UNIVERSIDADE FEDERAL DE RORAIMA
CENTRO DE CIÊNCIA E TECNOLOGIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
DCC703 - COMPUTAÇÃO GRÁFICA

RAFAEL NÓBREGA DE LIMA

ALGORITMOS DE RECORTE - RELATÓRIO

BOA VISTA - RR

2025

1. BASE DOS ALGORITMOS E ORGANIZAÇÃO

O algoritmo foi implementado usando a biblioteca “pygame”, com essa biblioteca é possível ver o resultado dos recortes em uma tela 800x600.

```
pygame.init()
WIDTH, HEIGHT = 800, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Recorte de Polígono - Sutherland Hodgman")
clock = pygame.time.Clock()
```

1.1. ESCOLHA DO POLÍGONO

```
# Polígonos ajustados conforme os exemplos das imagens
triangulo = [(310, 102), (314, 270), (577, 265)]
hexagono = [(232, 105), (366, 185), (310, 306), (139, 297), (111, 159)]
cruz = [(310, 320), (420, 320), (420, 360), (460, 360), (460, 440), (420, 440), (420, 480),
        (310, 480), (310, 440), (270, 440), (270, 360), (310, 360)]
u_bugado = [(250, 100), (250, 210), (290, 210), (290, 140), (370, 140), (370, 200), (410, 200), (410, 100)]

# Escolha do polígono descomentando um deles
# polygon = triangulo
# polygon = hexagono
# polygon = cruz
polygon = u_bugado
```

Nessa parte do código é onde estão definidos os polígonos e é onde se escolhe cada polígono apenas descomentando a linha que corresponde a cada polígono.

1.2. DESENHO DOS POLÍGONOS E A ÁREA DO RECORTE

```
# Desenhar o polígono original
pygame.draw.polygon(screen, (0, 255, 0), polygon, 1)

# Desenhar o retângulo de recorte
pygame.draw.rect(screen, (255, 0, 0), pygame.Rect(clip_rect[0], clip_rect[1],
                                                  clip_rect[2] - clip_rect[0], clip_rect[3] - clip_rect[1]), 1)

# Desenhar o polígono recortado
if len(clipped_polygon) > 2:
    pygame.draw.polygon(screen, (0, 0, 255), clipped_polygon, 0)
```

Através dessas linhas de código ocorre o desenho da área de corte em vermelho, também é desenhado o polígono original (antes do recorte) na cor verde e o polígono após o recorte na cor azul.

1.3. APLICAÇÃO DO RECORTE

```
clipped_polygon = sutherland_hodgman_clip(polygon, clip_rect)

# Printar as coordenadas do polígono após o recorte apenas uma vez, sem casas decimais
if not clipped_once:
    print("Coordenadas após o recorte:", [(round(x), round(y)) for x, y in clipped_polygon])
    clipped_once = True
```

Aqui é onde ocorre o recorte e a impressão dos novos pontos após o recorte.

2. ALGORITMO DE SUTHERLAND-HODGMAN

O algoritmo de Sutherland-Hodgman recorta um polígono contra um retângulo processando um lado de cada vez. Ele começa com todos os vértices do polígono original e verifica quais estão dentro da área de recorte. Se um vértice estiver fora, a interseção com a borda de recorte é calculada e adicionada.

Esse processo se repete para cada lado do recorte (esquerda, direita, baixo e topo), refinando o polígono até que apenas a parte totalmente visível permaneça. O resultado final é um novo polígono dentro da área de recorte.Passos do algoritmo:

- Percorrer todas as bordas da área de recorte: processamos as bordas esquerda, direita, inferior e superior, uma por vez.
- Processar os vértices do polígono original e gerar um novo polígono mantendo apenas os pontos visíveis para a borda atual.
- Manter os vértices que estão dentro da área de recorte.
- Se um vértice estiver fora e o próximo dentro, calcular e adicionar o ponto de interseção com a borda de recorte.
- Se um vértice estiver dentro e o próximo fora, adicionar apenas o ponto de interseção.
- Repetir esse processo para cada borda, refinando progressivamente o polígono até que ele esteja totalmente dentro da área de recorte.

2.1. IMPLEMENTAÇÃO

A função **sutherland_hodgman_clip(polygon, clip_rect)** recebe um polígono (lista de coordenadas) e um retângulo de recorte e retorna o polígono recortado. mas para isso acontecer existe algumas outras funções que devem ser processadas antes a primeira é a **inside(p, edge)**

```python
def inside(p, edge):
    x, y = p
    x_min, y_min, x_max, y_max = clip_rect
    if edge == 'left':
        return x >= x_min
    elif edge == 'right':
        return x <= x_max
    elif edge == 'bottom':
        return y >= y_min
    elif edge == 'top':
        return y <= y_max
```

Nessa função ocorre uma verificação para ver se tem algum ponto dentro da área de recorte para uma determinada borda, a verificação é feita nas quatro bordas do retângulo.

