Atividade 9: Mapeamento de Textura e Recorte de Linhas 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

Março de 2025

1 Introdução

Esta atividade abrange dois tópicos importantes de computação gráfica implementados com **Pygame**:

1. **Exercício 9.1 – Mapeamento de Textura em Esfera:** Uma esfera é renderizada com mapeamento de textura utilizando coordenadas UV derivadas de ângulos esféricos. A esfera pode ser rotacionada e os parâmetros da textura podem ser ajustados interativamente.

2. **Exercício 9.2 – Recorte de Linhas (Clipping):** Um conjunto de linhas é recortado contra uma janela de visualização retangular, utilizando o algoritmo de *Cohen-Sutherland*.

2 Descrição da Implementação

2.1 Exercício 9.1: Mapeamento de Textura em Esfera

Neste exercício, foi criada uma esfera simulada por meio de coordenadas esféricas, dividida em fatias (*slices*) e camadas (*stacks*). A textura é aplicada à esfera por meio de interpolação baricêntrica entre coordenadas UV mapeadas a partir das subdivisões angulares.

- A rotação da esfera é realizada em tempo real por meio de transformações 3D em torno dos eixos x e y.

- A projeção dos pontos tridimensionais na tela é feita usando projeção perspectiva simples.

- As coordenadas UV são calculadas diretamente com base nos índices angulares de cada subdivisão, e a textura é amostrada para colorir os triângulos que compõem a esfera.
- Em caso de ausência do arquivo de textura, uma imagem padrão gerada proceduralmente é usada.

Listing 1: Projeção perspectiva de um ponto 3D para 2D

```
def project(x, y, z):
    focal = 500
    scale = focal / (focal + z)
    px = width // 2 + int(x * scale)
    py = height // 2 + int(y * scale)
    return px, py
```

Listing 2: Rotação 3D de pontos da esfera

```
def get_vertex(phi, theta):
    x = sphere_radius * math.sin(phi) * math.cos(theta)
    y = sphere_radius * math.sin(phi) * math.sin(theta)
    z = sphere_radius * math.cos(phi)

    # Rota  o em X
    y_rot = y * math.cos(rotation_x) - z * math.sin(rotation_x)
    z_rot = y * math.sin(rotation_x) + z * math.cos(rotation_x)

    # Rota  o em Y
    x_rot = x * math.cos(rotation_y) + z_rot * math.sin(
       rotation_y)
    z_rot = -x * math.sin(rotation_y) + z_rot * math.cos(
       rotation_y)

    return x_rot, y_rot, z_rot
```

Listing 3: Interpolação baricêntrica e aplicação de textura

```
# Dentro da fun  o draw_textured_triangle
a = ((y1 - y2)*(x - x2) + (x2 - x1)*(y - y2)) / denom
b = ((y2 - y0)*(x - x2) + (x0 - x2)*(y - y2)) / denom
c = 1 - a - b

# Coordenadas UV interpoladas
u = a*uv0[0] + b*uv1[0] + c*uv2[0]
v = a*uv0[1] + b*uv1[1] + c*uv2[1]
```

O usuário pode controlar a esfera com as seguintes teclas:

- `Setas`: Rotacionam a esfera nos eixos x e y.
- `Q/A`: Aumentam ou diminuem a escala da textura.
- `W/S`: Deslocam a textura no eixo x.
- `E/D`: Deslocam a textura no eixo y.

2.2 Exercício 9.2: Recorte de Linhas

Este exercício implementa o algoritmo de recorte de linhas **Cohen-Sutherland**, que determina a parte visível de uma linha dentro de uma janela de visualização retangular.

- A janela de recorte é representada como um retângulo branco.
- Linhas não recortadas são desenhadas em vermelho.
- Linhas visíveis após o recorte são desenhadas em verde.
- O algoritmo classifica os pontos das linhas por regiões usando códigos de 4 bits e aplica testes sucessivos de trivial aceitação, rejeição ou interseção.

