UNIVERSIDADE FEDERAL DE RORAIMA CENTRO DE CIÊNCIA E TECNOLOGIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
DCC703 - COMPUTAÇÃO GRÁFICA (2024.2)
Prof. LUCIANO FERREIRA SILVA

Nome: Marcia Gabrielle Bonifácio De Oliveira - 2020011319

TRABALHO 5

1. INTRODUÇÃO

O recorte de polígonos é uma técnica essencial em computação gráfica, utilizada para remover partes de um objeto que estão fora da área visível da tela ou dentro de uma região de interesse. Isso é útil em aplicações como sistemas CAD, jogos, renderização de cenas 3D e visualização de dados.
Este trabalho tem como objetivo implementar e testar um algoritmo de recorte de polígonos utilizando uma janela retangular e comparar os resultados visuais obtidos.
O algoritmo escolhido para esse processo foi Sutherland-Hodgman, amplamente utilizado por sua simplicidade e eficiência.

2. CONCEITOS BÁSICOS

2.1 Algoritmos de Recorte

O processo de recorte consiste em eliminar partes indesejadas de um objeto gráfico que não pertencem a uma determinada região de recorte. Existem diferentes algoritmos para isso, dependendo do tipo de objeto a ser recortado:

- Recorte de linha: Algoritmos como Cohen-Sutherland e Liang-Barsky.
- Recorte de polígonos: Algoritmo de Sutherland-Hodgman.

2.2 Algoritmo de Sutherland-Hodgman

O algoritmo de Sutherland-Hodgman é utilizado para recortar polígonos convexos contra uma janela retangular. Ele funciona iterativamente, aplicando recorte contra cada borda da janela.

Passos do Algoritmo:

1. Para cada aresta da janela de recorte (esquerda, direita, topo e base):
    - Verificar se os vértices do polígono estão dentro ou fora dessa aresta.
    - Adicionar os vértices internos ao novo polígono.
    - Se um segmento cruza a borda, calcular e armazenar o ponto de interseção.
2. O processo é repetido para todas as arestas da janela, gerando o polígono recortado final.

3. Implementação em Python

A seguir, apresentamos a implementação do Algoritmo de Recorte de Sutherland-Hodgman.

Esta função “inside” verifica se um ponto está dentro ou fora da janela de recorte. Ela recebe um ponto e uma borda da região de recorte, retornando True se o ponto está dentro e False caso contrário.

```python
def inside(point, clip_edge):
    x, y = point
    edge_x1, edge_y1, edge_x2, edge_y2 = clip_edge
    return (x >= edge_x1 if edge_x1 == xmin else x <= edge_x1) if edge_x1 == edge_x2 else (y >= edge_y1 if edge_y1 == ymin else y <= edge_y1)
```

A Função “intersection” calcula a interseção entre uma aresta do polígono e uma borda da janela.

- Fórmulas para interseção:
    - Se a borda da janela for vertical (x constante), calcula y.
    - Se a borda for horizontal (y constante), calcula x.

```python
def intersection(p1, p2, clip_edge):
    x1, y1 = p1
    x2, y2 = p2
    edge_x1, edge_y1, edge_x2, edge_y2 = clip_edge

    if edge_x1 == edge_x2:
        x_int = edge_x1
        m = (y2 - y1) / (x2 - x1) if x2 != x1 else 0
        y_int = y1 + m * (x_int - x1)
    else:
        y_int = edge_y1
        m = (y2 - y1) / (x2 - x1) if x2 != x1 else 0
        x_int = x1 + (y_int - y1) / m if m != 0 else x1

    return (x_int, y_int)
```

A Função “clip_polygon” é a principal do Algoritmo de Sutherland-Hodgman. Ela aplica o recorte sucessivamente para cada borda da janela, gerando um novo polígono recortado.

```
def clip_polygon(polygon, clip_rect):
    x_min, x_max, y_min, y_max = clip_rect
    clip_edges = [
        (x_min, y_min, x_min, y_max),  # Esquerda
        (x_min, y_max, x_max, y_max),  # Superior
        (x_max, y_max, x_max, y_min),  # Direita
        (x_max, y_min, x_min, y_min)   # Inferior
    ]

    clipped_polygon = polygon[:]
    for clip_edge in clip_edges:
        new_polygon = []
        prev_point = clipped_polygon[-1]
        for curr_point in clipped_polygon:
            if inside(curr_point, clip_edge):
                if not inside(prev_point, clip_edge):
                    intersec = intersection(prev_point, curr_point, clip_edge)
                    new_polygon.append(intersec)
                new_polygon.append(curr_point)
            elif inside(prev_point, clip_edge):
                intersec = intersection(prev_point, curr_point, clip_edge)
                new_polygon.append(intersec)
            prev_point = curr_point
        clipped_polygon = new_polygon

    return clipped_polygon
```

Define as bordas da janela de recorte (esquerda, superior, direita e inferior).
Itera sobre cada borda, aplicando o recorte:

- Se um ponto está dentro, ele é mantido.
- Se uma aresta cruza a borda, o ponto de interseção é adicionado.
- Se um ponto está fora, ele é removido.

4. RECORTE DOS POLÍGONOS

Foram aplicados testes com diferentes polígonos antes e depois do recorte.