The Generation of Pseudo-Triangulated Spiral Polygon Using Convex Hull Layers

F. Taherkhani[1], **A. Nourollah**[1,2]
[1]Department of Computer Engineering & IT, Islamic Azad University, Qazvin, Iran
[2]Department of Electrical & Computer Engineering, Shahid Rajaee Teacher Training University, Tehran, Iran

Abstract - *The generation of random simple polygon and the pseudo-triangulation of a polygon are regarded as the proposed problems in computational geometry. The production of a random polygon is used in the context of the consideration of the accuracy of algorithms. In this paper, a new algorithm is presented to generate a simple spiral polygon on a set of random points S in the plane using convex hull layers in a way that pseudo-triangulation is also performed on it simultaneously. The new algorithm can be done in O(nlogn) time, so it is considered as one of the optimal algorithms.*

Keywords: simple spiral polygon, pseudo-triangulation, convex hull layers, convex and concave chain

1 Introduction

Polygons are suitable shapes to demonstrate the objects of real world and every object in nature is demonstrable as a set of polygons. The generation of random simple polygons has two main areas of application: a) testing the correctness and b) evaluating the CPU-time consumption of algorithms that operate on polygons.

The generation of random geometric objects has received some attention by researchers. Epstein studied the uniformly random generation of polygon triangulation [2]. Polygon pseudo-triangulation is a generalized form of polygon triangulation. The names pseudo-triangle and pseudo-triangulation were coined by Pocchiola and Vegter in 1993[3]. A pseudo-triangle is a simple polygon with exactly three convex vertices, called corners and three concave chains of edges joining the corners [4].

Let S be a set of random n points $p_0,\ldots, p_{n-1}$ in the plane. The goal is generation of a random simple polygon with a uniform distribution. A uniformly random polygon on S is a polygon generated with probability of $1/k$ if there exist k simple polygons on S in total [2, 5]. Since the generation of a polygon from a set of random points is frequently used to consider the performance of proposed algorithms in the context of polygons such as the Art gallery problem; therefore, the generation of similar polygons cannot show well the quality of the performance of the algorithms. Thus, we are looking for algorithms having the production ability of kinds of polygons with different structures, and up to now no solutions with the polynomial time to generate uniform random polygons have been known.

The following subjects of this paper have been organized in this way: In section 2 the initial definitions are presented. In section 3 the generation manner of convex hull layers are expressed. In section 4 the suggested algorithm to generate pseudo-triangulated spiral simple polygon and the analysis of its time complexity is proposed. Finally in section 5, conclusion will be presented.

2 Preliminaries

A sequence of line segments, such that the end of each one is the beginning of the following one, is referred as polygon and a polygon whose edges don't intersect one another is called *simple polygon*.

A simple polygon is called a *convex polygon* when all the internal angles are less than π. According to this definition, the set of points S on a plane is called convex if and only if in exchange for both the points $p,q \in S$, the line segment pq completely lies inside S ($pq \subseteq S$).

The most applicable structure in robatic geometry is *convex hull*. Convex hull of the given points $p_0,\ldots, p_{n-1}$ is the smallest convex set on the plane which contains the points.

Let three points $p_1(x_1, y_1)$, $p_2(x_2, y_2)$ and $p_3(x_3, y_3)$ are given in the plane. Hence matrix A is defined as follows:

$$A = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \quad (1)$$

Let $det\ (A)$ refers to determinant of matrix A. Three cases can be occurred.

- *Case a:* $det\ (A) > 0$, Sequence p_1, p_2, p_3 are *counter-clockwise* (left turn).

- *Case b: det (A) < 0*, Sequence p_1, p_2, p_3 are *clockwise* (right turn).
- *Case c: A = 0* implies that the three points p_1, p_2, p_3 are *collinear*.

Two points p and q on the Euclidean plane are *visible* towards each other if the line segment pq doesn't intersect any other line segments.

