UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
COLEGIADO DE CIÊNCIA DA COMPUTAÇÃO

Relatório Atividade Prática 03 – Computação Gráfica

Título: Implementação do Algoritmo FillPoly

Alunos: Fernando Seiji Onoda Inomata
Gabriel Pereira
Jonathan Tadei

Data: 18/08/2025

---

1. INTRODUÇÃO

Este relatório detalha a implementação do algoritmo de preenchimento de polígonos, conhecido como *Fillpoly*, no contexto da disciplina de Computação Gráfica. O objetivo principal do trabalho é demonstrar o entendimento e a aplicação da técnica de interpolação incremental para o preenchimento de polígonos com cores sólidas. A implementação foi realizada utilizando a Godot Engine e a linguagem GDScript.

O algoritmo Fillpoly, ou *Scanline Fill*, é um método eficiente para preencher polígonos, que funciona varrendo a área do polígono linha por linha (scanline) e preenchendo os pixels que se encontram dentro de suas fronteiras. A eficiência do algoritmo reside no uso da coerência de varredura, que permite o cálculo incremental das interseções das arestas com as scanlines.

2. INTERPOLAÇÃO INCREMENTAL

A interpolação incremental é uma técnica fundamental em computação gráfica que permite calcular valores de forma eficiente, aproveitando a coerência espacial. Em vez de recalcular valores a partir do zero para cada ponto, a interpolação incremental utiliza o valor do ponto anterior para calcular o próximo, adicionando um pequeno incremento.

No algoritmo Fillpoly, essa técnica é aplicada de duas formas principais:

1. **Interpolação de Coordenadas:** Para determinar onde as arestas do polígono cruzam cada scanline.

2. **Interpolação de Cores:** Em polígonos com cores distintas nos vértices, para determinar a cor de cada pixel no interior do polígono (sombreamento de Gouraud).

3. INTERPOLAÇÃO ENTRE SCANLINES (COORDENADAS)

O processo de interpolação de coordenadas entre as scanlines é o coração do algoritmo Fillpoly. Ele consiste em calcular de forma incremental os pontos onde as arestas do polígono interceptam cada linha de varredura horizontal.

O processo, conforme implementado no arquivo area_2d.gd, pode ser descrito nos seguintes passos:

1. **Determinação dos Limites Verticais:** O algoritmo primeiramente identifica os valores mínimo e máximo da coordenada Y do polígono (Ymin e Ymax) para determinar a primeira e a última scanline que interceptam o polígono.
2. **Processamento das Arestas:** Para cada aresta do polígono, definida pelos vértices p1 e p2, o algoritmo calcula a variação em X (dx) e em Y (dy).
3. **Cálculo do Incremento (Coeficiente Angular Inverso):** O incremento tx é calculado como a razão dx / dy. Esse valor representa o quanto a coordenada X da interseção da aresta com a scanline muda a cada incremento de uma unidade na coordenada Y. Essencialmente, é o inverso do coeficiente angular da reta que contém a aresta.
4. **Varedura Incremental:** O algoritmo então itera por todas as scanlines entre o vértice inicial (startPoint) e o vértice final (endPoint) da aresta. Para a primeira scanline que cruza a aresta, a coordenada X da interseção é a própria coordenada X do vértice inicial. Para cada scanline subsequente, o novo valor de X é calculado simplesmente somando o incremento tx ao valor de X anterior (x += tx).

Este processo é muito mais eficiente do que recalcular a interseção para cada scanline utilizando a equação da reta, pois envolve apenas uma soma por scanline, em vez de uma multiplicação e uma soma.

4. INTERPOLAÇÃO DENTRO DAS SCANLINES

Uma vez que todas as interseções de todas as arestas com uma determinada scanline são calculadas e armazenadas, o próximo passo é preencher os pixels entre essas interseções.

1. **Ordenação das Interseções:** As coordenadas X das interseções para a scanline atual são ordenadas em ordem crescente.

2. **Preenchimento entre Pares:** O algoritmo então preenche os pixels entre cada par de interseções. O primeiro par define o início e o fim do primeiro segmento de preenchimento, o segundo par define o segundo segmento, e assim por diante.

Na nossa implementação, o preenchimento é realizado pela função draw_line, que desenha uma linha horizontal entre os pontos line_start e line_end, utilizando a cor de preenchimento definida para o polígono (poligono["corDentro"]).

5. INTERPOLAÇÃO DE CORES

Embora a implementação atual utilize uma cor de preenchimento sólida para todo o polígono, a interpolação incremental também é a base para o preenchimento com sombreamento suave (Gouraud), onde cada vértice do polígono possui uma cor distinta. O processo seria uma extensão do que foi feito para as coordenadas:

1. **Interpolação de Cores nas Arestas:** Assim como a coordenada X é interpolada ao longo de uma aresta, os componentes de cor (R, G, B) de cada vértice também seriam interpolados. Para cada aresta, calcular-se-ia o incremento da cor por scanline.
2. **Armazenamento das Cores nas Interseções:** Ao calcular as interseções de X para cada scanline, os valores de cor interpolados correspondentes também seriam armazenados.
3. **Interpolação de Cores na Scanline:** Durante o preenchimento de uma scanline, entre um par de interseções, a cor seria interpolada pixel a pixel. O incremento da cor por pixel na direção de X seria calculado com base nas cores das duas interseções e na distância em pixels entre elas. Cada pixel ao longo da linha horizontal seria então preenchido com a cor interpolada correspondente.

Essa dupla interpolação — primeiro ao longo das arestas e depois ao longo das scanlines — garante um preenchimento suave e eficiente, característico do sombreamento de Gouraud.

6. CONCLUSÃO

A implementação do algoritmo Fillpoly demonstrou com sucesso a eficácia da interpolação incremental para o preenchimento de polígonos. A técnica reduz significativamente a carga computacional ao evitar cálculos repetitivos, substituindo-os por simples adições incrementais. O código desenvolvido em GDScript reflete os passos fundamentais do algoritmo de Scanline, desde a determinação das interseções das arestas até o preenchimento das linhas de varredura. A compreensão deste método é crucial não apenas para o preenchimento de polígonos, mas também como base para técnicas mais avançadas em computação gráfica, como o sombreamento de Gouraud.

7. REFERÊNCIAS

HEARN, D.; BAKER, M. P. *Computer Graphics with OpenGL*. 4ª ed. Upper Saddle River, NJ: Prentice Hall, 2011.

SANTACATARINA, A. *Algoritmos de Preenchimento*. Slides de aula da disciplina de Computação Gráfica, UNIOESTE. Disponível em: https://www.inf.unioeste.br/~adair/CG/Materiais/Slides/. Acesso em: 10 ago. 2025.

GODOT ENGINE. *Custom drawing in 2D*. Documentação Oficial da Godot Engine. Disponível em: https://docs.godotengine.org/en/stable/tutorials/2d/custom_drawing_in_2d.html. Acesso em: 15 ago. 2025.

GODOT ENGINE. *Stable documentation*. Documentação Oficial da Godot Engine. Disponível em: https://docs.godotengine.org/en/stable. Acesso em: 11 ago. 2025.