UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Ciência Da Computação

Relatório de Aula Prática 02 – Computação Gráfica

Título: Prática 2 – Fill Poly

Alunos: Fábio Kenji Sato, João Vitor da Silva, Vinicius Vieira Viana Data: 15/08/2025

---

1 INTRODUÇÃO

Este trabalho apresenta um algoritmo para preenchimento de polígonos, "Fillpoly", cujo objetivo é a conversão de polígonos vetoriais 2D, definidos por um conjunto de vértices, em uma representação rasterizada com seu interior preenchido (CATARINA, 2025).

A solução foi implementada como uma aplicação web interativa, utilizando HTML, CSS e JavaScript, que executa o algoritmo de preenchimento por meio de uma interface gráfica para a criação e manipulação dos polígonos. A implementação incorpora um preenchimento com interpolação de cores entre vértices, onde as cores definidas em cada vértice são suavemente mescladas no interior da forma por meio de um gradiente.

2 METODOLOGIA

A metodologia do projeto abrange desde a interação do usuário com a interface gráfica até a implementação do algoritmo de preenchimento *Scanline*.

2.1 Interface e Fluxo de Uso

O programa apresenta uma interface web onde o usuário pode realizar um fluxo completo de criação e manipulação de polígonos. As etapas são:

- Para iniciar a criação, o usuário clica no botão "Desenhar"e seleciona a cor desejada para os vértices.

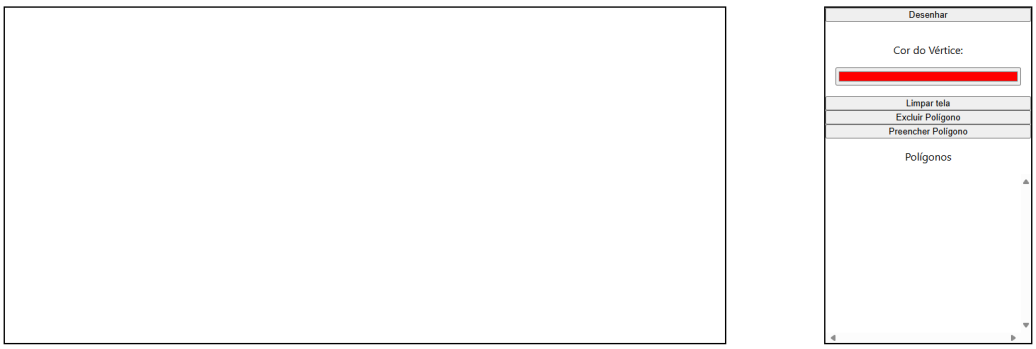

Figura 1: Painel de controle com o botão "Desenhar"e seletor de cor.

- Em sequência, o usuário clica na área de desenho para criar os vértices do polígono. É necessário um mínimo de três vértices.

