Trabalho Prático 1

Algoritmos 2

João Antonio Oliveira Pedrosa
Matrícula: 2019006752

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

joao.pedrosa@dcc.ufmg.br

1. Introdução

O trabalho propõe a resolução do seguinte problema:

Descobrir uma configuração de câmeras que cubra toda a área de uma galeria de arte representada por um polígono, utilizando triangulação de polígonos.

2. Implementação

Um Jupyter Notebook contendo a implementação utilizada, bem como testes e uma animação para visualização, pode ser encontrado no link:

https://github.com/mrcolorblind/tp1_alg2

Para a resolução do problema, duas classes fundamentais foram criadas, Point e Polygon:

2.1. Point

Essa classe possui apenas dois atributos, x e y, representando as coordenadas x e y do ponto. Os operadores de soma, subtração e multiplicação foram sobrecarregados para responder, respectivamente, à soma coordenada por coordenada, subtração coordenada por coordenada e produto vetorial. Duas funções que recebem a classe Point como parâmetro também foram implementadas. São elas:

- *counterClock(p1, p2, p3)*: Retorna se os pontos p_1, p_2 e p_3, seguindo essa ordem, estão em sentido anti-horário.
- *inTriangle(p, p1, p2, p3*: Retorna se o ponto p está dentro do triângulo formado por (p_1, p_2, p_3).

2.2. Polygon

Essa classe possui um único atributo, uma lista de objetos do tipo Point. A classe possui também os seguintes métodos:

- *plot()*: Desenha o polígono utilizando a biblioteca MatplotLib.
- *earClip()*: Retorna uma lista de triplas de pontos, cada uma representando um dos triângulos da triangulação do polígono. Uma explicação do algoritmo por trás desse método pode ser vista na próxima subseção.

2.3. Ear Clipping

O algoritmo de Ear Clipping funciona da seguinte maneira: Enquanto ainda houverem mais do que 3 vértices no polígono, olhamos para cada vértice e para seus dois vértices adjacentes. Se o triângulo formado por esses 3 vértices não possuir nenhum vértice interno à ele, isso significa que podemos adicionar esse triângulo à solução, remover um dos vértices e repetir o processo. O algoritmo repete esse mesmo procedimento até que sobrem apenas 3 vértices no polígono. Esses 3 vértices são adicionados à solução como o último triângulo do polígono. A complexidade, utilizando de maneira eficiente as classes implementadas é $\mathcal{O}(n^2)$, onde n representa o número de vértices do polígono.

3. Testes

Alguns polígonos foram utilizados como teste para o algoritmo. No Jupyter Notebook, presente no GitHub disponível no link referenciado no início da documentação, é possível ver uma animação do Ear Clipping de cada um desses testes. Essa animação permite um melhor entendimento do algoritmo a partir do momento em que permite a observação de todo o processo de construção do polígono triangulado. As triangulações encontradas para alguns polígonos podem ser vistas a seguir:

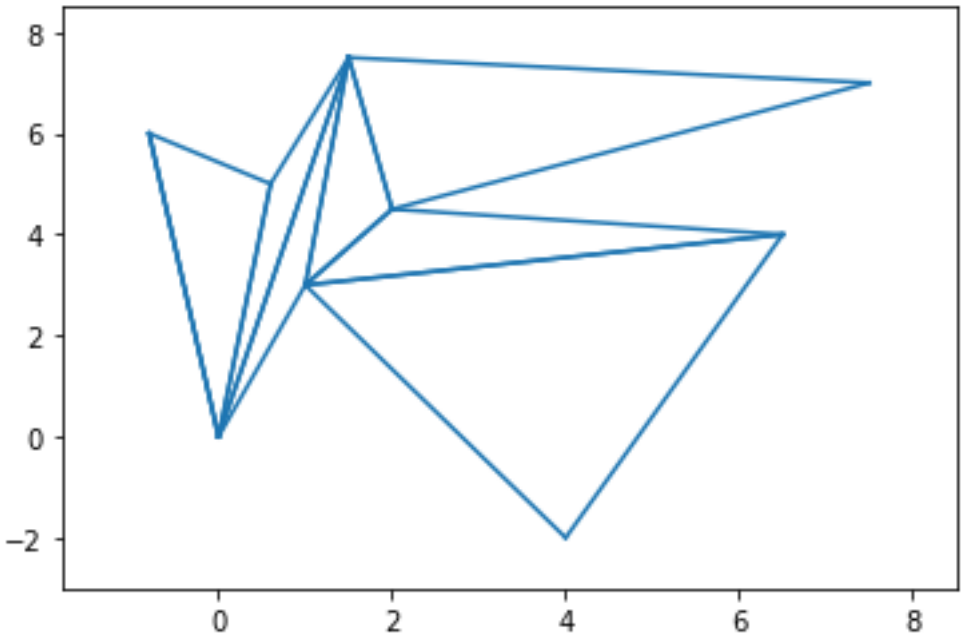

Figura 1. Triangulação encontrada para o primeiro polígono.

