A RECURSIVE ALGORITHM FOR TRIANGULATION OF ARBITRARY POLYGON BASED ON BSP TREE

Qiang LIU

Sichuan Bureau of Surveying and Mapping, 198, Sect. 2, Renminbei Road, Chengdu, China, 610081
liuqiang_em@sina.com

Commission II, WGII/6

KEY WORDS: Triangulation; BSP tree; Polygon; 3D GIS; Recursive algorithm

ABSTRACT:

Triangulation of polygon is of very importance in three-dimensional software applications. An algorithm for triangulation of polygon based on binary space-partitioning (BSP) tree idea is proposed in this paper. That is, this algorithm implements triangulation of poly-gon with recursive method according to BSP tree structure. It is suitable for not only simple polygons with arbitrary convex or conca-ve shapes but also complex polygons with islands. Moreover, taking into account elevation, it is also completely suitable for three-di-mensional representation for long-distance streams with terrain undulation.
Triangle is the simplest polygon in computer graphics. Most polygonal surfaces are rendered via triangles in many popular 3D rendering Engines such as OpenGL and Direct3D. Therefore, polygons must be correctly triangulated before rendering.
Some algorithms for triangulation of simple polygon were put forward. Nevertheless, they could not be applied perfectly for polygons of buildings, roads and rivers, and so on, in 3D Geo graphical Information Systems (GIS) on account of polygonal terrain feature and complexity.
A recursive algorithm for triangulation of polygon based on binary space partitioning (BSP) tree is proposed in this paper. That is, this algorithm implements triangulation of polygon with recursive method according to BSP tree structure. It is suitable for not only simple polygons with arbitrary convex or concave shapes but also polygons with islands. Moreover, taking into account elevation, it is also completely suitable for three-dimensional representation for long-distance streams with terrain undulation.

1. PARTITION PRINCIPLE

For a polygon with n vertices, correct triangulation should conform to principles as follows:

（1） Each triangle vertex should also be one of the polygon's;
（2） All triangles could not intersect the polygon;
（3） All triangles should lie inside the polygon;
（4） Triangle number should be n-2;

We first introduce related concepts in this context as follows:

Intersection between two line segments: If intersection point lies in between two endpoints of two line segments individually, then there exists intersection between the two line segments, otherwise, if not, called no intersection.

Line segment inside a polygon: If two endpoints of a line segment lie inside a polygon, then the line segment is in the polygon. Similarly, if a triangle is called inside a polygon, it means all the boundaries of this triangle are either on or inside the polygon. If not, we call that the line segment or triangle is outside of the polygon.

Correct triangle: If a triangle not only lies inside a polygon, but also does not intersect the polygon, then it is called a correct triangle.

2. TRIANGULATION ALGORITHM

First, we discuss the algorithm idea as follows:

2.1 Algorithm Idea

2.1.1 BSP Concept

In data structure theory, the tree structure characteristics are: it is a node set starting from the only start node called root one. A node may be regarded as a parent, whi- ch points to zero, one or more nodes called children. A node w- ithout child is called leaf. The root node has no parent.

BSP tree has such particular characteristics: each node has zero, one or two children, left node is called left child, and right node is called right child.

2.1.2 BSP Tree Structure of Triangulation

There might be many possible results of triangulation of polygon. After carry- ing out comprehensive analyses and studies, the author draws the conclusion that polygon could be triangulated according to BSP tree idea, that is, a polygon may be subdivided into a trian- gle mesh according to BSP tree structure.

We will discuss how to triangulate a polygon according to BSP tree structure below.

First, we set up a correct triangle, the root node of the BSP tree, with two adjacent vertices and another vertex in the polygonal vertex sequence. Obviously, the root triangle divides the primitive polygon into one or two sub-polygons except for this triangle, which lie in two sides of the triangle respectively. Then, we subdivide the two sub-polygons, respectively, thus, to generate left and right child triangles of the root triangle.

Further, recurs ively, grandchild, great-grandchild, and so on, could be built up. If current triangle has no child, that is, it is a leaf node, then we will return up to its parent node. If this parent node is left node of its last level node, then we will build up its right child subtree recursively down, otherwise, we will return up to its last node.