Listing 4: Cálculo do código de região (Cohen-Sutherland)

```
def compute_code(x, y):
    code = INSIDE
    if x < view_x1: code |= LEFT
    elif x > view_x2: code |= RIGHT
    if y < view_y1: code |= TOP
    elif y > view_y2: code |= BOTTOM
    return code
```

Listing 5: Trecho principal do algoritmo de Cohen-Sutherland

```
while True:
    if not (code1 | code2):  # Ambos dentro
        accept = True
        break
    elif code1 & code2:  # Ambos fora (mesma regi o)
        break
    else:
        code_out = code1 if code1 else code2
        ...
        # Atualiza ponto e recodifica
        if code_out == code1:
            x1, y1 = x, y
            code1 = compute_code(x1, y1)
        else:
            x2, y2 = x, y
            code2 = compute_code(x2, y2)
```

O usuário pode alternar entre os exercícios pressionando:

- 1: Exibe o exercício 9.1.
- 2: Exibe o exercício 9.2.

3 Adaptações Realizadas

Como alternativa ao uso de bibliotecas como OpenGL, as seguintes adaptações foram realizadas com a biblioteca Pygame:

- **Projeção 3D**: Foi implementada uma projeção perspectiva manual, baseada em distância focal.
- **Rotação 3D**: Foram aplicadas fórmulas de rotação em torno dos eixos x e y, utilizando transformações lineares.
- **Mapeamento de Textura**: As coordenadas de textura foram mapeadas diretamente usando subdivisões esféricas, e a interpolação foi feita ponto a ponto.
- **Renderização de Triângulos**: A rasterização de triângulos com interpolação baricêntrica foi feita manualmente.
- **Algoritmo de Clipping**: O algoritmo de Cohen-Sutherland foi codificado em Python e aplicado a várias linhas.

4 Instruções de Execução

1. Instale o Python e a biblioteca `pygame` (ex.: `pip install pygame`).
2. Coloque uma imagem chamada `textura.jpg` no mesmo diretório do script (opcional).
3. Salve o código da implementação como `atividade9.py`.
4. Execute com o comando `python atividade9.py`.
5. Utilize as seguintes teclas:
   - `1`: Ativa o exercício 9.1 (Esfera com textura).
   - `2`: Ativa o exercício 9.2 (Recorte de linhas).
   - **Durante o exercício 9.1:**
     - `Setas`: Rotacionam a esfera.
     - `Q/A`: Aumentam ou diminuem a escala da textura.
     - `W/S, E/D`: Ajustam o deslocamento da textura nos eixos x e y.

5 Resultados e Discussão

Exercício 9.1

A esfera texturizada apresenta um resultado visual interessante e interativo. A interpolação das coordenadas de textura com rotação realista permite a simulação de um objeto 3D com textura mapeada.

