Atividade 3: Transformações Geométricas 2D e 3D com Pygame

Discente: Nicolas Expedito Lana Mendes
Matrícula: 22.1.4028
Docente: Rafael Alves Bonfim de Queiroz
Universidade Federal de Ouro Preto
Departamento de Computação
BCC327 - Computação Gráfica

Fevereiro de 2025

1 Introdução

Este relatório descreve a implementação de uma aplicação gráfica interativa utilizando a biblioteca **Pygame**. A aplicação tem como objetivo demonstrar, por meio de exemplos práticos, diferentes transformações geométricas vistas em sala de aula, organizadas em quatro cenas:

- **Cena a)** Transformação 2D: rotação de um retângulo.
- **Cena b)** Transformação 3D: escala dinâmica de um cubo, com projeção para 2D.
- **Cena c)** Composição de transformações 2D: rotação seguida de escala aplicada ao retângulo.
- **Cena d)** Composição de transformações 3D: rotação em torno do eixo Y combinada com translação de um cubo.

A escolha do Pygame possibilita a renderização e a manipulação em tempo real dos objetos, evidenciando a aplicação prática dos conceitos de computação gráfica.

2 API Utilizada

A aplicação utiliza a biblioteca **Pygame**, que oferece recursos para criação de janelas, gerenciamento de eventos e desenho de formas geométricas. As transformações geométricas foram implementadas manualmente, utilizando as funções da biblioteca **math** do Python para cálculos trigonométricos e de escala.

3 Trechos de Código

A seguir, são apresentados os trechos mais relevantes da implementação.

3.1 Funções de Transformação

As funções `rotate_2d`, `scale_3d` e `project_3d_to_2d` são utilizadas para realizar, respectivamente, a rotação em 2D, a escala em 3D e a projeção de objetos 3D em um plano 2D. Já as funções `compose_2d` e `compose_3d` demonstram a composição de transformações.

```
def rotate_2d(points, angle, center):
    rotated_points = []
    angle_rad = math.radians(angle)
    for x, y in points:
        x -= center[0]
        y -= center[1]
        new_x = x * math.cos(angle_rad) - y * math.sin(angle_rad)
        new_y = x * math.sin(angle_rad) + y * math.cos(angle_rad)
        new_x += center[0]
        new_y += center[1]
        rotated_points.append((new_x, new_y))
    return rotated_points

def scale_3d(points_3d, scale_factor):
    return [(x * scale_factor, y * scale_factor, z * scale_factor
       )
            for x, y, z in points_3d]

def project_3d_to_2d(points_3d, distance=200):
    projected = []
    for x, y, z in points_3d:
        factor = distance / (distance + z)
        x_2d = x * factor + WIDTH/2
        y_2d = y * factor + HEIGHT/2
        projected.append((x_2d, y_2d))
    return projected

def compose_2d(points, angle, scale, center):
    rotated = rotate_2d(points, angle, center)
    scaled = []
    for x, y in rotated:
        x = (x - center[0]) * scale + center[0]
        y = (y - center[1]) * scale + center[1]
        scaled.append((x, y))
    return scaled

def compose_3d(points_3d, angle_y, translation):
    composed = []
    angle_rad = math.radians(angle_y)
    for x, y, z in points_3d:
        new_x = z * math.sin(angle_rad) + x * math.cos(angle_rad)
        new_z = z * math.cos(angle_rad) - x * math.sin(angle_rad)
        new_y = y
        new_x += translation[0]
        new_y += translation[1]
        new_z += translation[2]
```

```
        composed.append((new_x, new_y, new_z))
    return composed
```

