UNIOESTE – Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

Título: Fillpoly

Discentes: GNSantos, GYLHiguchi, MHGaspar, NSLemos

Data: 18 /08/2025

---

1. Introdução

Este relatório detalha a implementação do algoritmo de interpolação incremental para o preenchimento de polígonos. O objetivo é preencher um polígono com cores gradientes, interpolando as cores não apenas entre as arestas, mas também ao longo de cada linha de varredura (scanline). O processo pode ser dividido em duas etapas principais: a interpolação vertical entre as scanlines e a interpolação horizontal dentro de cada scanline.

2. Interpolação entre Scanlines (Interpolação Vertical)

A primeira etapa do processo de preenchimento é determinar os pontos de interseção de cada linha de varredura com as arestas do polígono. Para cada aresta do polígono, o código calcula a interseção com as scanlines que ela atravessa. O algoritmo itera através de cada aresta do polígono:

1. **Iteração de Arestas**: Para cada par de pontos consecutivos (x1,y1) e (x2,y2) que formam uma aresta, o algoritmo verifica se a aresta é

vertical. Se não for, ele determina a direção do crescimento de y para garantir que a varredura seja sempre de baixo para cima.

2. **Cálculo da Inclinação**: A inclinação (Tx) da aresta é calculada como dx/dy = (x2-x1)/(y2-y1). Isso permite encontrar o incremento de x para cada incremento de y.
3. **Interpolação de Coordenadas**: A coordenada x da interseção é calculada incrementalmente. A partir do ponto inicial da aresta (x1,y1), a coordenada x para a próxima scanline (y1+1) é x1+Tx, para a scanline y1+2 é x1+2Tx, e assim por diante. Essa abordagem evita recalcular a equação da reta para cada scanline, tornando o processo mais eficiente.
4. **Interpolação de Cores**: Simultaneamente à interpolação de coordenadas, o código realiza a interpolação de cores. Para cada aresta, são geradas duas cores aleatórias para seus pontos finais. A cor em qualquer ponto intermediário da aresta é calculada usando uma interpolação linear: Corint=Corinicial+t*(Corfinal−Corinicial), onde t=(y - yinicia)l/(yfinal- yinicia)l é o fator de interpolação. A função interpolar_cor realiza essa operação para os canais RGB.

Ao final dessa etapa, o algoritmo armazena, para cada scanline, uma lista de pontos de interseção, cada um com sua coordenada x e a cor interpolada correspondente.

3. Interpolação dentro das Scanlines (Interpolação Horizontal)

Após calcular todas as interseções para uma dada scanline, a próxima fase é preencher o espaço entre elas. Para cada scanline, os pontos de interseção são ordenados pela sua coordenada x. O preenchimento ocorre em pares de pontos:

1. **Varredura Horizontal**: O algoritmo percorre os pontos de interseção da scanline em pares, de dois em dois. Por exemplo, do ponto de interseção 1 para o 2, do 3 para o 4, e assim por diante.

2. **Interpolação de Coordenadas**: Para cada par de pontos (xini,Corini) e (xfim,Corfim), o algoritmo itera sobre os pixels de xini a xfim.
3. **Interpolação de Cores**: Para cada pixel na coordenada x, a cor é interpolada linearmente usando as cores dos pontos de interseção. A equação de interpolação é: Corpixel=Corini+t*(Corfim−Corini), onde t=(x-xini)/(xfim-xini)i é o fator de interpolação horizontal. A cor resultante é então usada para pintar o pixel.

A implementação usa canvas.create_line para desenhar uma linha de 1 pixel de comprimento (x, y, x + 1, y), o que efetivamente desenha um ponto com a cor interpolada.

4. Estruturas de Dados

O código utiliza as seguintes estruturas para gerenciar os dados:

- pontos: Lista temporária para armazenar os pontos do polígono que está sendo desenhado.
- interseções: Um dicionário onde as chaves são as coordenadas y das scanlines, e os valores são listas de tuplas (x, cor_interpolada), representando os pontos de interseção.
- linhas_preenchimento: Um dicionário que mapeia o ID de cada polígono para uma lista de IDs das linhas (pixels) desenhadas para o preenchimento. Isso permite a exclusão ou atualização do preenchimento

5. Metodologia

O processo de interpolação vertical e de cores ao longo das arestas é realizado na função preencher_poligono.

Loop para Iterar sobre Arestas: O código percorre cada aresta do polígono para calcular as interseções. A aresta é definida por dois pontos, (x1, y1) e (x2, y2).

Python

```python
for i in range(num_pontos):
    x1, y1 = pontos[i]
    x2, y2 = pontos[(i + 1) % num_pontos]
    cor1 = cores_arestas[i]
    cor2 = cores_arestas[(i + 1) % num_pontos]
```

Cálculo da Inclinação (Tx): A inclinação é calculada como dx / dy. O código garante que a varredura seja de baixo para cima (y1 < y2).

Python

```python
if y1 > y2:
    x1, y1, x2, y2 = x2, y2, x1, y1
    cor1, cor2 = cor2, cor1
dx = x2 - x1
dy = y2 - y1
Tx = dx / dy
x = x1
```

Interpolação de Coordenadas e de Cores: Um loop percorre as scanlines (y de y1 a y2). Para cada y, a coordenada x é calculada incrementalmente (x += Tx), e a cor é interpolada entre as cores dos pontos finais da aresta. O fator de interpolação t é calculado com base na posição y. A função interpolar_cor realiza a interpolação para os canais RGB.

Python

```python
for y in range(y1, y2):
    t = (y - y1) / dy
    cor_interpolada = interpolar_cor(cor1, cor2, t)
    intersecoes[y].append((x, cor_interpolada))
    x += Tx
```

2. Interpolação dentro das Scanlines (Interpolação Horizontal)

O preenchimento horizontal e a interpolação de cores dentro das scanlines também são realizados na função preencher_poligono.

Iteração sobre Scanlines e Ordenação de Interseções: O código percorre as scanlines (y) que possuem interseções. Os pontos de interseção são ordenados pela coordenada x.

Python

```python
self.linhas_preenchimento[poligono_id] = []
for y, pontos_x in intersecoes.items():
    pontos_x.sort(key=lambda p: p[0])
```

Varredura e Interpolação Horizontal: O loop percorre os pares de pontos de interseção. Para cada par, a cor é interpolada para cada pixel (x) entre os pontos x_ini e x_fim. A função interpolar_cor é usada novamente, mas desta vez o fator de interpolação t é baseado na coordenada x.

Python

```python
for i in range(0, len(pontos_x) - 1, 2):
    x_ini, cor_ini = pontos_x[i]
    x_fim, cor_fim = pontos_x[i + 1]
    for x in range(round(x_ini), round(x_fim)):
        t = (x - x_ini) / (x_fim - x_ini) if x_fim != x_ini else 0
        cor_ini_rgb = hex_para_rgb(cor_ini)
        cor_fim_rgb = hex_para_rgb(cor_fim)
        cor = interpolar_cor(cor_ini_rgb, cor_fim_rgb, t)
        linha_id = self.canvas.create_line(x, y, x + 1, y, fill=cor)
        self.linhas_preenchimento[poligono_id].append(linha_id)
```

5. Conclusão

O método de interpolação incremental, permite um preenchimento eficiente de polígonos com gradientes de cores. Ao combinar a interpolação vertical ao longo das arestas com a interpolação horizontal dentro das scanlines, o algoritmo cria uma transição de cores suave e visualmente agradável. A abordagem incremental é computacionalmente vantajosa, pois minimiza a quantidade de cálculos de ponto flutuante repetitivos, tornando a renderização rápida.