Available online www.jocpr.com

Journal of Chemical and Pharmaceutical Research, 2014, 6(5):279-283

Research Article

ISSN : 0975-7384
CODEN(USA) : JCPRC5

Several optimization strategies for CDT algorithm

Haiying Sun and Liang Ma*

Chuzhou University, Chuzhou, Anhui, China

ABSTRACT

[i]*The Constrained Delaunay Triangulation is one of the most typical algorithm in computer graphics. But it still has some limitation. Thus we want to do some changes to make it more efficient. Our main work can be divided into two parts. Firstly, we design a algorithm to improve its readability and execution efficiency. In this part, we adapt the thought of polygon's successive division and create a polygon class, for this mainly using C++ and OpenGL languages to realize. Secondly, we design another algorithm to expand its application scope. Main thought is the cross determination for selecting the eligible diagonals. Experiment showed we have achieved expectation effect to some extent.*

Key words: CDT algorithm; optimization; execution efficiency; cross determination

---

INTRODUCTION

As one of the most important topic in computer graphics, triangulation has wide application in [ii]computer graphics processing ,pattern recognition, curved surface description of three-dimensional geometry modeling system and many other fields. Planar polygons' triangulation is to divide the polygon into a series of triangles and generate no more new vertexes[1-2]. It is meaningful to study the polygon triangulation. On the one hand, as to the most simple planar graph, it is more convenient to represent, analyze and process the computer data by using triangles. On the other, triangulation is the foundation when to deal with many other issues[3]. Any polygons can be transformed into triangle meshes, therefore it has a common sense.

Currently, there are many algorithms for polygon triangulation, such as the Divide-and-Conquer Algorithm, Greedy Algorithm, Incremental Inserting Algorithm, Triangulation Network Growth Algorithm, and so on. Among all of these algorithms, the CDT(Constrained Delaunay Triangulation) proposed by Bern & Eppstein is one of the most typical one. Its main thoughts is to choose the shortest edge from all of the polygon's diagonals, then delete those that intersect with it. Keep on iteration until finish triangulation. So it is easy to understand and realize. But it still has so[iii]me limitation, that is only convex polygons can use it. And the efficiency is lower because of much deleting operation. [4]

The intent of this paper is to make some optimizing for the CDT algorithm. Therefore, the following description is not exhaustive, and does not give a detailed analysis of the theories surrounding triangulation. Our contributions can be summarized as follows:Firstly, we designed a algorithm using polygon class and realize the triangulation without deleting operation. Secondly, we designed another algorithm so that it can be used in concave polygons. Lastly, we made some test to verify our algorithms.

OPTIMIZATION I: READABILITY AND EXECUTION EFFICIENCY

In this part, we will attempt to make some changes to improve the implement efficiency. Among this the biggest change is that we design the polygon class and do without delete operation. Main steps can be showed as follows: [5-7]

(1)Create the polygon class, including the information of polygon's diagonals and vertices.
(2)Compute the length of each diagonal, sort in ascending order.
(3)Draw the shortest diagonal, therefore the polygon will the divided into two smaller ones. Respectively determine the number of the two polygons' vertices. When the number is three or less than three, the polygons have been the triangles, and then finish the triangulation. Otherwise compute the length of polygons' diagonals, sorting in ascending order. Find out the shortest one and draw it.
(4)Repeat the steps above until total number of vertices is three.

The following figures show the triangulation process.

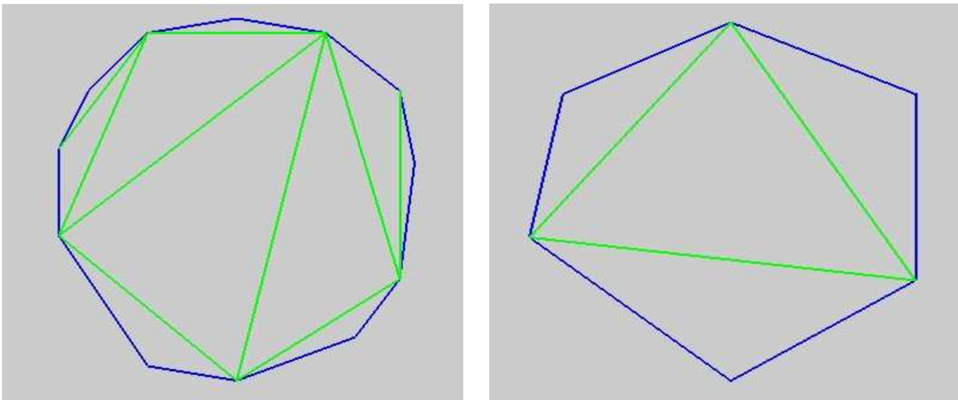

Fig.1: Polygon triangulation

The following is the class polygon we designed：

```
class PolyTriangulor {
public:
struct Diag {                                   //diagonal structure
float    length;                          //diagonal length
int        pnt0ID;                        //diagonal end 0
int        pnt1ID;                        //diagonal end 1
};
public:
typedef MathN::Vec3f Point;               //coordinates type of polygon vertices
typedef std::vector<Point> Polygon;    //list type of polygon vertices
typedef std::vector<Diag> Diags;       //list type of diagonals
PolyTriangulor(Polygon* _poly=0) : poly_(_poly) {}//constructed function
public://public functions
void renderPoly(void);
void renderDiags(void);
void renderTri(void);
void setPolygon(Polygon* _poly) { poly_ = _poly; }
private://private functions
void _assertPoly(void);
void _createDiagInfo(void);
void _sortDiags(void);
void _cullDiags(void);
private:
Diags        diags_;                                    //list of all diagonals
Polygon*    poly_;                                     //polygons to triangulate
};
```

