UNIVERSIDADE FEDERAL DE RORAIMA CENTRO DE CIÊNCIA E TECNOLOGIA BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
DCC703 – COMPUTAÇÃO GRÁFICA (2024.2)

Data de entrega: 19/03/2025

DISCENTES:
FELIPE RUBENS DE SOUSA BORGES (2020020120)

COMPUTAÇÃO GRÁFICA

Relatório Recorte de Polígonos - Sutherland

Boa Vista-RR
2025.2

Relatório de Recorte de Polígonos - Sutherland

Relatório apresentado para o projeto de Recorte de Polígonos por meio do algoritmo de Sutherland da disciplina de Computação Gráfica, ofertada pelo curso de Ciência da Computação da Universidade Federal de Roraima.

Prof. Luciano Ferreira

Boa Vista-RR
2025.2

Sumário

Resumo

Este relatório descreve a implementação de uma interface gráfica para visualização do algoritmo de Sutherland, sua construção e os resultados obtidos utilizando a biblioteca Matplotlib, em Python. O algoritmo Sutherland-Dodgman é um método clássico para recortar polígonos em relação a uma janela de recorte retangular, para garantir que apenas a parte visível de um polígono seja renderizada. O programa tem como objetivo realizar o recorte de polígonos através do algoritmo de Sutherland.

Algoritmo e Fundamentação

O algoritmo de Sutherland-Hodgman é um método clássico em computação gráfica para recortar (clipping) polígonos contra uma janela retangular convexa. Ele é utilizado para determinar quais partes de um polígono estão dentro de uma área de visualização definida - janela de recorte - e descartar as partes que estão fora. O algoritmo funciona processando o polígono vértice por vértice em relação a cada borda da janela de recorte (esquerda, direita, inferior e superior), gerando um novo polígono que contém apenas a parte visível.

O algoritmo baseia-se em:

1. Teste de visibilidade: Verifica se um vértice está dentro ou fora de uma borda específica da janela.
2. Cálculo de interseções: Quando um segmento cruza uma borda da janela, calcula o ponto de interseção.
3. Casos de recorte: Avalia pares de vértices consecutivos e decide quais pontos incluir no polígono resultante com base em quatro casos possíveis:
    - Ambos os vértices dentro: adiciona o vértice final.
    - Primeiro dentro, segundo fora: adiciona o ponto de interseção.
    - Primeiro fora, segundo dentro: adiciona o ponto de interseção e o vértice final.
    - Ambos fora: não adiciona nada.

Resultando em um novo polígono que dentro dos limites da janela de recorte.

Passo a Passo do algoritmo:

- Definição da Função **inside**:
    - Recebe um ponto p (x, y) e uma borda(tipo e valor, como "left" e -1).
    - Verifica se o ponto está dentro da borda:
        - Para "left": x >= valor.
        - Para "right": x <= valor.
        - Para "bottom": y >= valor.
        - Para "top": y <= valor.
    - Retorna True se o ponto está dentro, False caso esteja fora.
- Definição da Função **intersecao**:
    - Calcula o ponto de interseção entre um segmento (definido por p1 e p2) e uma borda.
    - Usa equações lineares para determinar as coordenadas (x, y):
        - Para bordas verticais ("left" ou "right"): fixa x e calcula y.

        - Para bordas horizontais ("bottom" ou "top"): fixa y e calcula x.
    - Retorna o ponto de interseção como uma tupla (x, y).
- Implementação do Algoritmo Sutherland-Hodgman (**sutherland_hodgman_clip**):
    - Recebe o polígono original e a janela de recorte.
    - Para cada borda da janela:
        - Cria uma lista de vértices (new_poligono).
        - Para cada par de vértices consecutivos (prev_point e current_point):
            - Verifica se estão dentro ou fora da borda usando inside.
            - Aplica os quatro casos:
                - Ambos dentro: adiciona o vértice atual.
                - Dentro para fora: adiciona o ponto de interseção.
                - Fora para dentro: adiciona o ponto de interseção e o vértice atual.
                - Ambos fora: ignora.
        - Atualiza o polígono recortado com a nova lista.
    - Retorna o polígono recortado.
- Definição dos **Polígonos** e **Janela de Recorte**:
    - polygons: Dicionário com quatro polígonos de teste (retângulo, triângulo, cruz e pentágono) definidos por listas de vértices (x, y).
    - clip_janela: Lista de tuplas representando as bordas da janela (-1 a 1 em x e y).
- Visualização:
    - Para cada polígono:
        - Aplica o recorte com sutherland_hodgman_clip.

