M_T = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}$$

- $$M_T = \begin{bmatrix} \cos\theta & -\sin\theta & T_x\cos\theta - T_y\sin\theta \\ \sin\theta & \cos\theta & T_x\sin\theta + T_y\cos\theta \\ 0 & 0 & 1 \end{bmatrix}$$

Combinações de transformações

▣ Exemplo: concatenação de uma translação seguida de uma rotação:

- Em seguida, multiplicamos a matriz de transformação (M_T) pelo ponto P_o ;

- $P_u = M_T P_o$

- $$P_u = \begin{bmatrix} \cos\theta & -\sin\theta & T_x\cos\theta - T_y\sin\theta \\ \sin\theta & \cos\theta & T_x\sin\theta + T_y\cos\theta \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_o \\ y_o \\ 1 \end{bmatrix}$$

Referências e material de apoio

Material do Professor Guilherme Chagas Kurtz, 2023.

GOMES, Jonas; VELHO, Luiz. Computação gráfica. Rio de Janeiro: Impa, 1998.

HEARN, Donald; Baker, M. Pauline. Computer graphics: C version. London: Prentice Hall, 1997.

HETEM JUNIOR, Annibal. Computação gráfica. Rio de Janeiro, RJ: LTC, 2006. 161 p. (Coleção Fundamentos de Informática).

HILL Jr, Francis S. Computer graphics using open GL. New Jersey: Prentice Hall, 2001.

WATT, Alan. 3D computer graphics. Harlow: Addison-Wesley, 2000

Thank you for your attention!!



Email: andre.flores@ufn.edu.br