Let $p_0,\ldots, p_{n-1}$ be the vertices of a simple polygon P which lie in counter-clockwise direction (Fig. 1). We call $\delta(p_i,p_j)$, the shortest path between the two vertices p_i, p_j from the vertices of P. $\delta(p_i, p_j)$ path is called convex chain If we move from vertex p_i towards vertex p_j on the path, the relevant path will be counter-clockwise, otherwise the $\delta(p_i, p_j)$ path is called concave chain.

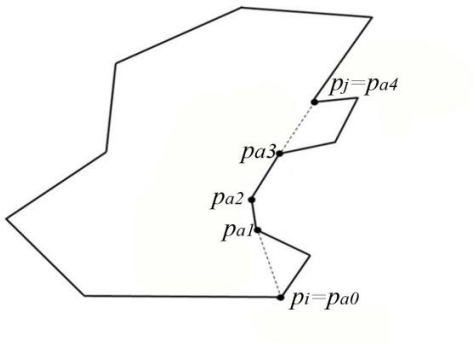


Fig. 1 Convex and concave chain

3 The generation of convex hull layers

In this section we will consider algorithm of the generation of convex hull. In order to generate convex hull, the coordinates of vertices and the order of their connections are required. There are different algorithms in order to generate convex hull. In this paper the Graham algorithm version *B* has been used. In order to compute convex hull, first one should find boundary points. In this algorithm, the lowest point is the first starting extreme point.

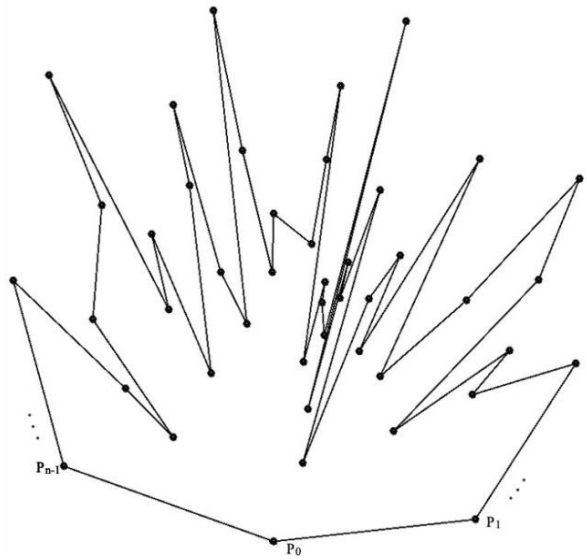
The set S with n points on a plane is given. According to Graham scan algorithm version *B*, the following steps are taken:

- *Step 1*- Find the lowest point and call it point p_0.
- *Step 2*- The remaining points are put in order based on the angle around point p_0. If two points have the same angle with p_0, (i.e. they are collinear) then the point which has a larger distance from p_0 is taken into consideration. We call these points $p_1,\ldots, p_{n-1}$ and connecting these points to one another generating a star shaped polygon (Fig. 2-a).
- *Step 3*- Line segment p_0p_1 definitely lies on the convex hull. Thus these two vertices are pushed into a stack so that p_1 lies on the top of the stack. Two top of stack vertices together with the following vertex (p_2) are considered and the clockwise or the counter-clockwise directions of these consecutive three vertices are determined. If the angle is counter-clockwise, the vertex will be pushed into the stack and the next vertex is considered, otherwise the top of stack is popped and similarly the algorithm is continued. Eventually, all the vertices which lie on the stack are the same vertices sorted on the most external convex hull layer. The time complexity of the presented algorithm is *O(nlogn)* (Fig. 2-b).