We will describe the algorithm idea below in detail. Assume that polygonal boundary to be partitioned is made up of a sequence with n vertices such as *P0*, *P1*, *P2*, *P3*, *P4*, …, *Pn-1*, etc.. The procedure is as follows:

(1) Set up root node triangle. First, let $V_1 = P_0$ and $V_2 = P_0$. Suppose alternatively that V_3 (called active vertex in this context) is one of the remaining vertices in the sequence, if vertices V_1, V_2 and V_3 compose a correct triangle, then, the root node of the BSP tree is gener-ated.

(2) Determine whether the current triangle has a child one or not. If V_3 is adjacent to V_1, then this root node triangle has no left child node, on the contrary, it do-es; similarly, right child node may be determined via adjacent relationship between V_3 and V_2.

(3) Set up child triangle. Assume that current triangle has a left child triangle. If V_2 lies in left side of line segment V_1V_3, we could generate a new correct triangle in the right sub-polygon of the primitive polygon, called current triangle's left child triangle. Similarly, we could also generate right child triangle.

(4) Determine whether current new triangle is a leaf node or not. If it is, then we return up to the last level node, otherwise, we repeat step 2, 3, and 4, and build its child triangle. Recursively successively, finally the triangle mesh of the polygon triangulated could be set up.

2.2 Example

As shown in Fig. 1, we assume that a polygon is made up of twenty-nine vertices (P_0, P_1, P_2, P_3, …, and P_{28}, etc) . First, let$V_1 = P_0$ and $V_2 = P_1$. Then alternatively suppose $V_3 = P_{28}$, P_{27}, P_{26}, P_{25}, etc, until $\Delta V_1V_2V_3$ is a correct triangle. Thus, we generate a new correct triangle. Lastly, we could build up the triangle mesh via BSP tree idea.

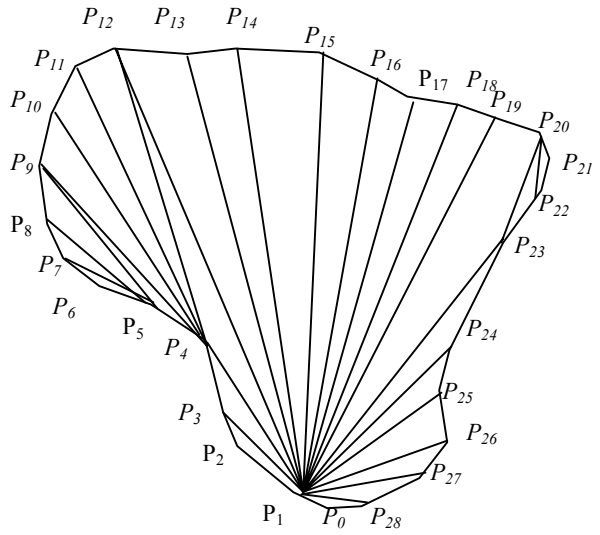

Figure 1. Demonstration of triangulation based on this algorithm

According to BSP tree structure, the order of triangles generated above is: $\Delta P_0P_1P_{28} \rightarrow \Delta P_1P_{28}P_{27} \rightarrow \Delta P_1P_{27}P_{26} \rightarrow \Delta P_1P_{26}P_{25} \rightarrow \Delta P_1P_{25}P_{24} \rightarrow \Delta P_1P_{24}P_{23} \rightarrow \Delta P_1P_{23}P_{20} \rightarrow \Delta P_1P_{20}P_{19} \rightarrow \Delta P_1P_{19}P_{18} \rightarrow \Delta P_1P_{18}P_{17} \rightarrow \Delta P_1P_{17}P_{16} \rightarrow \Delta P_1P_{16}P_{15} \rightarrow \Delta P_1P_{15}P_{14} \rightarrow \Delta P_1P_{14}P_{13} \rightarrow \Delta P_1P_{13}P_{12} \rightarrow \Delta P_1P_{12}P_4 \rightarrow \Delta P_1P_4P_3 \rightarrow \Delta P_1P_3P_2 \rightarrow \Delta P_{12}P_4P_{11} \rightarrow \Delta P_4P_{11}P_{10} \rightarrow \Delta P_4P_{10}P_9 \rightarrow \Delta P_4P_9P_5 \rightarrow \Delta P_9P_5P_8 \rightarrow \Delta P_5P_8P_7 \rightarrow \Delta P_5P_7P_6 \rightarrow \Delta P_{23}P_{20}P_{22} \rightarrow \Delta P_{20}P_{22}P_{21}$.

