TP01 - Algoritmos geométricos

Rita Rezende Borges de Lima - 2020065317

[1]Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte - MG - Brasil

ritaborgesdelima@dcc.ufmg.br

1. Introdução

O objetivo do trabalho consiste em resolver o problema da galeria de arte. É necessário vigiar uma galeria de arte com o menor número de câmeras que juntas possam observar esta por completo. A forma da galeria de arte é representada por sua planta que por sua vez é sempre um polígono simples para que cada câmera seja colocada em algum vértice deste. Para a execução do código entregue no github basta a execução de todas as células na ordem pré estabelecida.

2. Modelagem

O problema pode ser reduzido nos seguintes passos:

- A triangulação do polígono inicial por intermédio do algoritmo ear-clipping.
- A 3-coloração do grafo obtido pela triangulação.
- A escolha da cor menos presente na coloração para que os vértices com estas cores recebam as câmeras.

2.1. Classes Utilizadas

Dada a natureza geométrica do problema, na modelagem a Orientação à Objetos foi utilizada. Elementos como polígonos, pontos e triângulos foram modelados em classes da seguinte maneira:

- **Ponto:** A classe ponto contém como atributos suas duas coordenadas e funções estáticas que operam sobre instâncias de ponto.
- **Polígono:** A classe polígono contém como atributo uma lista de pontos representados seus vértices e funções que retornam listas com suas coordenadas x e y separadamente e métodos que retornam o *plot* deste no plano cartesiano.
- **Triângulo:** A classe triângulo herda da classe Polígono os métodos descritos acima. Além destes, esta contém métodos que verificam se um ponto passado como parâmetro está contido dentro do triângulo ou se é um de seus vértices e uma função estática que verifica se dois triângulos possuem a mesma face.

3. Implementação

3.1. Ear-Clipping

O algoritmo `Ear-Clipping` recebe uma instância de polígono e itera sobre sua lista de vértices até que esta tenha sido reduzida ao tamanho três. Para esta redução são sempre retirados vértices que são considerados ”pontas de orelha”, três vértices de um polígono são considerados uma orelha se dois destes formam uma diagonal. Ao se descobrir uma orelha o programa instancia estes três vértices em um triângulo e adiciona este a uma lista de triângulos. Ao final do loop, o triângulo formado pelos vértices restantes do polígono original é adicionado à lista de triângulos e esta é retornada.

3.2. 3-Coloração do grafo

Inicialmente é gerada uma lista de adjacência contendo os triângulos encontrados no ear-clipping. Para isso, para cada par de triângulo verificamos se estes têm um lado em comum e se sim adicionamos esta aresta em nossa lista de adjacência. Tendo este grafo onde cada nó é um triângulo e cada aresta representa que dois triângulos tem uma face em comum, inicialmente escolhemos um triângulo qualquer e colorimos seus vértices cada um de uma cor, a partir deste triângulo inicial fazemos uma busca em largura colorindo os vértices dos outros triângulos. Como os triângulos são processados por ordem de adjacência e um triângulo inicialmente foi todo colorido, os triângulo subsequentemente sempre terão dois vértices coloridos e um ”em branco”, já que compartilham uma face com um triângulo já processado. Assim, basta descobrir a cor restante e colorir o vértice ”em branco” com esta.
Para a representação de cada cor para vértice foi escolhido um array de tamanho n onde n é a quantidade de vértices. Cada uma de suas posições varia de 0 a 2, as cores disponíveis para a 3-coloração. Para a indexação de cada um dos vértices existe um dicionário único que converte o par de coordenadas em um inteiro variando de 0 a n.

3.3. Escolha dos vértices que receberão câmeras

De posse do array da coloração de cada vértice basta descobrir qual das 3 cores é menos comum. Os vértices que receberão câmeras serão os que foram coloridos pelo nosso procedimento anterior com a cor menos comum.