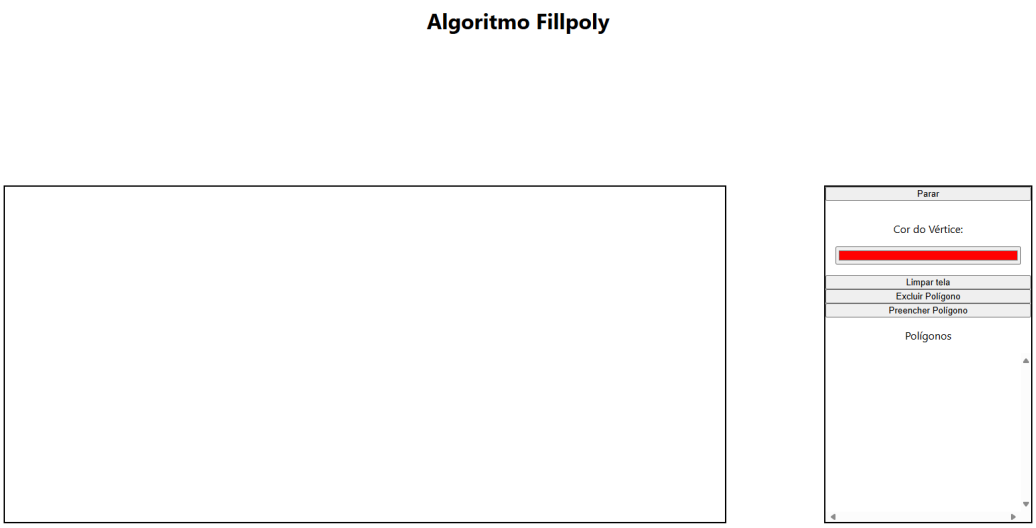

Figura 2: Criação de vértices e arestas no canvas.

- Após criar os pontos, o usuário clica no botão "Parar"para finalizar o polígono, que é automaticamente fechado e adicionado à lista de polígonos.

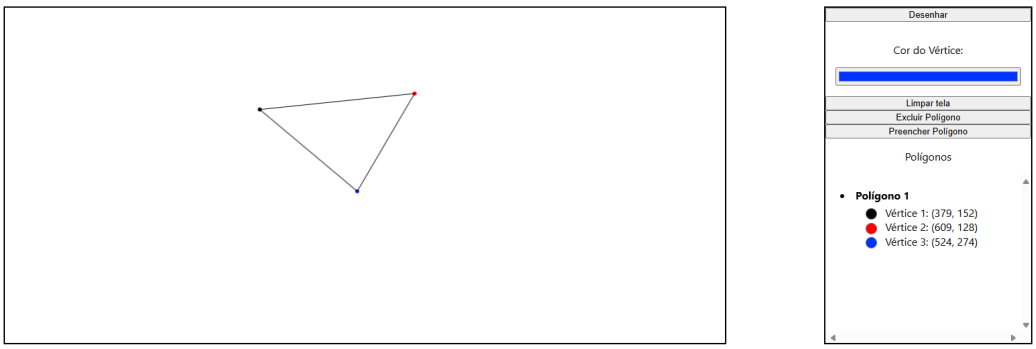

Figura 3: Polígono finalizado e listado no painel lateral.

- Para preencher os objetos, o usuário clica no botão "Preencher Polígono", aplicando o algoritmo a todos os polígonos na tela.

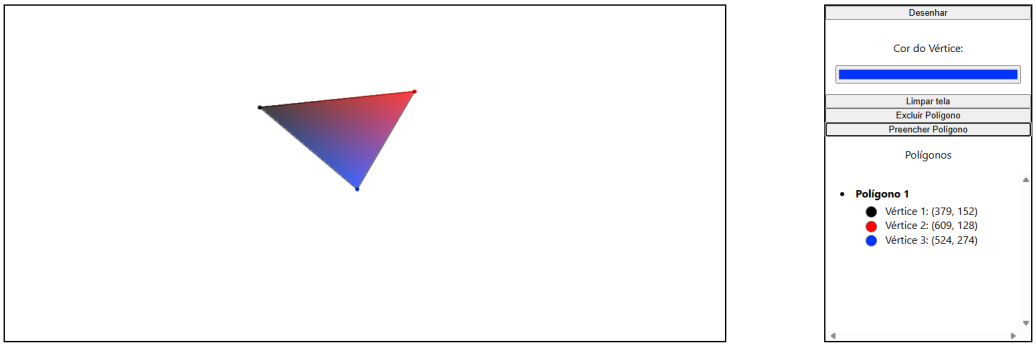

Figura 4: Polígono preenchido com interpolação de cores entre os vértices.

- A aplicação permite a criação de múltiplos polígonos na mesma cena.