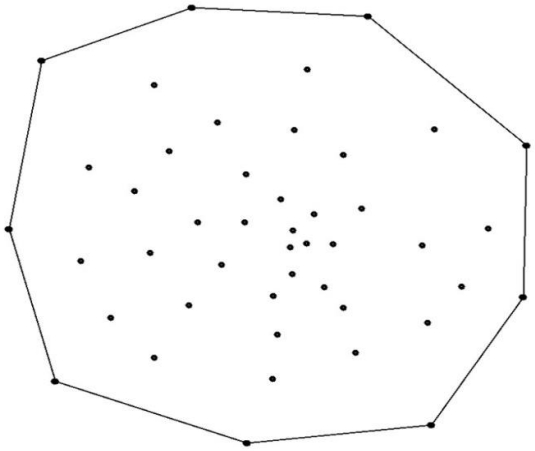
Pseudo code of the Graham algorithm version B:

Procedure Graham

$p_0 \leftarrow$ find the point whose y coordinate is minimum
Sort the other points around p_0 and call them $p_1,\ldots,p_{n-1}$
Push (p_0)
Push (p_1)
for $i \leftarrow 2$ ***to*** *n-1* ***do***
  while Right (stack [top-1], stack [top], p_i) ***do***
    Pop
  Repeat
  Push (p_i)
Repeat



(a)



(b)

Fig. 2 (a) Star shape polygon (b) the most external convex hull layer

By extracting the convex hull points, the algorithm is repeated on the remaining points, a new convex hull is generated and this action goes on until it comes to less than three points. This means that just one or two points remains. Hence, the convex hull layers are generated. In the following section these generated convex layers will be typically used. It means that these layers are not depicted for the set of points *S* on the plane and they are computed as preprocessing (Fig. 3).

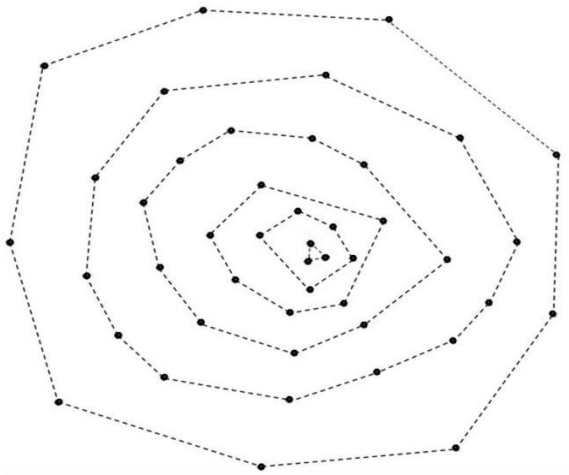


Fig. 3 The convex hull layers

4 The generation of pseudo-triangulated spiral simple polygon

In this section a new algorithm is presented for the generation of spiral simple polygon which is also pseudo-triangulated simultaneously.

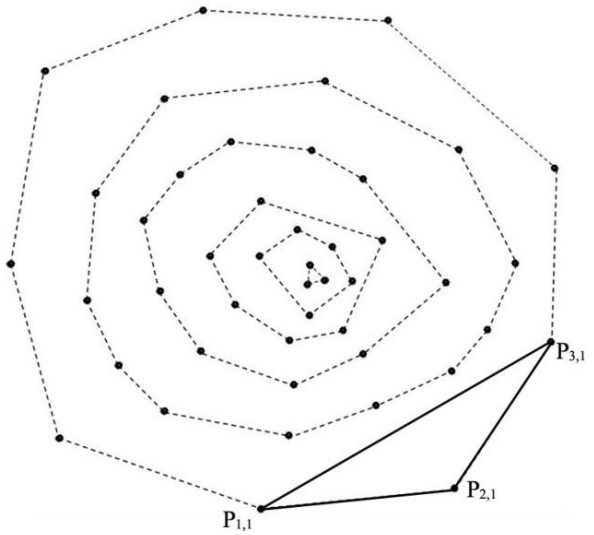
4.1 The suggested algorithm

The suggested algorithm consists of two stages. Applying these two stages to the generated convex layers in the preceding stage, and by generating consecutive pseudo-triangles, pseudo-triangled spiral polygon is eventually obtained.

