Disciplina:Computação Gráfica
Docente: Prof. Agma Juci Machado Traina

Preenchimento de Polígonos

Relatório - Trabalho Prática I

Alunos:
Luís Filipe Silva Forti - 14592348
Luiza Rodrigues Cardoso - 14593332

1. Introdução

A Computação Gráfica é uma área da Computação que estuda a síntese, processamento de imagens e visualização de dados. Tendo aplicabilidade em muitas atividades cotidianas. A Síntese de Imagem é uma área da computação gráfica que estuda a produção de reproduções visuais a partir de especificações geométricas.

Um dos tópicos que compõem a síntese de imagens é o preenchimento de polígonos, onde se compreende como as formas básicas, como linhas e formas geométricas, são reproduzidas nos sistemas gráficos.

Em suma, sabemos que os sistemas gráficos são sistemas discretos e que a partir de algoritmos incrementais e eficientes é possível renderizar primitivas básicas. Uma forma de aumentar a complexidade desta ação é analisar os métodos envolvidos na renderização de contornos (ou bordas) de polígonos e o preenchimento destes.

A partir dos conceitos apreendidos em sala de aula, o grupo trabalhou na implementação e desenvolvimento de uma interface que permite que o usuário consiga desenhar e preencher polígonos de forma arbitrária.

A divisão de trabalho entre as partes envolvidas ficaram:

- Luis: Criação do algoritmo em C++, tradução dos inputs coletados do usuário para dados utilizados pelo algoritmo, implementação de um algoritmo de anti aliasing de tipo cone para espessura do contorno.
- Luiza: Compreensão de uso da plataforma Qt Creator e especificidades desta, definição de slots, rotinas referentes ao desenho de ponto, linhas e escolha de cores, observação do mouse e configuração de janela.

2. Métodos

Com o objetivo de desenhar qualquer polígono e preenchê-lo, primeiramente seria necessário detectar seus vértices e estabelecer seu contorno. Assim se tornou necessário achar uma IDE que facilitasse a interação com o usuário.

Tendo uma forma de interação com o usuário, agora restava interpretá-la. Como o trabalho tem como foco a análise de um único polígono, ficou decidido que o usuário definiria onde ficam os vértices do polígono, com suas arestas sendo conectadas de forma automática. Para facilitar o código e tornar a interação mais simples e segura, o código conecta os vértices de forma “circular”, com todo vértice conectado com o inserido anteriormente e o último conectado sempre com o primeiro.

Sendo capaz de definir o contorno do polígono, agora surge o algoritmo foco do trabalho: o algoritmo de preenchimento de polígonos por regra de paridade.

Ele segue o seguinte algoritmo: para todo valor de Y dentro do contorno, inicia-se uma linha de varredura indo do menor X pro maior da linha. À cada iteração avança o X na linha em 1. Sempre que o X encontra ou atravessa uma aresta ele aumenta a contagem de paridade, que inicia par. Sempre que a paridade for ímpar ele irá pintar o pixel atual, assim pintando apenas dentro do polígono.

Para verificar as arestas, o algoritmo usa duas tabelas: a ET (*Edge Table*) e a AET (*Active Edge Table*). A ET é criada no início do algoritmo. Para cada aresta a tabela salva o Y máximo da aresta, o X do vértice com menor Y e o inverso do coeficiente angular m, ou seja, 1/m (equação 1). Estes dados são salvos na linha com índice igual ao valor de Y do vértice inferior, ou seja, no Y que a aresta inicia.

