Nome: Felipe Braz Marques
Disciplina: BCC327 - Computação Gráfica
Professor: Rafael Alves Bonfim de Queiroz

Descrição da atividade:

Etapas da Atividade:

1. **Transformação 2D - Rotação:**
   Um polígono foi criado e rotacionado em torno de seu centro. A aplicação exibe como as coordenadas do polígono mudam dinamicamente, utilizando fórmulas de rotação baseadas em trigonometria.
2. **Transformação 3D - Escala:**
   Um cubo tridimensional foi escalado, demonstrando como os vértices do cubo são ajustados proporcionalmente em relação à origem. O cubo é projetado em um espaço bidimensional para visualização.
3. **Composição de Transformação 2D:**
   Foi implementada uma composição de duas transformações geométricas: rotação e translação. Um polígono é primeiro rotacionado em torno de seu centro e, em seguida, transladado para uma nova posição na tela.
4. **Composição de Transformação 3D:**
   Um cubo tridimensional foi submetido a uma composição de duas transformações: rotação em torno de um eixo e escalonamento simultâneo. A transformação resultante ilustra como as operações geométricas podem ser combinadas para produzir efeitos complexos.

Interatividade e Recursos:

- Um menu foi implementado para permitir que o usuário escolha qual transformação deseja visualizar.
- Após cada visualização, o programa retorna ao menu principal, garantindo flexibilidade na navegação.
- A aplicação é interativa, respondendo a eventos do usuário, como fechamento da janela.

Resultados Alcançados:

A aplicação gráfica desenvolvida demonstrou com sucesso os conceitos de transformações geométricas em 2D e 3D. O uso de bibliotecas como o Pygame possibilitou a manipulação em tempo real e a criação de um ambiente visual dinâmico. Os principais resultados incluem:

- **Visualização Dinâmica:** Foi possível observar as mudanças geométricas em tempo real, facilitando a compreensão de como as transformações afetam as formas.
- **Interatividade:** A aplicação permite que o usuário explore diferentes transformações de forma independente, promovendo um aprendizado prático e intuitivo.