OPTIMIZATION STRATEGY : APPLICATION SCOPE

As CDT algorithm only applies to the convex polygons, we want to do some work in order it could be used in the concave polygons. That is another important job we have done. Improved thoughts can be described as follows:[8]

Assume that the number of vertices of the polygon is N , then we can get the number of the diagonals is $N(\mathrm{N}-3)/2$. According to the definition of simple polygon triangulation, the line segment used for triangulation can be selected from all the diagonals. Main idea of our improved algorithm is to choose the eligible diagonals step by step. Firstly, we can delete those which are outside of the polygon, then delete those crossing with the side of the polygon. After these, we draw the eliminating marks on those which are in the polygon and cross with each other. When finish these tasks, we have accomplished the triangulation. Main steps are as follows.

(1)Calculate the length of the diagonals and save them in an array A, two endpoints of the diagonal are saved as well.

(2)Judge the diagonals outside of the polygon (Such as P_1P_2 in Figure 2.) and make deleting mark. We can decide whether the mid-point (or any other points except endpoints) of the diagonal is inside or outside of the polygon. If it is outside, then delete it. [9]

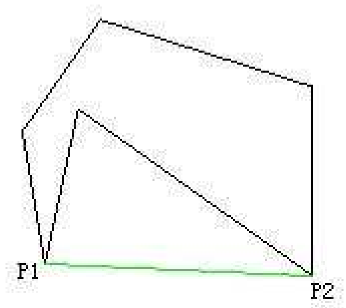

Fig.2: Illegal diagonals

(3)Select the diagonals which intersect with others and delete them. If two lines intersect, they must cross each other. We can apply the cross determination. If P_1P_2 crosses Q_1Q_2, then vectors (P_1-Q_1) and ((P_2-Q_1) must locate on the two sides of the vector (Q_2-Q_1), according to the cross determination, we can get the following expression:

$$(P_1-Q_1)\times(Q_2-Q_1) * (P_2-Q_1)\times(Q_2-Q_1)\langle 0 \qquad (1)$$

It also can be expressed as follows:

$$(P_1-Q_1)\times(Q_2-Q_1) * (Q_2-Q_1)\times(P_2-Q_1)\rangle 0 \qquad (2)$$

The limiting case is $(P_1-Q_1)\times(Q_2-Q_1)=0$ (3)

It means that (P_1-Q_1) and (Q_2-Q_1) locate in the same line, so P_1 must lie in the line Q_1Q_2; Similarly, when the following equation is true

$$(Q_2-Q_1)\times(P_2-Q_1)=0 \qquad (4)$$

It means that P_2 must lie in the line Q_1Q_2. So the condition judging whether Q_1Q_2 crosses P_1P_2 can be expressed as:

$$(Q_1-P_1)\times(P_2-P_1) * (P_2-P_1)\times(Q_2-P_1)\geq 0. \qquad (5)$$

(4)As to the diagonals which lie in the polygon, we can do as follows: sort these diagonals in an array A, make marks on the diagonals which intersect with the shortest one, and save the shortest one in another array B. Repeat the process until all the diagonals have been marked, which means we have finished selecting the needed diagonals.
(5)Lastly, diagonals stored in the array A are the needed. Draw them and then we have finished the polygon's triangulation. The following Figure 3 and 4 show the triangulation process of convex polygon and concave ones using our algorithm.