Implementação

```python
# função para verificar se um ponto está dentro da janela
def inside(p, borda):
    x, y = p
    borda_type, borda_value = borda
    if borda_type == "left":
        return x >= borda_value
    elif borda_type == "right":
        return x <= borda_value
    elif borda_type == "bottom":
        return y >= borda_value
    elif borda_type == "top":
        return y <= borda_value
    return False
```

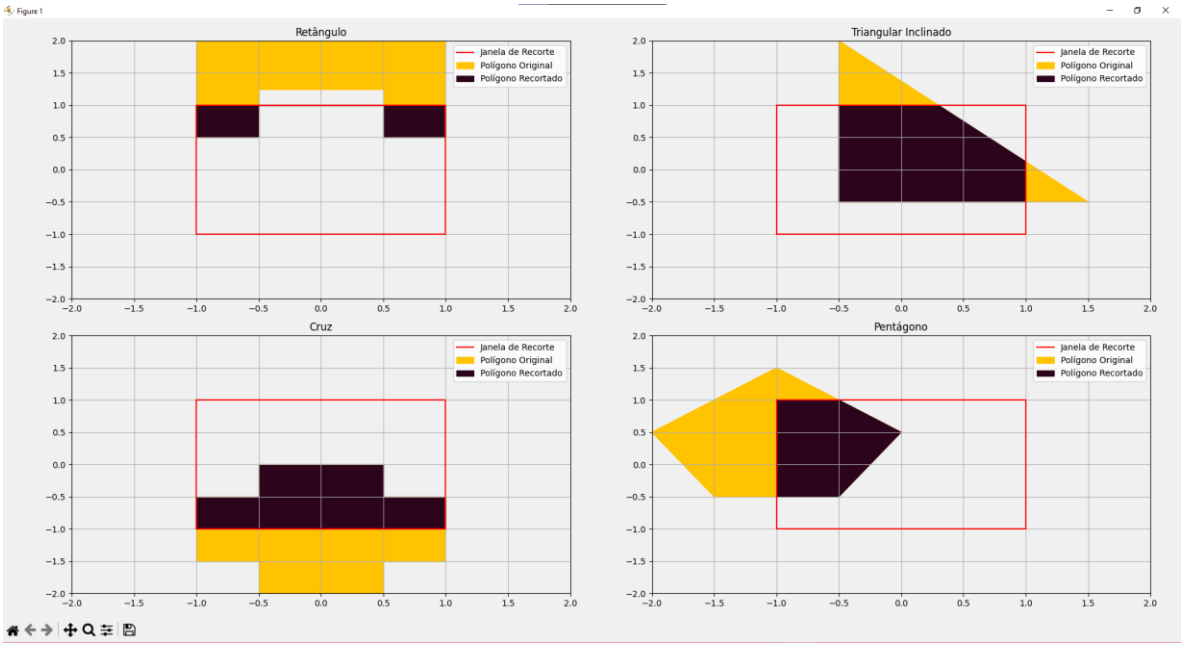
```python
# função para calcular a interseção de um segmento com a borda
def intersecao(p1, p2, borda):
    x1, y1 = p1
    x2, y2 = p2
    borda_type, borda_value = borda

    if borda_type in ["left", "right"]:
        x = borda_value
        y = y1 + (y2 - y1) * (x - x1) / (x2 - x1)
    else:
        y = borda_value
        x = x1 + (x2 - x1) * (y - y1) / (y2 - y1)

    return (x, y)
```

```python
# algoritmo de Sutherland-Hodgman
def sutherland_hodgman_clip(poligono, clip_janela):
    clipped_poligono = poligono

    # 2. Para cada lado, observa-se a relação entre vértices sucessivos e as janelas (limites)
    # percorre os lados do polígono e compara cada vértice com as bordas da janela de recorte
    for borda in clip_janela:
        new_poligono = []
        for i in range(len(clipped_poligono)):
            current_point = clipped_poligono[i]
            prev_point = clipped_poligono[i - 1]

            current_inside = inside(current_point, borda)
            prev_inside = inside(prev_point, borda)

            if prev_inside and current_inside:
                new_poligono.append(current_point) # CASO 1: Um dos dois vértices é adicionado à lista de saídas (p)
            elif prev_inside and not current_inside:
                new_poligono.append(intersecao(prev_point, current_point, borda)) # CASO 2: O ponto “i” de interseção é tratado como um vértice de saída
            elif not prev_inside and current_inside:
                new_poligono.append(intersecao(prev_point, current_point, borda))
                new_poligono.append(current_point) # CASO 4: Os dois pontos “i” e “p” são colocados na lista de vértices de saída

        # 3. Lados definidos pelos vértices da lista de saídas serão apresentados na tela
        clipped_poligono = new_poligono

    return clipped_poligono
```

```python