3. TRIANGULATION OF POLYGON WITH ISLANDS

We discuss algorithm for triangulation of simple polygon above. Similarly, we still could use BSP tree idea to triangulate polygon with islands, such as streams. As shown in Fig. 2, we assume the boundary of polygon *A* with island *B* includes an internal boundary and an external boundary. The triangulation idea is illustrated as follows: first, we search for and find an external boundary vertex, whose connection line with a certain vertex ofthe internal boundary does not intersect the two polygonal boundaries. For example, we produce line segment *l* in Fig. 2 by connecting vertex P_0 with P_9, thus, polygon *A* with island B is changed into a new polygon without any island, and the former two vertex sequences are merged. New vertex sequence is in order: P_0, P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_0, P_9, P_{16}, P_{15}, P_{14}, P_{13}, P_{12}, P_{11}, P_{10} and P_9, etc. Here, line segment *l* may be considered as two boundary line segments of the new polygon. After we perform above conversion, then we apply the method for triangulation of simple polygon discussed above, thus, we also could implement triangulation of polygon with islands successfully.

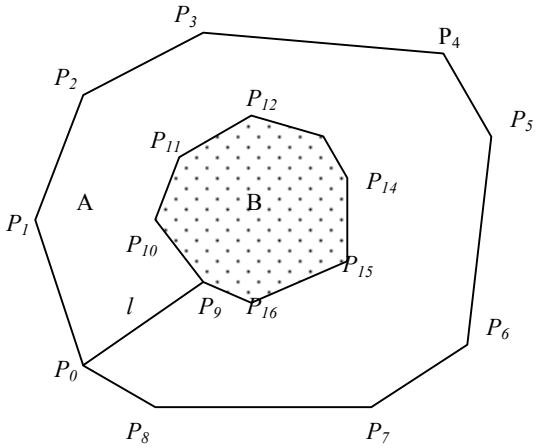

Figure 2. A polygon with an island

4. TRIANGULATION OF UNPLANAR POLYGON AND OPTIMIZATION FOR GRAPHICS SHAPE

4.1 Triangulation of Unplanar Polygon

We discussed triangulation of planar polygon emphatically above. Nevertheless, in three-dimensional applications, especially in 3D GIS applications, some surface-shaped objects, such as roads, streams, etc., are not planar polygons but unplanar ones. If we accomplish triangulation of these simple unplanar surfaces with above algorithm, it will lead to incorrect representation relative to reality.

So, for unplanar polygons, such as roads and streams, etc., to avoid incorrect representation, we should add an extra condition

taking into account relief feature to above algorithm for triangulation of planar polygon. For example, to generate a child triangle of current triangle, in the above algorithm for triangulation of planar polygon, we regard the vertex first composing a correct triangle with vertices V_1 and V_2 as active vertex V_3. Nevertheless, in the process of selecting active vertex V_3 for triangulation of unplanar polygon, first, we generate a point set of all remaining vertices forming correct triangles with vertices V_1 and V_2 from the vertex sequence. Then, we regard the vertex whose elevation value is the closest to the average value of vertices V_1 and V_2 among the point set as active vertex V_3. Thus, vertices V_1, V_2 and V_3 constitute $\Delta V_1V_2V_3$. For example, as shown in Fig. 3, when starting triangulation, let $V_1 = P_0$ and $V_2 = P_1$, this moment, P_2, P_3, P_4, P_5, P_6, P_7, P_9, P_{10}, P_{11}, and P_{12} could alternatively form correct triangle with V_1 and V_2. These active vertices form a point set. As the elevation value of Vertex P_5 is the closest to the average value of V_1 and V_2 among this point set, we regard vertex P_5 as active vertex V_3, and then generate $\Delta V_1V_2V_3$. Thus, The rendering effect of the triangle mesh generated with improved BSP algorithm is smooth and real enough.