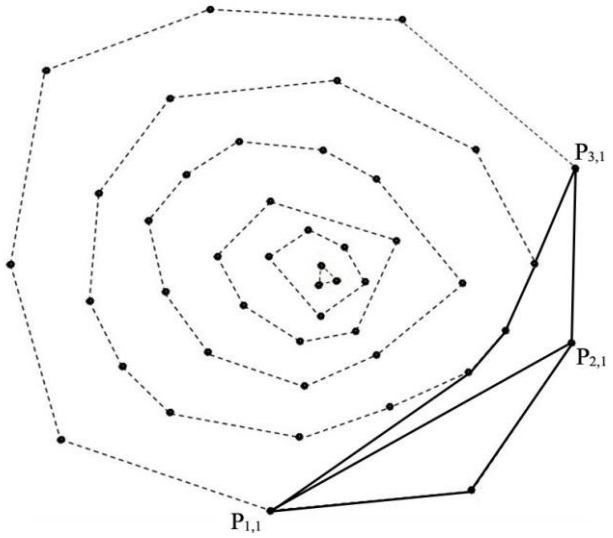
Let M be the number of convex layers. L_j is the jth layer that $j = 1,\ldots,M$ and $P_{i,j}$ is the ith vertex in the jth layer.

4.1.1 The first stage of the algorithm:

- *Step 1*- Take $j \leftarrow 1$ into account and choose a point on L_1 (The most external convex layers) as the starting point and call it $p_{1,j}$.
- *Step 2*- Choose two other points in counter-clockwise direction respectively and call them $p_{2,j}$ and $p_{3,j}$. Take these three points as the pseudo-triangle vertices into account and generate the line segments $p_{1,j}p_{2,j}$ and $p_{2,j}p_{3,j}$.
- *Step 3*- In this step, in order to generate the connecting line segment between two vertices of $p_{1,j}$ and $p_{3,j}$, in case of non existing intersection with layer L_{j+1} , the foregoing line segment is depicted. Otherwise, we should choose and depict points of layer L_{j+1} from vertex $p_{1,j}$ to vertex $p_{3,j}$ which form a concave chain with these two vertices. In this stage of algorithm, a pseudo-triangle has been generated (Fig. 4).



(a)



(b)

Fig. 4 (a) Non existing intersection with layer L_{j+1}. (b) Intersection with layer L_{j+1}.

- *Step 4-* In order to generate the following pseudo-triangle the local position of vertices $p_{1,j}$, $p_{2,j}$ and $p_{3,j}$ should be changed by considering the following conditions:
  a) The local position of point $p_{1,j}$ changes in case of intersection of line segment $p_{1,j}p_{3,j}$ with layer L_{j+1} and is exchanged to the neighboring point $p_{3,j}$ on the concave chain (Fig. 5).

  b) The vertex $p_{2,j}$ is transferred to the local position of the present vertex $p_{3,j}$ and call its neighboring point in counter-clockwise direction on the layer L_j vertex $p_{3,j}$ and generate line segment $p_{2,j}p_{3,j}$ (Fig. 5 and Fig. 6).

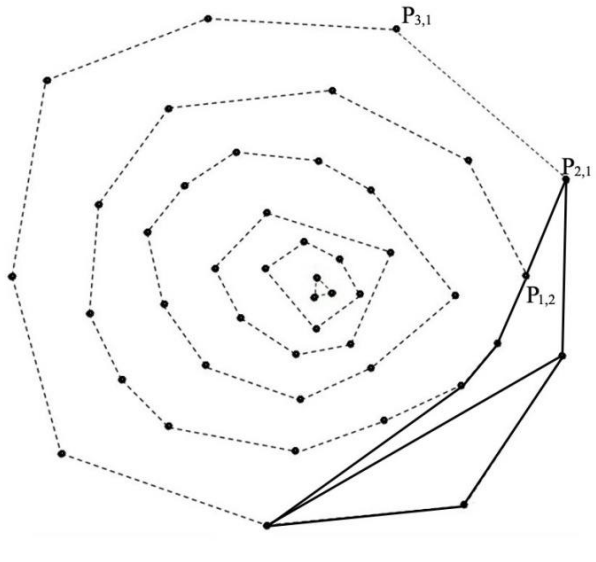


Fig. 5 Change the local position of triangle vertices (Intersection with layer L_{j+1}).