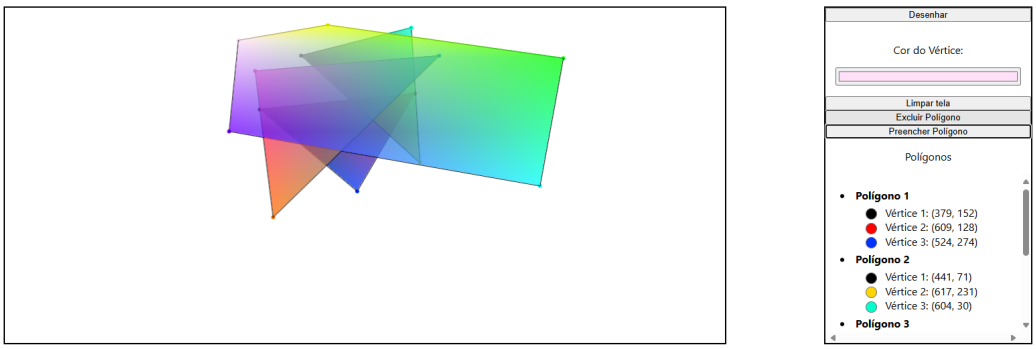

Figura 5: Múltiplos polígonos desenhados e preenchidos.

- Para remover um polígono, o usuário o seleciona na lista (clicando em seu título) e em seguida clica no botão "Excluir Polígono".

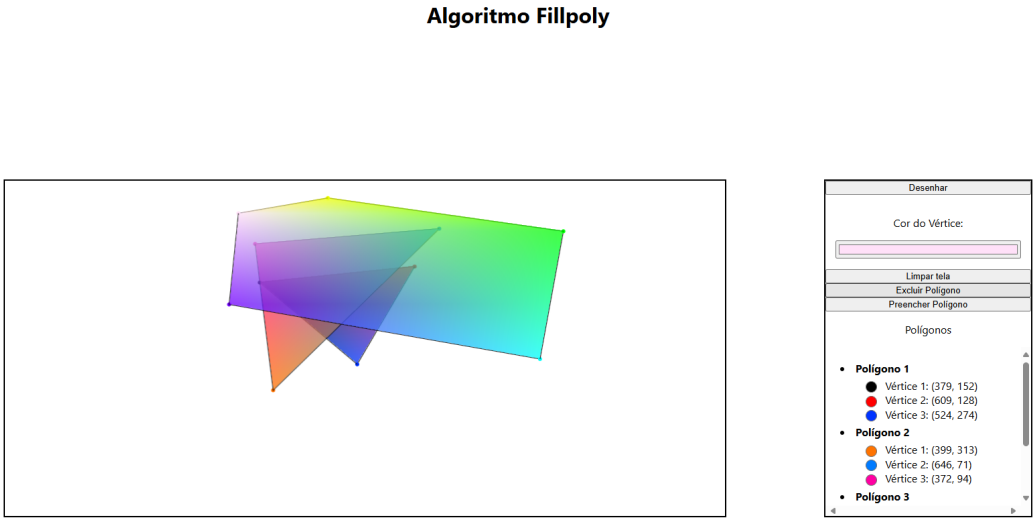

Figura 6: Seleção e exclusão do polígono dois.

- É possível recolorir um vértice individualmente. Para isso, o usuário seleciona a nova cor e clica no círculo correspondente ao vértice na lista. O polígono é atualizado instantaneamente no canvas.

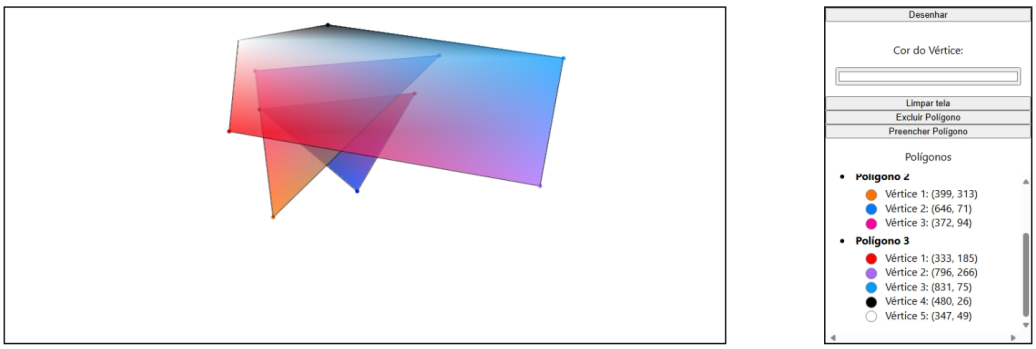

Figura 7: Alteração interativa da cor de um vértice.