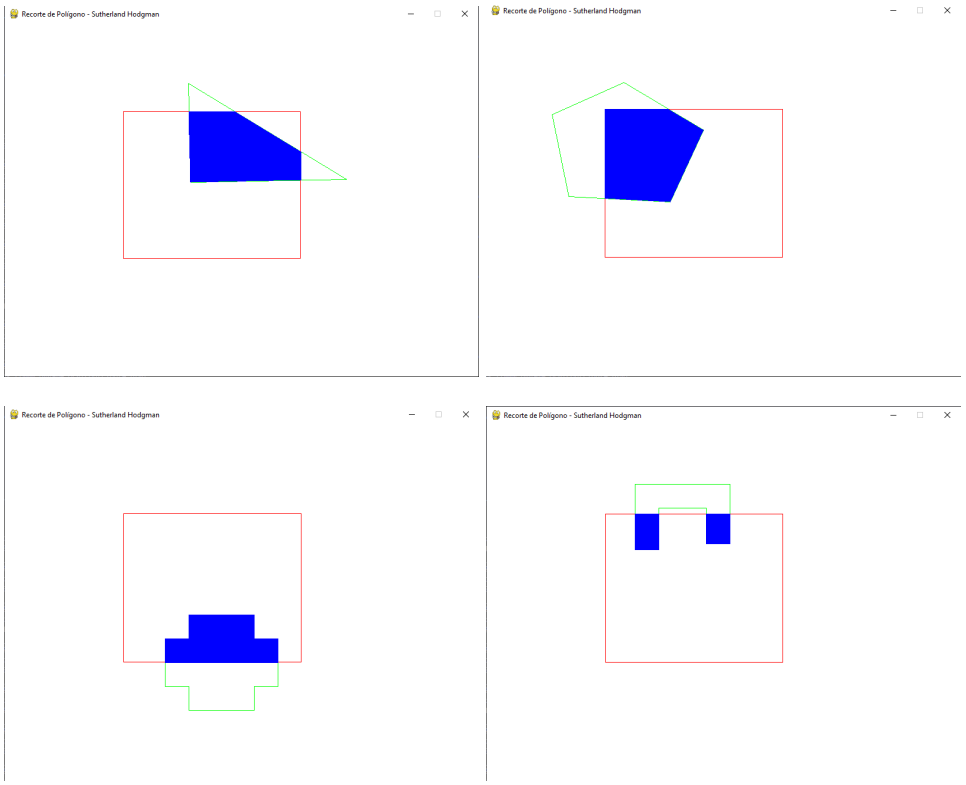
```
def compute_intersection(p1, p2, edge):
    x1, y1 = p1
    x2, y2 = p2
    x_min, y_min, x_max, y_max = clip_rect

    if edge == 'left':
        x = x_min
        y = y1 + (y2 - y1) * (x_min - x1) / (x2 - x1)
    elif edge == 'right':
        x = x_max
        y = y1 + (y2 - y1) * (x_max - x1) / (x2 - x1)
    elif edge == 'bottom':
        y = y_min
        x = x1 + (x2 - x1) * (y_min - y1) / (y2 - y1)
    elif edge == 'top':
        y = y_max
        x = x1 + (x2 - x1) * (y_max - y1) / (y2 - y1)

    return (round(x), round(y))
```

Essa função é responsável por calcular o ponto de interseção, então quando uma linha do polígono cruza a borda do retângulo de recorte, essa função calcula a interseção exata da linha com a borda de recorte.

```
edges = ['left', 'right', 'bottom', 'top']
clipped_polygon = polygon[:]

for edge in edges:
    new_polygon = []
    prev_point = clipped_polygon[-1]  # Começamos com o último vértice

    for point in clipped_polygon:
        if inside(point, edge):  # Se o ponto atual está dentro
            if not inside(prev_point, edge): # Mas o anterior estava fora
                new_polygon.append(compute_intersection(prev_point, point, edge))
            new_polygon.append(point) # Mantemos o ponto
        elif inside(prev_point, edge): # Se o ponto anterior estava dentro, mas este não
            new_polygon.append(compute_intersection(prev_point, point, edge))

        prev_point = point # Atualiza o ponto anterior para o próximo laço

    clipped_polygon = new_polygon

return clipped_polygon
```

Por último, nessa parte do código é onde o polígono é progressivamente reduzido ao ser cortado contra cada uma das quatro bordas

2.2. RESULTADOS



3. CONCLUSÃO

O algoritmo de Sutherland-Hodgman é uma solução eficiente para recorte de polígonos, processando cada borda da área de recorte de forma progressiva. Ele é amplamente utilizado em computação gráfica, garantindo que apenas as partes visíveis de um polígono sejam preservadas. Sua principal vantagem é a simplicidade e eficiência, tornando-o ideal para polígonos convexos. No entanto, para polígonos côncavos, técnicas adicionais podem ser necessárias. No geral, o algoritmo permanece uma ferramenta essencial no processamento gráfico, oferecendo um método confiável para recorte preciso de polígonos.