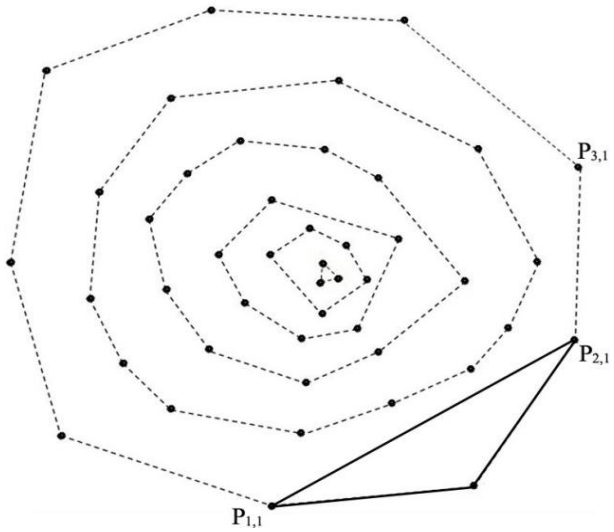


Fig. 6 Change the local position of triangle vertices (Non-existing intersection with layer L_{j+1}).

Repeat steps 3 and 4 as far as the last remaining point on the layer L_j.

4.1.2 The second stage of the algorithm:

Meeting the last point on layer L_j, in this stage among the remaining points on layer L_{j+1} which haven't been used to generate concave chain in the first stage of algorithm, find the farthest visible point from the last point on the layer L_j that is vertex $p_{3,j}$ and call it v, then depict the connecting line segment between the two points. Also, move from vertex $p_{1,j}$

to the visible point on layer L_{j+1} and depict the line segments among the existing points one by one in this path.

After finishing this stage, taking $j \leftarrow j+1$ into account, enter the following layer and consider the point neighboring the farthest visible point (in the second stage of algorithm) in clockwise direct as the starting point of this layer, and from the second step of the first stage we continue the algorithm with the remaining points (except for the farthest visible point) in this layer and repeat the stages till $j < M$ (Fig. 7).

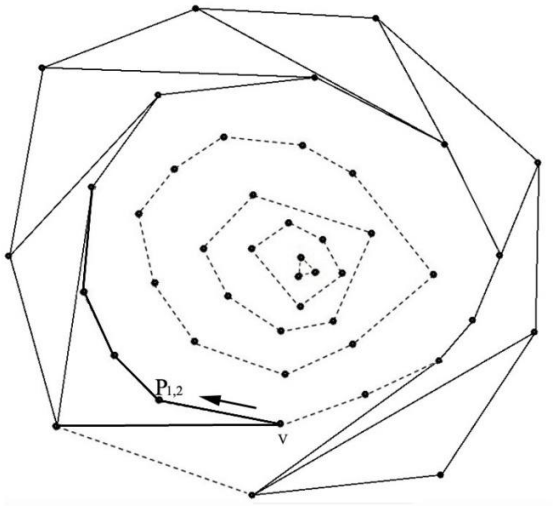


Fig. 7 Entering the next layer (L_{j+1}).

Hence by repeating the stages of the suggested algorithm until $j < M$, a pseudo-triangulated spiral simple polygon is generated (Fig. 8) that in subsection of the following section deal with analyzing the suggested algorithm.

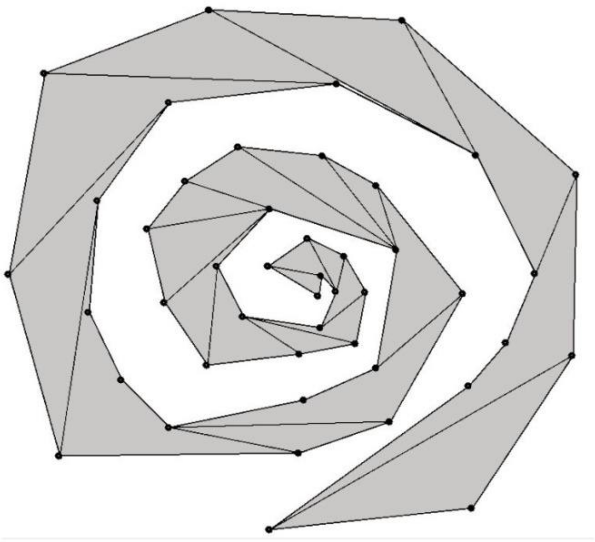


Fig. 8 Pseudo-triangulated spiral simple polygon.