# 1. Suponha um polígono de lados dados por vértices: v1, v2,..., vn
# define os polígonos
poligonos = {
    "Retângulo": [(-1, 2), (1, 2), (1, 0.5), (0.5, 0.5), (0.5, 1.25), (-0.5, 1.25), (-0.5, 0.5), (-1, 0.5)],
    "Triangular Inclinado": [(-0.5, 2), (1.5, -0.5), (-0.5, -0.5)],
    "Cruz": [(-0.5, 0), (0.5, 0), (0.5, -0.5), (1, -0.5), (1, -1.5), (0.5, -1.5), (0.5, -2), (-0.5, -2), (-0.5, -1.5), (-1, -1.5), (-1, -0.5), (-0.5, -0.5)],
    "Pentágono": [(-1.5, -0.5), (-2, 0.5), (-1, 1.5), (0, 0.5), (-0.5, -0.5)]
}
```

```python
# define da janela de recorte
clip_janela = [("left", -1), ("right", 1), ("bottom", -1), ("top", 1)]

fig, axes = plt.subplots(2, 2, figsize=(10, 10))
axes = axes.flatten()

for ax in axes:
    ax.set_facecolor('#f0f0f0')

for ax, (name, poligono) in zip(axes, poligonos.items()):
    # aplica o recorte
    clipped_poligono = sutherland_hodgman_clip(poligono, clip_janela)

    # exibe os novos pontos após o recorte
    print(f"\nPolígono {name}")
    for i, point in enumerate(clipped_poligono):
        print(f"Ponto {i}: {point}")

    # janela de recorte
    janela_x = [-1, 1, 1, -1, -1]
    janela_y = [-1, -1, 1, 1, -1]
    ax.plot(janela_x, janela_y, 'r', label="Janela de Recorte")
```

Resultados



Os polígonos testados foram determinados nos slides da matéria:

- Retângulo
- Triângulo Inclinado
- Cruz
- Pentágono

Como a complexidade depende do número de arestas do polígono e do número de bordas da janela de recorte, polígonos mais complexos devem requerer mais iterações, tornando mais custoso, mas ainda viável do ponto de vista computacional.

Conclusão

O código apresentado implementa o algoritmo de Sutherland-Hodgman, uma técnica fundamental em computação gráfica para recortar polígonos contra uma janela retangular convexa. Ele combina uma fundamentação teórica sólida — baseada em testes de visibilidade e cálculo de interseções — com uma execução prática que processa vértices de forma iterativa. A estrutura do código permite determinar de forma eficiente quais partes de um polígono estão dentro dos limites definidos, ajustando-o passo a passo. Em suma, o código destaca sua utilidade em aplicações gráficas e sua lógica sistemática para resolver problemas de recorte, com tuplas garantindo a integridade dos dados e uma abordagem passo a passo que reflete os princípios da geometria computacional.