2.2 Estrutura de Dados e Algoritmo Scanline (FillPoly)

Internamente, cada vértice é armazenado como um objeto JavaScript contendo suas coordenadas (x, y) e sua cor no formato RGB, facilitando os cálculos de interpolação. O processo de preenchimento segue as etapas do algoritmo *Scanline*:

- **Determinação dos Limites:** O algoritmo identifica os limites verticais do polígono, y_{min} e y_{max}, para definir o intervalo de *scanlines* que o atravessam.
- **Cálculo Incremental das Interseções:** O núcleo do método consiste em processar cada aresta do polígono para encontrar onde ela intercepta as *scanlines*. Arestas horizontais são ignoradas. Para as demais, o cálculo é feito de forma incremental:
  - Para cada aresta, é calculada uma taxa de variação constante, $T_x = \frac{dx}{dy}$, que representa o quanto a coordenada *x* se desloca para cada passo unitário em *y*.
  - A partir da primeira interseção, a coordenada *x* de cada interseção subsequente é encontrada pela simples soma $x_{novo} = x_{anterior} + T_x$.
- **Armazenamento das Interseções:** As coordenadas *x* das interseções são armazenadas em um array de listas, onde cada lista corresponde a uma *scanline* específica.
- **Finalização e Preenchimento:** Após processar todas as arestas, o algoritmo finaliza a pintura:

  - Para cada *scanline*, a lista de interseções é ordenada em valores crescentes da coordenada *x*.
  - As interseções ordenadas são agrupadas em pares, definindo segmentos horizontais (x_{ini}, x_{fim}) internos ao polígono.
  - Uma linha horizontal é desenhada de $\lceil x_{ini} \rceil$ até $\lfloor x_{fim} \rfloor$ para preencher cada segmento.

- **Interpolação de Cores:** A implementação avança a teoria ao aplicar um preenchimento com gradiente através de uma dupla interpolação linear:
  - *Interpolação Vertical:* Ao calcular as interseções ao longo de uma aresta, a cor no ponto de interseção é interpolada a partir das cores dos dois vértices da aresta.
  - *Interpolação Horizontal:* Ao desenhar um segmento horizontal, a cor de cada pixel é interpolada a partir das cores (já interpoladas) do par de interseções que delimita o segmento.

  O resultado é um preenchimento gradiente por varredura, que produz uma aparência suavizada das cores.

3 RESULTADOS E DISCUSSÃO

A implementação do projeto resultou em uma aplicação web funcional, capaz de executar as tarefas de desenho e preenchimento de polígonos conforme especificado na metodologia. Os resultados foram validados por meio de testes interativos na interface do programa.

Referências

CATARINA, A. S. *Notas de Aula*. 2025. Curso de Ciência da Computação.

GEEKSFORGEEKS. *Scan-Line Polygon Area Fill Algorithm Computer Graphics*. 2017. Disponível em: <https://www.geeksforgeeks.org/c/scan-line-polygon-filling-using-opengl-c/>.

HANGRAE, D. *Scanline Filling Algorithm*. 2024. Disponível em: <https://medium.com/@dillihangrae/scanline-filling-algorithm-852ad47fb0dd>.

RIT.EDU. *Polygon Fill Teaching Tool*. 2025. Disponível em: <https://www.cs.rit.edu/~icss571/filling/how_to.html>.