Trabalho Prático 01 - O problema da galeria de arte
Algoritmos II

Fernando Tonucci de Cerqueira Oliveira

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

1. Introdução

O problema da galeria de arte consiste em encontrar um conjunto mínimo de câmeras que consigam ter visão em todo um espaço. Neste trabalho foi implementada uma solução em python para este problema, e todos os passos do algoritmo podem ser visualizados a partir de imagens geradas ao longo da execução.

2. O algoritmo

O algoritmo implementado é dividido em duas partes: primeiro é feita uma triangulação do polígono, e em seguida é gerado um grafo baseado nela. Em seguida, é feita uma 2-coloração do grafo, e a resposta será o conjunto de vértices coloridos com a cor que menos aparece.

2.1. Triangulação

A triangulação é feita a partir de um método ear-clipping. A ideia por trás deste método é encontrar orelhas em um polígono simple, ou seja, vértices que podem ser facilmente retirados fazendo um corte ligando seus vizinhos.

As orelhas são encontradas de forma simples a partir de uma observação geométrica: assumindo que os pontos estejam ordenados no sentido anti-horário, um vértice v será uma orelha se ele seguir duas regras:

. A aresta *vs* deve estar a esquerda de *av*, onde *a* e *s* são o antecessor, e o sucessor de v, respectivamente.

. O triângulo *avs* não pode conter nenhum outro vértice do polígono.

Encontradas as orelhas, podemos seguir com o processo de triangulação. Note que cada orelha naturalmente dará origem a um triângulo. Assim, devemos ir adicionando os triângulos oriundos das orelhas ao conjunto solução, enquanto retiramos os vértices orelha, lembrando sempre de atualizar os vizinhos dos vértices retirados para verificar se eles viraram orelha. Quando tivermos apenas três vértices restantes no polígono, já teremos a sua trinagulação.

2.2. 3-coloração

O próximo passo é formar dois grafos: um que representa o polígono triangulado, que será chamado de G, e outro em que cada triângulo será um vértice, e teremos uma aresta ligando dois vértices se os triângulos compartilharem um lado, que será chamado de G'.

Em seguida, é feito um DFS no grafo G', com o intuito de percorrer os triângulos vizinhos, e a cada passo, iremos colorir os vértices correspondentes em G.

3. Implementação

3.1. Estruturas de dados

3.1.1. Grafos

Para representar os grafos, foram escolhidos dicionários. Cada vértice do grafo será uma chave, e o valor dela será uma lista com duas informações. O primeiro elemento dessa lista é outra lista, onde serão registradas as arestas, ou seja, se tiver uma aresta ligando os vértices *v1 e v2,* o valor *v2* estará na lista da chave *v1*. A segunda informação na lista é a cor do vértice.

Esta escolha foi feita pela facilidade de criação dos dicionários, além de sua velocidade para acesso às chaves. Assim, é possível rapidamente acessar as arestas ligadas a qualquer vértice desejado.

3.1.2. Polígonos

Para representação dos polígonos foi escolhida uma lista de tuplas, em que cada tupla representa um ponto. Estes pontos devem estar ordenados, porém podem estar tanto no sentido horário, quanto anti-horário.

3.2. Triangulação

Para realizar a triangulação dos pontos, primeiro é necessário encontrar as orelhas. Para isso, o polígono deve ter seus pontos ordenados no sentido horário. Para garantir que isso aconteça, o primeiro passo feito pelo programa é detectar a ordenação dos pontos e inverter a ordem caso necessário. Isso é feito calculando a área com sinal do polígono. Caso a área encontrada seja negativa, a ordem dos pontos será invertida.

Em seguida, o programa itera por cada ponto do polígono e verifica se ele forma uma orelha. Caso uma orelha seja detectada, ela será adicionada na lista ears. Nessa lista, uma orelha é composta de tuplas de 3 pontos, em que o primeiro é o antecessor, o segundo a orelha, e o sucessor o terceiro.

Feito isso, o programa irá iterar por cada elemento de ears e o acrescentar a lista de triângulos. Em seguida, a orelha atual será retirada de ears, e o vértice orelha será retirada do

polígono, e todas as orelhas que o continham serão retiradas de ears. Por fim, os demais vértices que compõe o triângulo serão reavailiados, para verificar se eles viraram orelhas com seus novos vizinhos.

3.3. Coloração

A coloração dos vértices é feita a partir de uma busca em profundidade no grafo G'. A cada chamada desta função para um nó específico, é colorido os vértices equivalente a esse triângulo no grafo G.

A busca em profundidade foi feita a partir de uma função recursiva dfs, que passa por todos os vértices e retorna, no final, a cor menos frequente no processo de coloração.

3.4. Testes

O programa foi testado com diversos polígonos diferentes, atentando para a possibilidade de os pontos serem dados tanto no sentido horário, quanto anti-horário. O algoritmo executou sem problema em todos os casos,.

Foram deixados alguns polígonos de exemplo no código, que podem ser rapidamente testadas descomentando um trecho de código.

4. Conclusões

De modo geral, a inclusão das imagens ao longo da execução do algoritmo são de grande ajuda em seu entendimento. A possibilidade de ver passo a passo o que está sendo feito é bem interessante para problemas geométricos, que por vezes podem ser complicados de entender.