$$\frac{1}{m} = \frac{x_2 - x_1}{y_2 - y_1} \text{ (equação 1)}$$

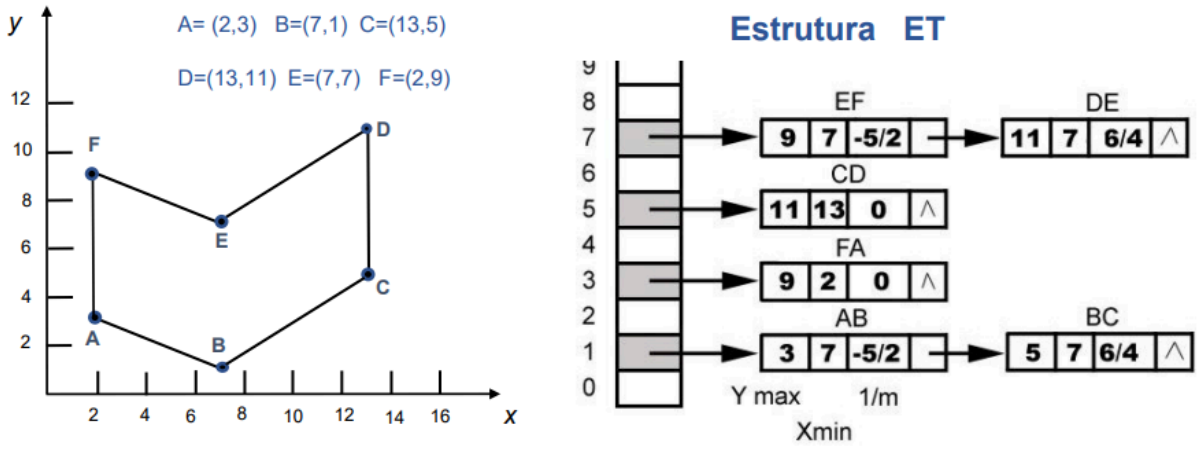


Figura 1: exemplo de tabela ET (fonte: slides da professora)

Com essa estrutura definida cria-se a tabela AET, inicialmente vazia, e iniciam-se as iterações das linhas de varredura. Ao entrar no nível da iteração atual, a tabela verifica se alguma das arestas salvas tem Ymax no nível atual, removendo-as. Das arestas que restaram, ela aumenta seu valor de Xmin pelo seu respectivo 1/m. Por fim, ela verifica na tabela ET quais arestas iniciam no nível atual, ou seja, quais tem índice igual ao Y atual.

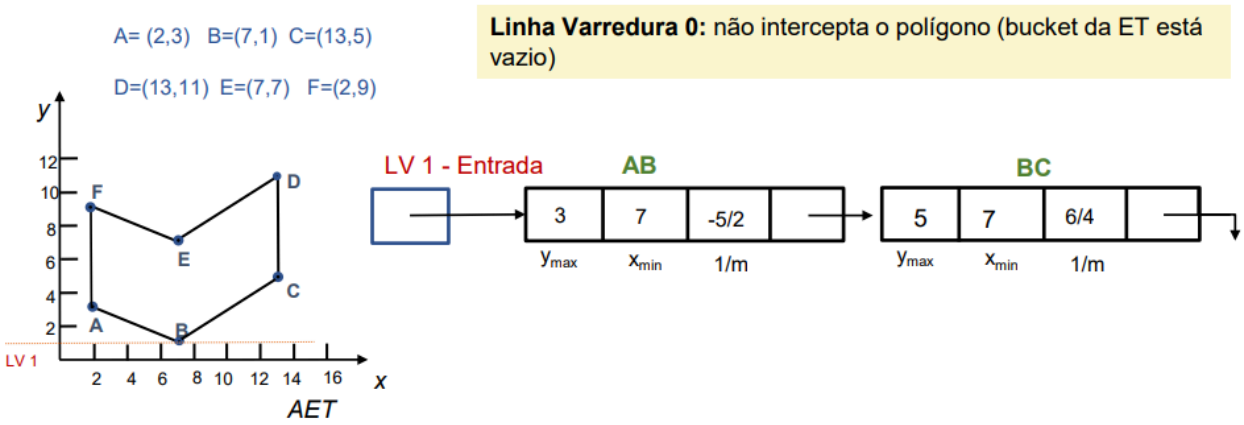


Figura 2: exemplo de tabela AET (fonte: slides da professora)

Com os dados do nível registrados, a tabela ordena eles em ordem crescente de Xmin. Desta forma ela sabe que a próxima aresta que será interceptada é sempre a próxima da tabela.