- **Composição de Transformações:** As composições 2D e 3D ilustraram como múltiplas transformações podem ser aplicadas em conjunto, destacando a flexibilidade e o poder das operações geométricas.

```python
import pygame
import math

# Configuração geral
WIDTH, HEIGHT = 800, 600
center = (WIDTH // 2, HEIGHT // 2)

def transformacao_2d_rotacao(screen):
    """Exemplo de Transformação Geométrica 2D: Rotação"""
    rect_points = [[100, 100], [200, 100], [200, 200], [100, 200]]
    center = (150, 150)
    angle = 0

    running = True
    clock = pygame.time.Clock()
    while running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
        angle += 1
        radians = math.radians(angle)
        rotated_points = [
            [
                center[0] + (x - center[0]) * math.cos(radians) - (y - center[1]) * math.sin(radians),
                center[1] + (x - center[0]) * math.sin(radians) + (y - center[1]) * math.cos(radians),
            ]
            for x, y in rect_points
        ]
        screen.fill((0, 0, 0))
        pygame.draw.polygon(screen, (0, 128, 255), rotated_points)
        pygame.display.flip()
        clock.tick(60)

def transformacao_3d_escala(screen):
    """Exemplo de Transformação Geométrica 3D: Escala"""
    cube_points = [
        [-50, -50, -50], [50, -50, -50], [50, 50, -50], [-50, 50, -50],
        [-50, -50, 50], [50, -50, 50], [50, 50, 50], [-50, 50, 50],
    ]
    scale = 2

    def project_3d_to_2d(point):
        z = point[2] + 200
        factor = 300 / z
        x = int(point[0] * factor + center[0])
        y = int(point[1] * factor + center[1])
        return x, y

    running = True
    clock = pygame.time.Clock()
    while running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
        scaled_points = [[p[0] * scale, p[1] * scale, p[2] * scale] for p in cube_points]
        projected_points = [project_3d_to_2d(p) for p in scaled_points]
        edges = [
            (0, 1), (1, 2), (2, 3), (3, 0),
            (4, 5), (5, 6), (6, 7), (7, 4),
            (0, 4), (1, 5), (2, 6), (3, 7)
        ]
        screen.fill((0, 0, 0))
        for edge in edges:
            pygame.draw.line(screen, (0, 128, 255), projected_points[edge[0]], projected_points[edge[1]])
        pygame.display.flip()
        clock.tick(60)

def composicao_2d(screen):
    """Composição de Transformação 2D: Rotação + Translação"""
    rect_points = [[100, 100], [200, 100], [200, 200], [100, 200]]
```

```
    center = (150, 150)
    angle = 0

    running = True
    clock = pygame.time.Clock()
    while running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
        angle += 1
        radians = math.radians(angle)
        rotated_points = [
            [
                center[0] + (x - center[0]) * math.cos(radians) - (y - center[1]) * math.sin(radians),
                center[1] + (x - center[0]) * math.sin(radians) + (y - center[1]) * math.cos(radians),
            ]
            for x, y in rect_points
        ]
        translated_points = [[x + 100, y + 50] for x, y in rotated_points]
        screen.fill((0, 0, 0))
        pygame.draw.polygon(screen, (0, 255, 128), translated_points)
        pygame.display.flip()
        clock.tick(60)

def composicao_3d(screen):
    """Composição de Transformação 3D: Rotação + Escala"""
    cube_points = [
        [-50, -50, -50], [50, -50, -50], [50, 50, -50], [-50, 50, -50],
        [-50, -50, 50], [50, -50, 50], [50, 50, 50], [-50, 50, 50],
    ]
    scale = 1.5
    angle = 0

    def project_3d_to_2d(point):
        z = point[2] + 200
        factor = 300 / z
        x = int(point[0] * factor + center[0])
        y = int(point[1] * factor + center[1])
        return x, y

    running = True
    clock = pygame.time.Clock()
    while running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
        angle += 1
        rotated_points = [
            [
                p[0] * math.cos(math.radians(angle)) - p[2] * math.sin(math.radians(angle)),
                p[1],
                p[0] * math.sin(math.radians(angle)) + p[2] * math.cos(math.radians(angle)),
            ]
            for p in cube_points
        ]
        scaled_points = [[p[0] * scale, p[1] * scale, p[2] * scale] for p in rotated_points]
        projected_points = [project_3d_to_2d(p) for p in scaled_points]
        edges = [
            (0, 1), (1, 2), (2, 3), (3, 0),
            (4, 5), (5, 6), (6, 7), (7, 4),
            (0, 4), (1, 5), (2, 6), (3, 7)
        ]
        screen.fill((0, 0, 0))
        for edge in edges:
            pygame.draw.line(screen, (255, 128, 0), projected_points[edge[0]], projected_points[edge[1]])
        pygame.display.flip()
        clock.tick(60)

def main():
    pygame.init()
```

```
    screen = pygame.display.set_mode((WIDTH, HEIGHT))
    pygame.display.set_caption("Transformações Geométricas com Pygame")

    while True:
        print("Escolha uma transformação para visualizar:")
        print("1 - Transformação 2D: Rotação")
        print("2 - Transformação 3D: Escala")
        print("3 - Composição 2D: Rotação + Translação")
        print("4 - Composição 3D: Rotação + Escala")
        print("5 - Sair")

        choice = input("Digite o número da transformação desejada: ")

        if choice == "1":
            transformacao_2d_rotacao(screen)
        elif choice == "2":
            transformacao_3d_escala(screen)
        elif choice == "3":
            composicao_2d(screen)
        elif choice == "4":
            composicao_3d(screen)
        elif choice == "5":
            print("Encerrando o programa.")
            break
        else:
            print("Opção inválida. Tente novamente.")

    pygame.quit()

if __name__ == "__main__":
    main()
```