3.4. Representação de cada passo do problema

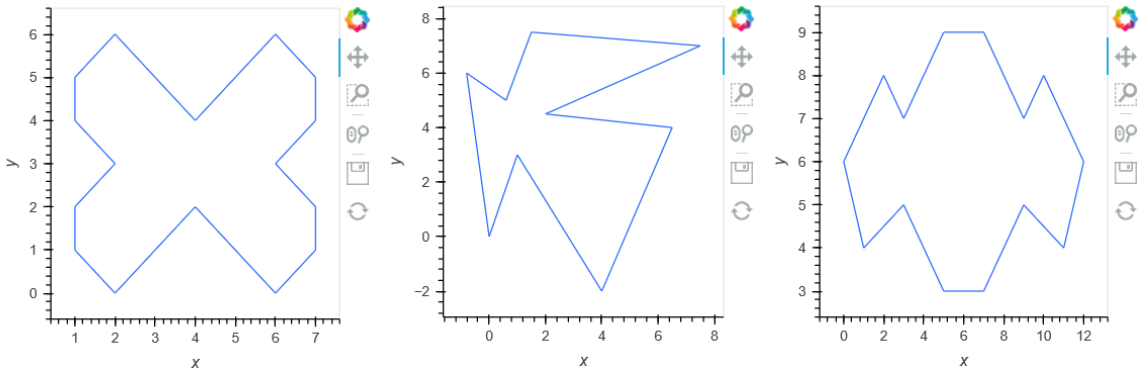
Para a representação gráfica foi utilizada a biblioteca holoviews, 3 funções de *plot* foram feitas para o programa:

- A primeira faz o *plot* da triangulação, esta recebe a lista de triângulos e para cada um deles cria um objeto com as coordenadas de cada vértice e a cor desejada e adicionada a uma lista de triângulos coloridos. Esta é passada para uma função específica do módulo holoviews que por sua vez retorna os triângulos coloridos num plano cartesiano, como visto na figura 1.
- A segunda função faz o plot dos vértices coloridos por cima da triangulação, esta recebe a lista de vértices e suas respectivas cores e coloca estes no formato para a entrada da função holoview com a cor correta.
- A terceira função é extremamente similar a de cima, contudo esta recebe como parâmetro também a cor desejada, isto é, a cor dos vértices que receberão uma câmera de segurança. Assim, no momento de processamento de cada um dos vértices, o vértice só é adicionado no plano caso tenha a cor passada por parâmetro.

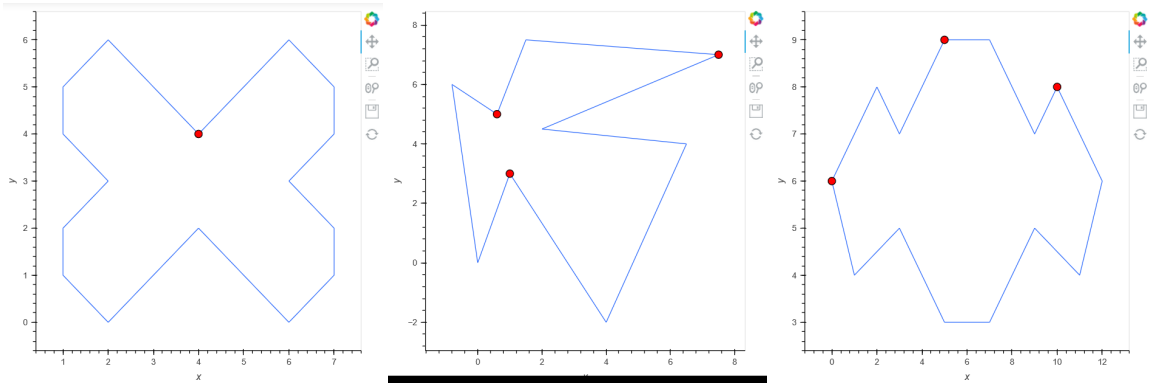
4. Conclusões

Por intermédio do algoritmo de ear-clipping e a 3-coloração do grafo gerado por este foi possível encontrar uma solução para o problema da galeria de arte que resulta em uma quantidade de no máximo n/3 câmeras onde n é a quantidade de vértices da planta da galeria. Abaixo temos alguns exemplos de casos teste e seus respectivos resultados:

Plantas das galerias:



Em vermelho onde devemos posicionar as câmeras:



5. Bibliografia

- Seção 1.6.5; Computational Geometry in C. O'Rourke
- Cap. 3; Computational Geometry Algorithms and Applications. Berg et al.
- Documentação do Holoviews, 2021.Disponível em `http://holoviews.org/user_guide/Dashboards.html`. Acesso em 10 de jul. de 2021