Inicia-se finalmente a pintura da linha: com X inicialmente no valor de Xmin da primeira aresta, ele aumenta em incrementos de 1, sempre verificando se atravessou novas arestas e atualizando a contagem de paridade. Sempre que a paridade for ímpar, ele desenha o pixel atual. A linha de varredura termina quando tiver atravessado todas as arestas, avançando pro próximo Y e repetindo os cálculos definidos anteriormente.

Uma consequência desse algoritmo é que os pontos nas arestas da direita e superiores nunca serão desenhadas, pois são os pontos que invertem a paridade, para as arestas da direita, ou foram eliminados por estarem no Ymax das arestas que as conectam, para as superiores.

3. Resultados

À recomendação do enunciado do trabalho, foi utilizada a IDE Qt Creator. Esta facilitou muito o desenvolvimento da interface de usuário, uma vez que possui componentes básicos que lidam com essa interação, além de usar o framework Qt que é fortemente orientado a objetos.

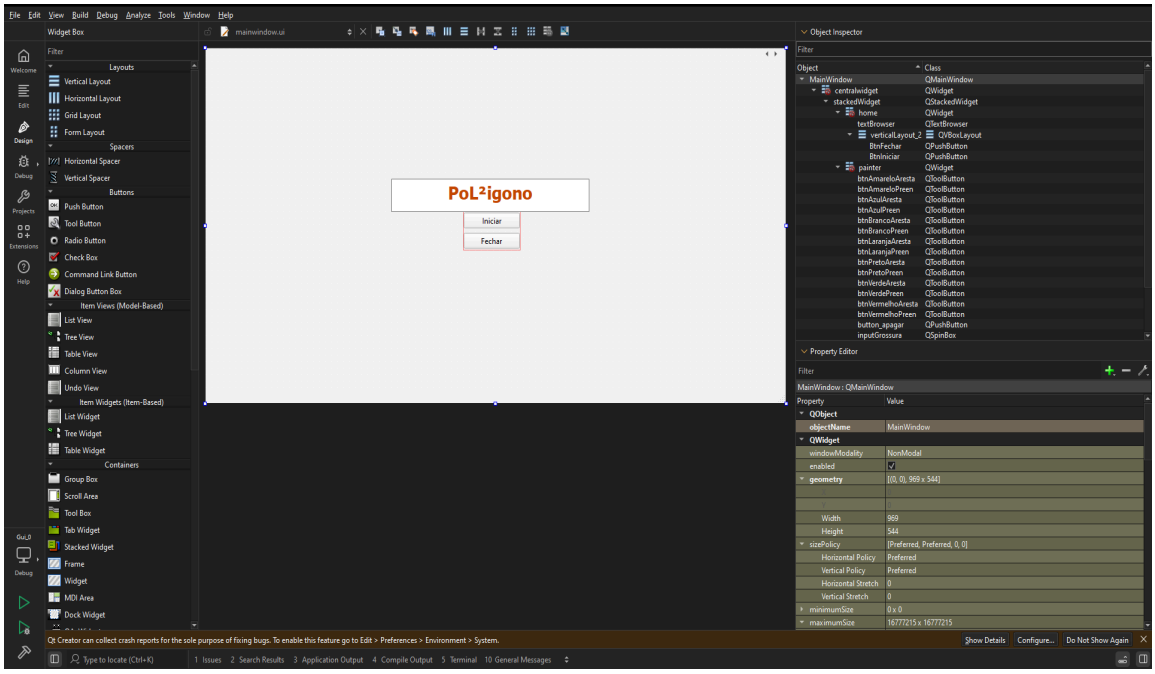


Figura 3: tela de design da interface do usuário do Qt Creator

O código foi feito em linguagem C++ e contou com o apoio das bibliotecas QMainWindow, QMouseEvent, vector e array, que são nativas do framework Qt e da linguagem de programação utilizada.