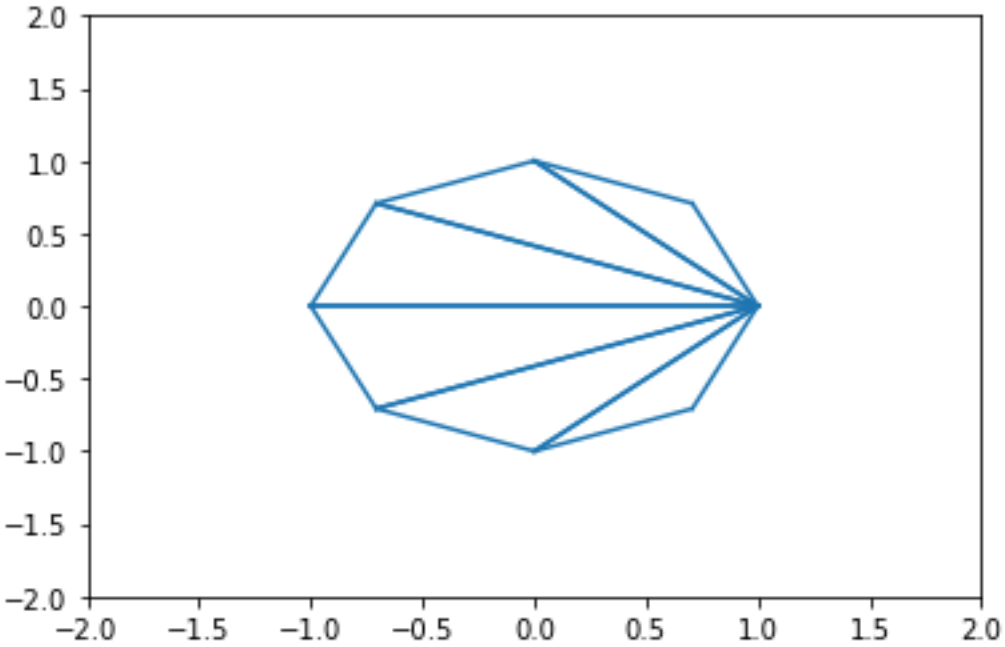

Figura 2. Triangulação encontrada para o segundo polígono.

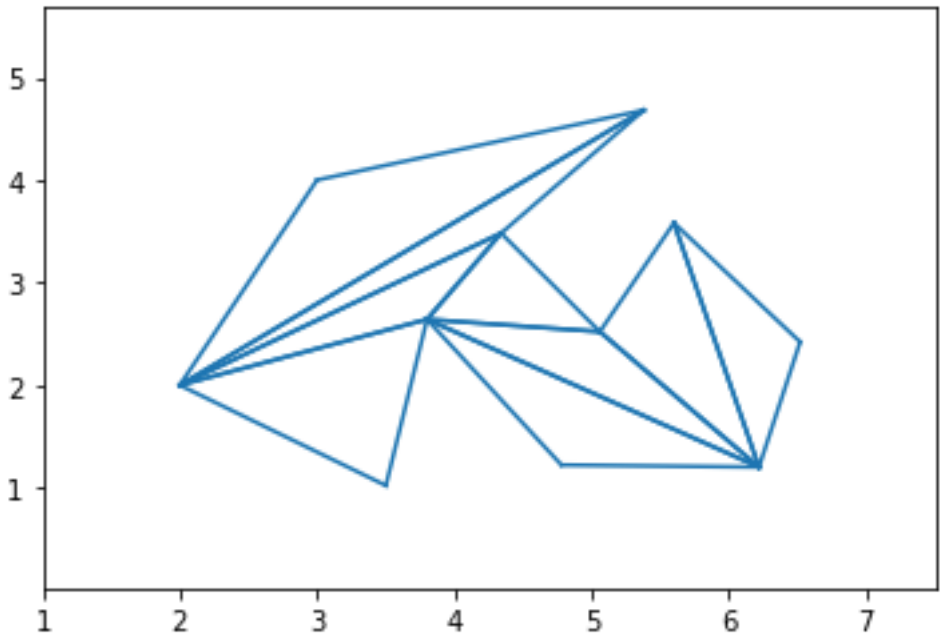

Figura 3. Triangulação encontrada para o terceiro polígono.

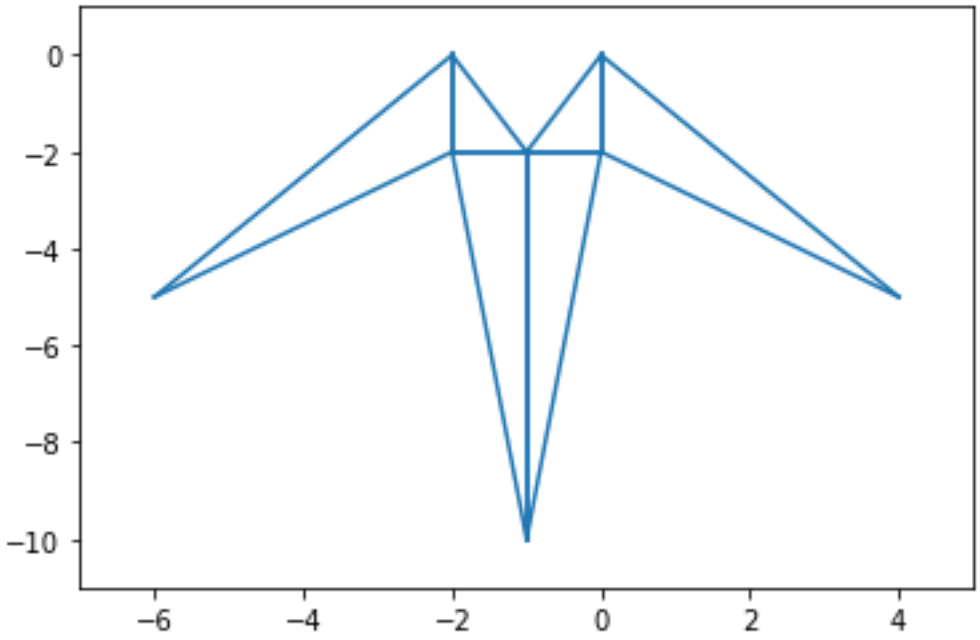

Figura 4. Triangulação encontrada para o quarto polígono.