3.2 Loop Principal e Controle de Cenas

No loop principal, são tratadas as entradas do usuário para alternar entre as quatro cenas (teclas 1 a 4) e a renderização é atualizada continuamente, demonstrando as transformações aplicadas aos objetos.

```
# Objetos iniciais
rect_2d = [(200, 200), (300, 200), (300, 300), (200, 300)]
cube_3d = [(-50, -50, -50), (50, -50, -50), (50, 50, -50), (-50,
   50, -50),
           (-50, -50, 50), (50, -50, 50), (50, 50, 50), (-50, 50,
              50)]

angle = 0
current_scene = 1
font = pygame.font.Font(None, 36)

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_1:
                current_scene = 1  # Rota  o 2D
            elif event.key == pygame.K_2:
                current_scene = 2  # Escala 3D
            elif event.key == pygame.K_3:
                current_scene = 3  # Composi  o 2D (rota  o +
                    escala)
            elif event.key == pygame.K_4:
                current_scene = 4  # Composi  o 3D (rota  o Y
                    + transla  o)

    screen.fill(BLACK)

    if current_scene == 1:
        center_2d = (250, 250)
        rotated_rect = rotate_2d(rect_2d, angle, center_2d)
        pygame.draw.polygon(screen, RED, rotated_rect, 2)
        text = font.render("a) Rota  o 2D (Tecla 1)", True,
           WHITE)
        screen.blit(text, (10, 10))

    elif current_scene == 2:
        scaled_cube = scale_3d(cube_3d, 1 + math.sin(angle/20)/2)
        projected_scaled = project_3d_to_2d(scaled_cube)
        for i in range(4):
```

```
            pygame.draw.line(screen, BLUE, projected_scaled[i],
                projected_scaled[(i+1)%4], 1)
            pygame.draw.line(screen, BLUE, projected_scaled[i+4],
                projected_scaled[(i+1)%4+4], 1)
            pygame.draw.line(screen, BLUE, projected_scaled[i],
                projected_scaled[i+4], 1)
        text = font.render("b) Escala 3D (Tecla 2)", True, WHITE)
        screen.blit(text, (10, 10))

    elif current_scene == 3:
        center_2d = (250, 250)
        composed_2d = compose_2d(rect_2d, angle, 1.5, center_2d)
        pygame.draw.polygon(screen, GREEN, composed_2d, 2)
        text = font.render("c) Rota  o + Escala 2D (Tecla 3)",
            True, WHITE)
        screen.blit(text, (10, 10))

    elif current_scene == 4:
        composed_3d = compose_3d(cube_3d, angle, (0, 0, 50))
        projected_composed = project_3d_to_2d(composed_3d)
        for i in range(4):
            pygame.draw.line(screen, WHITE, projected_composed[i
                ], projected_composed[(i+1)%4], 1)
            pygame.draw.line(screen, WHITE, projected_composed[i
                +4], projected_composed[(i+1)%4+4], 1)
            pygame.draw.line(screen, WHITE, projected_composed[i
                ], projected_composed[i+4], 1)
        text = font.render("d) Rota  o Y + Transla  o 3D (
            Tecla 4)", True, WHITE)
        screen.blit(text, (10, 10))

    instructions = font.render("Use teclas 1-4 para mudar entre
        transforma  es", True, WHITE)
    screen.blit(instructions, (10, HEIGHT - 40))

    angle += 1
    pygame.display.flip()
    clock.tick(60)

pygame.quit()
```

4 Instruções de Execução

Para executar a aplicação:

1. Instale o Python e a biblioteca **Pygame** (por exemplo, via `pip install pygame`).
2. Salve o código em um arquivo `.py` (por exemplo, `atividade3.py`).
3. Execute o script com o comando `python atividade3.py`.

4. Utilize as teclas 1 a 4 para alternar entre as cenas e visualizar as diferentes transformações.

5 Funcionalidade e Resultados

A aplicação apresenta quatro cenas que demonstram:

- **Cena a)** Uma rotação 2D de um retângulo em torno de seu centro.
- **Cena b)** Uma escala dinâmica de um cubo 3D, com projeção para 2D, permitindo observar variações de tamanho.
- **Cena c)** A composição de uma rotação seguida de uma escala aplicada ao retângulo, evidenciando a importância da ordem das transformações.
- **Cena d)** Uma composição em 3D onde o cubo sofre rotação no eixo Y combinada com uma translação, demonstrando o efeito de transformações compostas.