A interface conta com uma tela inicial composta por 2 botoẽs que permitem que o usuário inicie ou feche o programa (Figura 4).

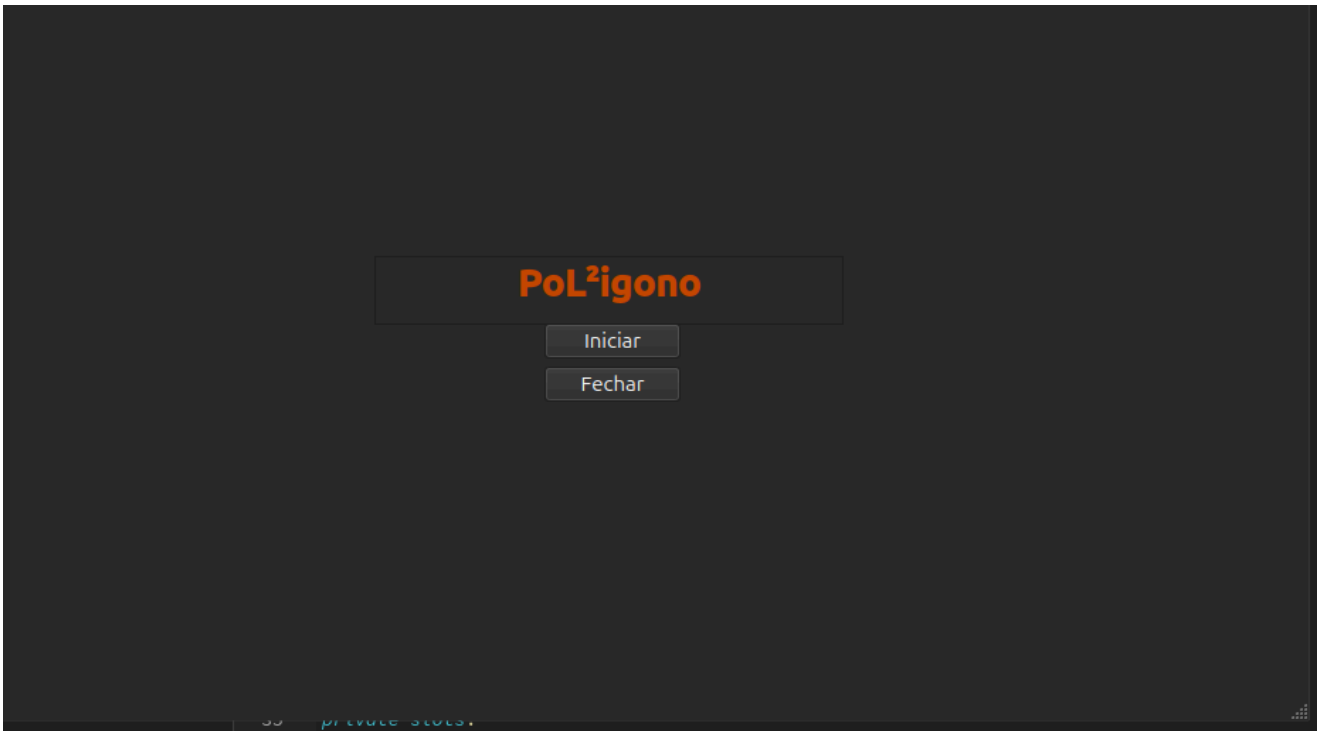


Figura 4: Tela inicial da interface.

