Geometria Computacional - Relatório T1

Nico I. G. Ramos

GRR20210574

Abril 2023

1 Introdução

O problema de classificação de polígonos em não simples e simples, e, estes em convexos ou não convexos, pode ser abordado de diversas formas. Para este trabalho, foi escolhido classificar os polígonos quanto a sua simplicidade através da verificação de auto-intersecções e, quanto à convexidade, utilizou-se o fato de que um polígono simples é uma *Curva de Jordan* e, a partir disso, a quantidade de vezes que uma semi-reta horizontal com origem no ponto intersecta o polígono.

O código foi implementado em C++ e estruturado nas seguintes classes: ponto, aresta e polígono. As principais funções implementadas foram a de intersecção de arestas, verificação da simplicidade de um polígono, verificação da convexidade de um polígono e a contenção de um ponto em um polígono.

Neste relatório, primeiro será discutido os principais algorítimos implementados, seguido dos testes realizados e dos resultados obtidos e, por fim, a conclusão e as considerações finais.

2 Intersecção de arestas

O algoritmo implementado para verificar a intersecção de duas arestas consiste em igualar as equações que as definem:

$$m_1 * x + b_1 = m_2 * x + b_2$$

$$x * (m_1 - m_2) = b_2 - b_1$$

$$x = \frac{b_2 - b_1}{m_1 - m_2}$$

E, para o y, utilizando a equação da segunda reta:

$$y = m_2 * x + b_2$$

Assim, têm-se as coordenadas do ponto de intersecção entre as duas retas. Para saber se este ponto é a intersecção das arestas, basta verificar se está dentro dos limites das arestas.

Entretanto, este algoritmo não funciona caso as arestas sejam paralelas ou se uma das arestas for vertical. Assim, é necessário verificar se as arestas paralelas são também colineares e possuem algum ponto em comum, e, no caso de uma aresta ser vertical, é verificado se o x da aresta vertical está contido nos limites da outra aresta e se o y de aresta horizontal está contida nos limites da vertical.

Por fim, para os dois casos, é verificado se a intersecção é um vértice de alguma das duas arestas. E, o custo desse algoritmo é $O(1)$.

3 Verificação de simplicidade

Para verificar se um polígono é ou não simples basta verificar se as arestas par a par se intersectam e se todas tem tamanho maior que zero. Como o algoritmo de intersecção de arestas trata do caso de intersecção em vértices, não é necessário realizar essa verificação neste algoritmo.

O custo desse algoritmo é $O(n^2)$, no qual n é o número de arestas.

4 Verificação de Convexidade

Para verificar se um polígono é ou não convexo é analisado o sinal do produto escalar entre todo par de arestas consecutivas. Se o sinal de todos os produtos for o mesmo, o polígono é convexo, caso contrário é não convexo.

O custo desse algoritmo é $O(n)$, onde n é o número de arestas.

5 Contenção de ponto em polígono

Para verificar se um ponto está contido em um polígono simples, uma semi-reta horizontal para a direita é traçada a partir do ponto, e são contadas quantas vezes essa semi-reta intersecta as arestas do polígonos. Caso seja par, o ponto está fora do polígono, caso seja ímpar, dentro.

O algoritmo implementado conta uma intersecção em dois casos possíveis: quando o ponto é vértice e quando a semi-reta intersecta a aresta em algum ponto que não seja um dos próprios vértices da aresta.

O custo desse algoritmo é $O(n)$, no qual n é o número de arestas.

6 Resultados

Para verificar se os algoritmos funcionam da maneira esperada, os seguintes testes foram realizados: classificar um polígono entre simples e não simples, e, para os polígonos simples, classifica-los entre convexo e não convexo e verificar a quais polígonos simples um ponto pertence.

6.1 Polígonos Não Simples

Para verificar se o algoritmo classifica corretamente polígonos não simples, foram escolhidos alguns casos de teste pensando na implementação do algorítimo de intersecção, que utiliza a equação da reta. Dessa forma, foram verificados os seguintes casos: um polígono com uma auto-intersecção e sem lados horizontais ou verticais, um polígono com uma auto-intersecção com dois lados horizontais paralelos, um polígono com uma auto-intersecção com dois lados verticais paralelos, um polígono com arestas sobrepostas, um polígono com duas auto-intersecções e um polígono de área zero. Todos os polígonos escolhidos foram classificados corretamente.

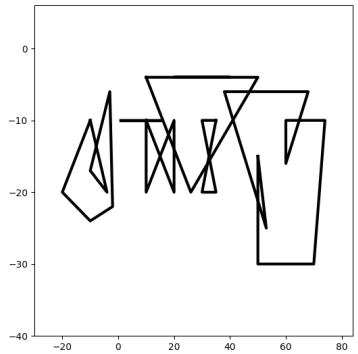

Figura 1: Polígonos não simples testados

6.2 Polígonos Simples e Convexos

Para verificar se o algoritmo classifica corretamente polígonos simples e convexos, foram escolhidos pensando na implementação do algorítimo de verificação de convexidade, que utiliza o produto escalar. Dessa maneira, os casos escolhidos foram: um retângulo, um triângulo com um lado horizontal, um triângulo com um lado vertical e um hexágono. Todos os polígonos escolhidos foram classificados corretamente.