A cada atualização do loop principal, o ângulo de rotação é incrementado, gerando animação contínua e possibilitando a visualização dos efeitos em tempo real.

6 Conclusão

A implementação utilizando Pygame permitiu explorar de forma prática os conceitos de transformações geométricas 2D e 3D, assim como a composição de transformações. A aplicação interativa demonstra como operações de rotação, escala e translação podem ser combinadas para manipular objetos gráficos, contribuindo para o entendimento dos fundamentos da computação gráfica.

7 Imagens da aplicação

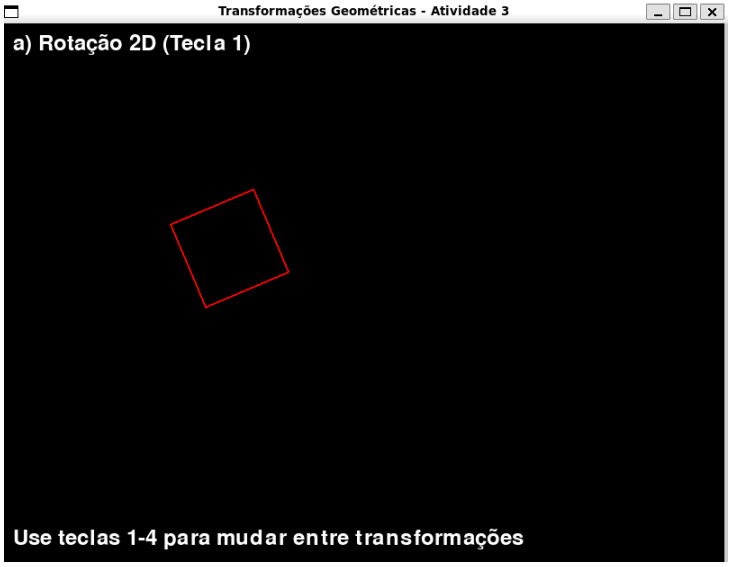

Figure 1: Cena a) Rotação 2D do retângulo.

Figure 2: Cena b) Escala 3D do cubo com projeção.

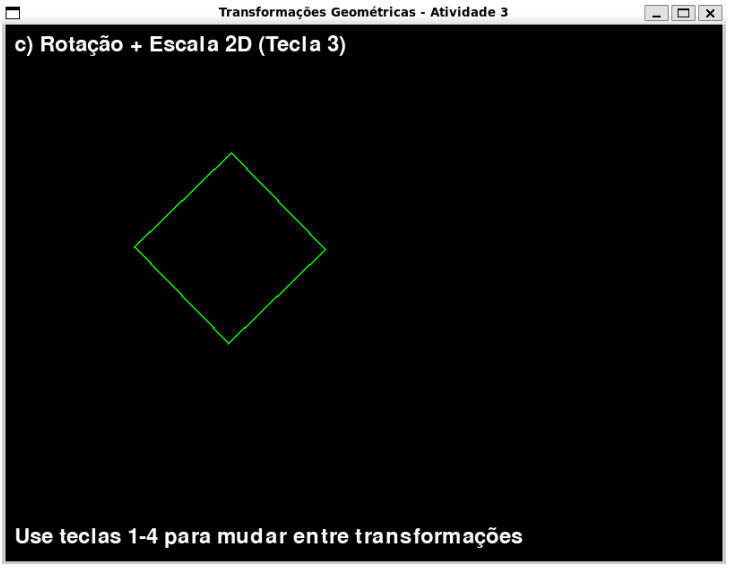

Figure 3: Cena c) Composição de rotação e escala 2D.

Figure 4: Cena d) Composição de rotação Y e translação 3D.