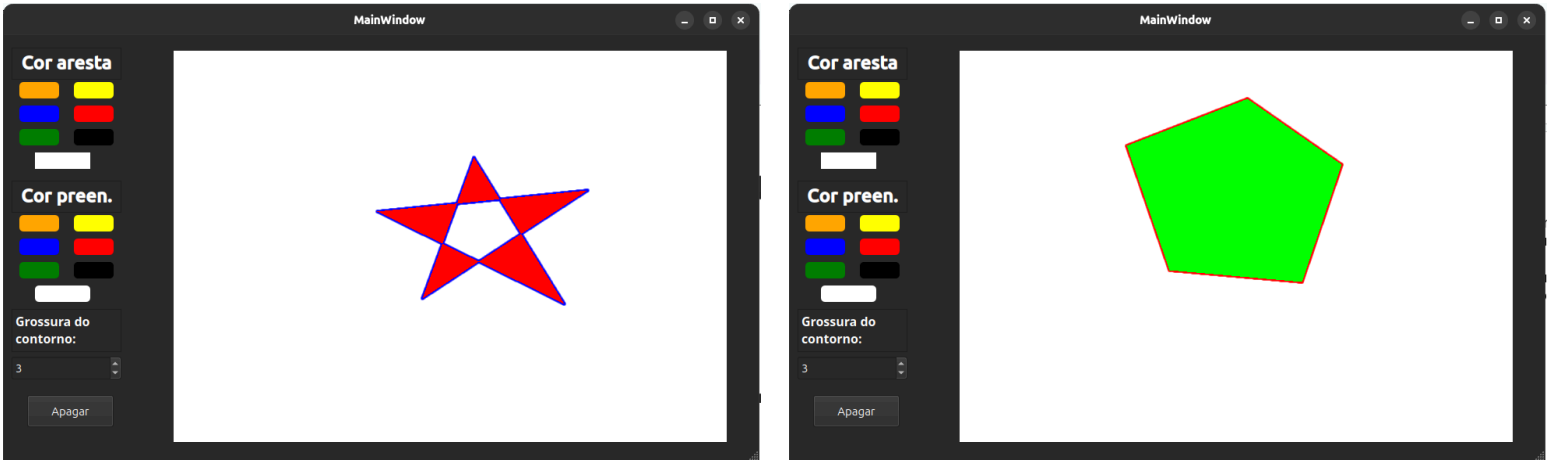
Como tela principal, há um canvas que permite que o usuário desenhe um polígono e um controle que apresenta como opções:

- 6 cores de contorno
- 6 cores de preenchimento
- Definir a espessura do contorno do polígono.

(a) (b)

Figura 5: Tela Principal: (a) demonstração polígono sem preenchimento; (b) demonstração de polígono com preenchimento e variação da espessura do contorno.

Ao testar com diversas formas de polígono, se tornou perceptível um problema: o algoritmo não lida bem com polígonos complexos (figura 6-a). Quando uma aresta corta a região interna de um polígono o algoritmo inverte a paridade, fazendo com que ele não pinte esse trecho interno. Isso infelizmente é uma consequência do algoritmo em si, mostrando que, por mais eficiente e simples que seja, ele não é a solução ideal para polígonos complexos.



(a) (b)

Figura 6: Tela Principal: (a) polígono complexo preenchido; (b) polígono simples preenchido

4. Conclusão

Durante o desenvolvimento do trabalho, o grupo teve a oportunidade de aplicar os conceitos teóricos aprendidos em sala de aula e ter como resultado uma interface que cumpriu com os objetivos delineados durante a fase inicial do projeto.

Habilidades que consideramos importantes e que foram adquiridas neste desenvolvimento foram:

- Manipulação de uma nova IDE, Qt Creator;
- Manipulação de um framework muito utilizado na área da computação gráfica e fortemente orientado a objetos.
- Desenvolvimento de um algoritmo eficiente capaz de desenhar e preencher polígonos de forma arbitrária, embora não perfeito.