Theorem1. *The presented algorithm produces random simple polygons twice as much as the number of the existing points on the most external convex layer.*

Proof: Since every point of the convex polygon of the most external layer can be the starting point of algorithm and it can be selected clockwise or counter-clockwise in order to be run, so simple random polygons will be produced twice as much as the number of the existing points on the most external convex layer. ■

Pseudo code of the counter-clockwise algorithm:

Algorithm Random Polygon Generation

```
j←1
i←1
P1,j ← pi, j
while j < M do
    while i < nj do
        i ← i + 1
        P2,j ← pi, j
        Draw Line (P1,j , P2,j)
        i ← i + 1
        P3,j ← pi, j
        Draw Line (P2,j , P3,j)
        if (P1,j , P3,j) ∩ lj+1 ≠ Ø then
            find the reflex chain on lj+1 between P1,j , P3,j and draw it
            P1,j ← last element on the reflex chain
        else
            Draw Line (P1,j , P3,j)
            i ← i + 1
            P2,j ← p3, j
            P3,j ← pi, j
        end if
    end while
    v ← find last point which is visible from P3,j among all remain
        points in lj+1
    Draw Line (P3,j , v)
        Draw a chain from P1,j to v
        i ← index of the nearest point to v in clockwise direction
        j ← j + 1
end while
end of algorithm
```

4.2 Analysing the suggested algorithm

The suggested algorithm requires a preprocessing stage called visibility graph, then the generation of convex hulls and eventually the implementation of the suggested algorithm in section 4. The generation of visibility graph can be done in *O(nlogn)*[1]. The generation of convex hull layers can be done in *O(nlogn)*[6]. In section 4, implementing the suggested algorithm to generate the line segment between the two points, the issue of the visibility of the two points should be considered that by performing the preprocessing stage, its time complexity is *O(nlogn)*. With regard to the planarity of the pseudo-triangulation graph to generating all of the pseudo-triangles *O(n)* is required. Thus, the algorithm can be done in *O(nlogn)*.

5 Conclusions

In this paper a new algorithm was presented to generate pseudo-triangulated spiral simple polygons from the set of random points *S* on the plane. The work trend was such that first the convex hull layers were generated for the set of random points *S*. By using this suggested algorithm from the most external convex layer to the most internal layer respectively, by creating consecutive pseudo-triangles, pseudo-triangulated spiral polygon whose time complexity is *O(nlogn)*. The generation of simple polygons out of the set of random points, have such applications as the consideration of heuristic algorithms in issues like Art gallery, and thus algorithms to produce polygon are very efficient in this affair.

6 References

[1] Berg M. D., *Computational Geometry: Algorithms and Applications*, 3rd edition, published by Springer-Verlag, 2008.

[2] Aure T., and Held M., *"Heuristic for generation of random polygons"* 8th Canadian Conference On Computational Geometry (CCCG), Ottawa, Canada, pp.38-44, 1996.

[3] Rote G., Santos F., and Streinu I., *"Pseudo-Triangulation – a Survey"* Discrete Comput. Geom. 2007.

[4] Aichholzer O., Aurenhammer F., Krasser H., and Speckmann B., *"Convexity minimizes pseudo-triangulations"* Computational Geometry 28.2004.3-10.

[5] Dailey D., and Whitfield D., *"Constructing Random Polygons"* SIGITE'08, USA, pp.119-124, 2008.

[6] Chazelle B., *"On the Convex Layers of a Planar Set"* IEEE Tran. Information Theory, Vol. IT-31, No. 4, pp. 509-517, 1985.