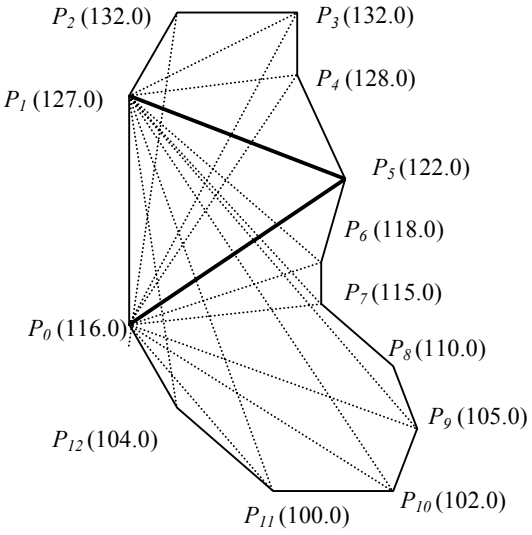

Figure 3. An unplanar polygon

4.2 Optimization for Graphics Shape

In practical applications, we need to optimize graphics shape preliminarily. For example, a building top vertices digitized with Digital Photographical Workstation (namely DPW) should be situated in the same plane in theory, however, for the sake of elevation errors produced in the process of digitizing, they are not. Thus, rendering effect is not satisfactory. So, the set of the polygonal boundary vertices need to be preprocessed, that is, if the elevation errors of adjacent vertices are within a limit, then they will be adjusted to the same plane.

5. APPLICATIONS

This algorithm could be further used for building modeling in addition to generating triangle mesh of area-shaped objects. First, we generate triangle mesh with its top polygon data digitized with DPW, and then convert them into a 3ds file.

Finally, we could edit, map this model in 3DS Max software, and then convert it into X file format used in Direct 3D rendering engine, or other formats for further applications. Thus, for the development of 3D GIS software, some modules, such as building model editing, could be replaced with 3DS Max software, we may make the development of three-dimensional software easier than ever.

6. CONCLUSIONS

Triangulation of polygon is of very importance in 3D GIS. A recursive algorithm for triangulation of polygon based on BSP tree idea is proposed in this paper. This algorithm uses BSP tree idea. It is close in theory. It could be in common use. Therefore, planar polygons with arbitrary convex or concave shapes could be correctly triangulated with this algorithm. Furthermore, it is suitable for complex polygons with islands, and unplanar surface-shaped objects with relief undulation. So, it is rather valuable in three-dimensional GIS applications.

ACKNOWLEDGMENTS

The research described in this paper was funded by '(01) 0302 Funded by Open Research Fund Program of LIESMARS' founded by National Laboratory for Information Engineering in Surveying, Mapping and Remote Sensing, China. In particular, I would like to thank my advisor Professor Li Deren. Moreover, I am thankful to Professor Zhuqing for his help.

REFERENCES

Arkin E. M., Held M., Mitchell J. S. B., and Skiena S. S., 1994. Hamiltonian Triangulations for Fast Rendering. *Second Annual European Symposium on Algorithms*, 855, pp.36–47.

Evans F., Steven S. Skiena, and Varshney A., 1996. Optimizin-g Triangle Strips for Fast Rendering. In *IEEE Visualizat-ion'96 Proceedings*, California, pp. 319–326.

MA Xiao-Hu, PAN Zhi-Geng, SHI Jiao-Ying, 1999. Delaunay. Triangulation of Simple Polygon Based on Determination of Convex-Concave Vertices. *Journal of Computer Aided Design and Computer Graphics*. 11 (1), pp.1-3.

Siu-Wing Cheng, Jae-Sook Cheong, 2001. A Triangulation for Optimal Strip Decomposition in Simple Polygons. http://www.cse.cuhk.edu.hk/~acm-hk/activity/pg/ust-jscheong.pdf.

YANG Jie, *2000*. Triangulation of Simple Polygon Based on Det-ermination of Convex-Concave Vertices. *Mini-Micro System.*, 21 (9), pp.974-975.