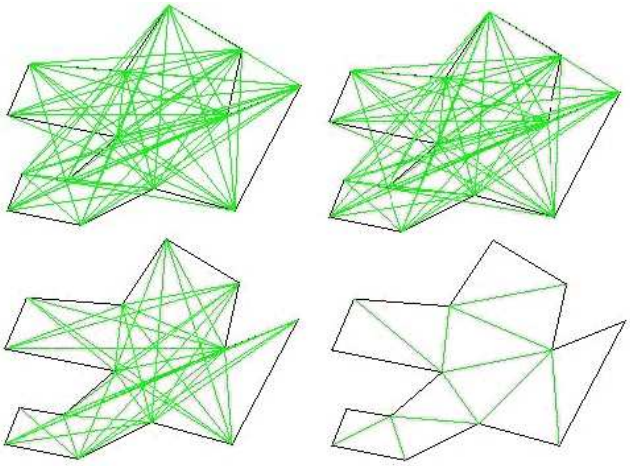

Fig.3: Triangulation of convex polygon

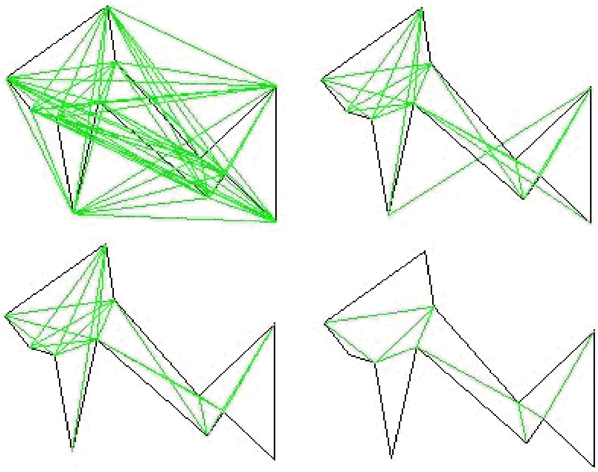

Fig.4: Triangulation of concave polygon

CONCLUSION

For our first algorithm, we adopt the thought of polygon successive division and the polygon class , use C++ and OpenGL languages, which make it easy to understand and realize. On the other hand, the execution efficiency will be improved without much deleting operation. But as CDT algorithm, it cannot be used in concave polygons.

The second we applied cross determination to the CDT algorithm. Because of this, the algorithm can break the limitation that it can be only used in the convex polygons. Instead it also can be used in the concave ones.

The quality of triangles' shape is another most directive criteria to determine whether the triangulation is well or bad. Generally, we can use ∂ evaluation factors. It can be calculated as follows: [10]

$$\partial = \frac{4\sqrt{3}S}{a^2 + b^2 + c^2} \qquad (6)$$

Generally, the ∂ is greater than 0.1. When $\partial \leq 0.1$, we can call it narrow triangle and when $\partial = 1$, it is the regular triangle and has the best shape quality.

According to this, after computing, there are none narrow triangles by our second algorithm, shown in figure 3 and figure 4. In other simple planar polygons, this is still relatively well. So it has greatly improved the triangles' shape quality, which also can be observed visually.

In the future, we will continue to optimize the CDT algorithm to make be used in more complex polygons, such as the polygons with the hole.(as shown in Figure 5) Meanwhile, we will consider how to further improve the execution efficiency as well.

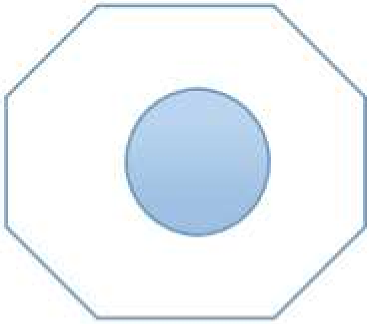

Fig.5: Polygon with a hole

Acknowledgments
Many thanks to the professor Ming-yong Pang from Nanjing Normal University, China. He has put forward many constructive ideas for us. It gives us great help and encourage with his meticulously attitude to knowledge and minded guidance, and add to the excitement of my paper. Thanks our parents and family! Your thoughtfulness is my motivation to make great achievement. Thanks our colleague Guo-zhu Zhao, he has provided a lot of support.

REFERENCES

[1] LIN T. Y. *Granular Computing: From Rough Sets and Neighborhood Systems to Information Granulation and Computing in Words. European Congress on Intelligent Techniques and Soft Computing, Heidelberg, Germany,* pp.1602-1606, September 8-12, **1997**.
[2] Xiao-hu Ma , Zhi-geng Pan, et al, *Journal of Computer Aided Design and Computer Graphics* 1, pp.1-3, **1999**.
[3] Pei-De Zhou *Computational Geometry - Algorithm Design and Analysis, Beijing,China,* pp.46-50, **2005**.
[4] Jia-wen Zhou, Zhi-xin Xue, et a, *Survey of Triangulation Methods. Computer and Modernization* 7, pp.12-15, **2010**.
[5] Zahid Raza and Deo Prakash Vidyarthi, *International Journal of Artificial Intelligence* 3,pp.86-106, **2009**.
[6] M. Bern, and D. Eppstein, *Mesh Generation and Optimal Triangulation, Technical Report CSL-92-1, Xerox PARC*, **1992**.
[7] Kong X S, Everett H,Toussaint G T, *Pattern Recognition Letters* 11, pp.713-716, **1990**.
[8] Chaiwat Kosapattarapim, Yan-Xia Lin and Michael McCrae, *International Journal of Mathematics and Statistics* 12,pp.1-15, **2012**.
[9] Jian-bo Ren, Zengyi Ma, *Journal of Gansu Science* 9,pp. 55-60, **2007**.
[10] Li Che, *The Research of Adaptive Meshing Based on Moment Method in Communication Vehicular System, M.S thesis, XiDian University, Xian, China,* pp.25 , **2010**.