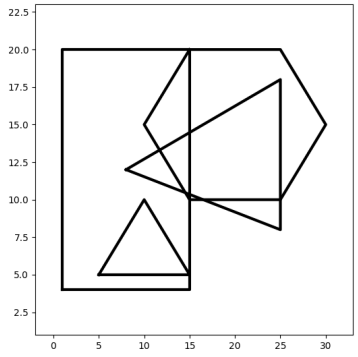

Figura 2: Polígonos simples e convexos testados

6.3 Polígonos simples e não convexos

Para verificar se o algoritmo classifica corretamente polígonos simples e não convexos, os testes também foram escolhidos pensando na implementação do algorítimo de verificação de convexidade. Dessa maneira, os testes escolhidos foram: um polígono em formato de 'L', uma estrela, um polígono em formato de 'U' invertido, uma espiral, um polígono retangular com uma região de não convexidade, um polígono triangular com uma região de não convexidade e um polígono losangular com um dos lados invertido. Todos os polígonos escolhidos foram classificados corretamente.

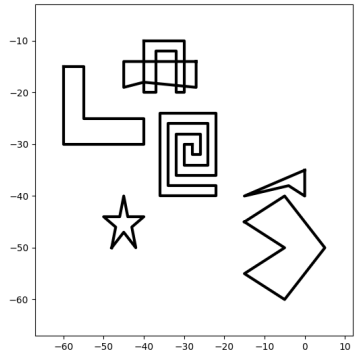

Figura 3: Polígonos simples e não convexos testados

6.4 Ponto em polígono

Para verificar se o algoritmo classifica corretamente a quais polígonos um ponto pertence, os testes foram divididos entre polígonos simples e convexos e polígonos simples e não convexos. Além disso, em ambos os casos, os testes foram escolhidos pensando na implementação do algorítimo de intersecção, principalmente na maneira com a qual ele trata os casos de arestas paralelas ou colineares, com intersecções em vértices e com o erro de ponto flutuante.

Os testes em polígonos simples e convexos foram: ponto não pertence a nenhum polígono, ponto pertence a mais de um polígono, ponto perto da fronteira, ponto em uma aresta horizontal, ponto em um vértice de um polígono e em uma aresta de outro e ponto em uma aresta de um polígono e colinear a uma aresta de outro polígono do qual ele é externo.

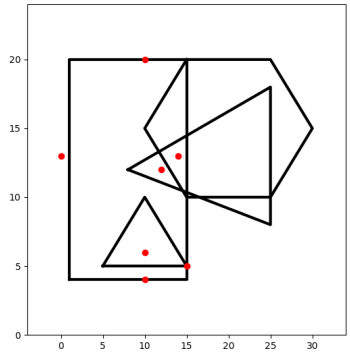

Figura 4: Ponto em polígono simples e convexo

Os testes em polígonos simples e não convexos foram: ponto é externo a todos os polígonos, ponto pertence a uma aresta inclinada, ponto em uma região de não convexidade, ponto é interno a um polígono, ponto é interno a um polígono e está na mesma altura de um vértice de outro polígono do qual ele é externo, ponto em uma região de não convexidade de um polígono e na aresta de outro, ponto é interno a dois polígonos perto da fronteira, ponto é interno a um polígono com várias curvas, ponto é um vértice de um polígono com várias curvas, ponto é externo a um polígono com várias curvas e localizado em uma região qualquer entre duas curvas do polígono ou colinear a uma aresta do polígono e também entre duas curvas do polígono.

Com estes testes, foi possível observar que o algorítimo não foi corretamente implementado, pois no caso do ponto interior a estrela e exterior ao polígono mais a direita e na mesma altura de um de seus vértices, o ponto é classifica como também sendo interior ao polígono da direita.

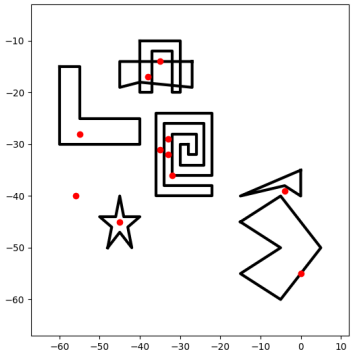

Figura 5: Ponto em polígono simples e não convexo

7 Conclusão

Os algorítimos desenvolvidos resolvem o problema proposto na maior parte dos casos. Entretanto, por conta da maneira na qual foram implementados, em alguns casos a resposta gerada não é a correta. A principal causa é o algorítimo de intersecção, que foi implementado de maneira a não considerar que duas arestas se intersectam quando o ponto de intersecção entre elas é vértice de alguma das duas.

Essa implementação funciona corretamente para a detecção de intersecção para classificar polígonos entre simples e não simples. Contudo, para detectar intersecções para verificar se um ponto é interno a um polígono ela falha. Isso acontece pois, no caso da semi-reta horizontal com origem no ponto intersectar o polígono em um vértice e em uma aresta, como o algorítimo da intersecção irá considerar que não houve intersecção no vértice, mas houve com a aresta, o algorítimo de ponto em polígono contabiliza que há apenas uma intersecção, o que o faz classificar erroneamente o ponto como sendo interno ao polígono.

Para corrigir esse problema, o algorítimo de intersecção pode ser modificado para que ele retorne o ponto onde a intersecção ocorre e deixando a cargo de quem o chamou de verificar se esse ponto deve ser considerado ou não. Dessa maneira, também seria necessário alterar o algorítimo de verificar se é simples para que ele mesmo verifique se o ponto de intersecção é vértice de uma das arestas ou não e, o de ponto em polígono, deveria ser alterado para contabilizar as intersecções nos vértices, com exceção dos casos nos quais o vértice é formado por duas arestas de sentidos opostos.

Outro problema no algorítimo de intersecção é que, ao verificar a interseção através das equações das retas, o ponto encontrado é pode não ser inteireo, o faz necessário tratar o erro de ponto flutuante nos algorítimos.