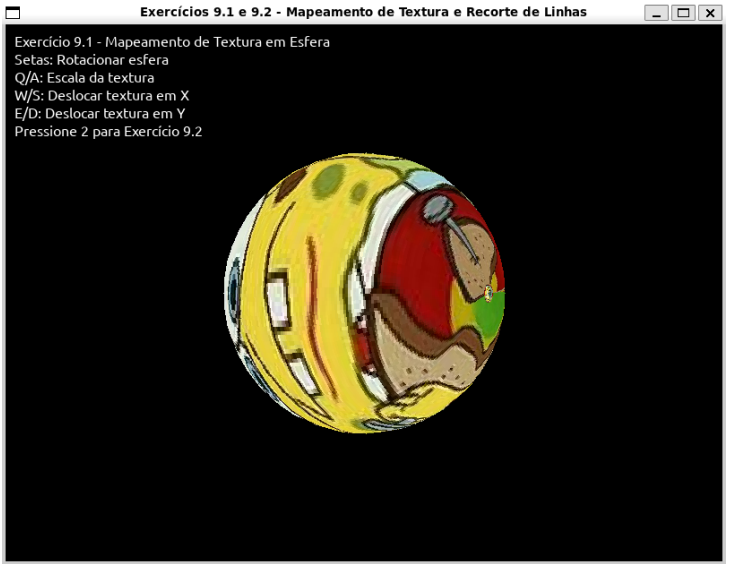

Figure 1: Esfera com textura aplicada, rotacionada interativamente. Parte 1

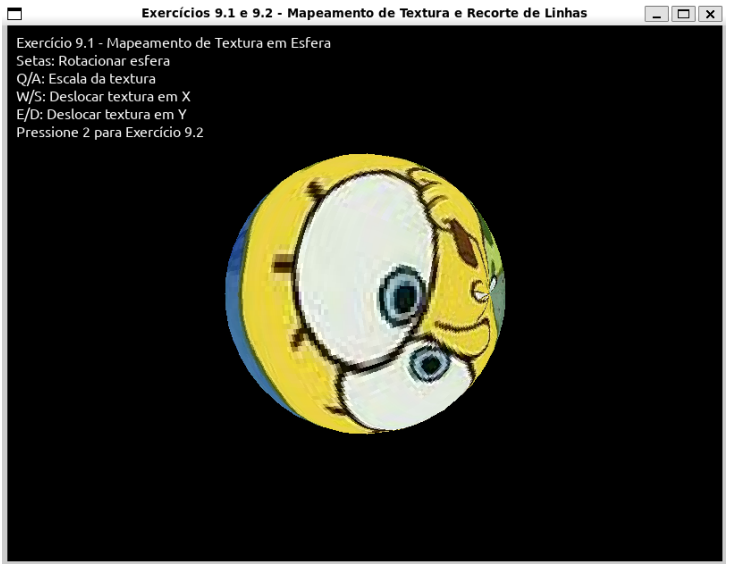

Figure 2: Esfera com textura aplicada, rotacionada interativamente. Parte 2

Exercício 9.2

O algoritmo de Cohen-Sutherland funciona corretamente, recortando linhas com base na janela de visualização. As linhas visíveis dentro da janela são renderizadas em verde, demonstrando o sucesso do recorte.

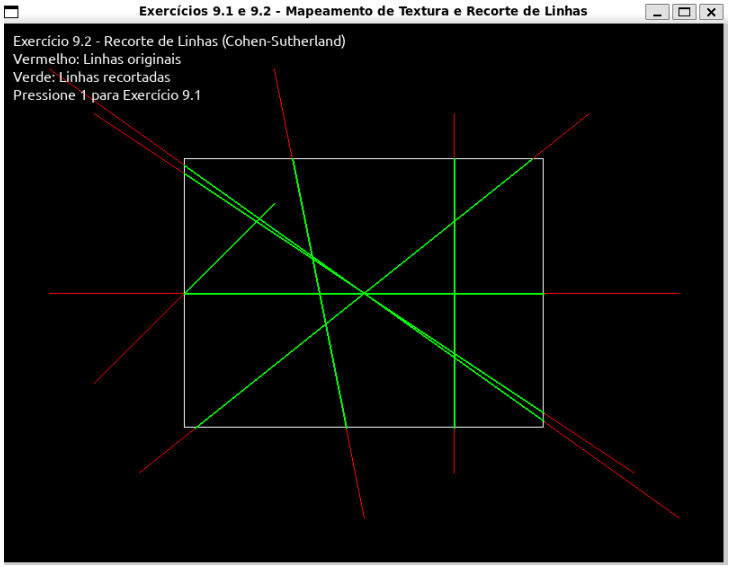

Figure 3: Exercício 9.2: Linhas recortadas com algoritmo de Cohen-Sutherland

6 Conclusão

A implementação dos exercícios 9.1 e 9.2 em **Pygame** mostrou que é possível realizar mapeamento de textura e recorte de linhas com algoritmos clássicos sem depender de bibliotecas gráficas 3D como OpenGL. A esfera foi renderizada de forma eficiente com textura mapeada e rotação 3D, enquanto o algoritmo de clipping demonstrou corretamente o funcionamento de técnicas de visibilidade em janelas de visualização.