Beteckning:________________

HÖGSKOLAN
I GÄVLE

Akademin för teknik och miljö

A comparison of Ear Clipping and a New Polygon Triangulation Algorithm

Ran Liu
June 2010

Bachelor Thesis, 15 hp, C
Computer Science

Computer Science program
Examiner: Anders Hast
Co-examiner: Stefan Seipel
Supervisor: Anders Hast

A comparison of Ear Clipping and a new Polygon Triangulation Algorithm

by

Ran Liu

Akademin för teknik och miljö
Högskolan i Gävle

S-801 76 Gävle, Sweden

Email:
nfk08rlu@student.hig.se

Abstract

Simple polygon triangulation is a classic problem in computational geometry and the techniques are widely used in many fields. Many existing techniques have short running times but are hard to implement. Some are easy to achieve, however, the output quality always bad and is time consuming. This paper proposes a novel diagonal inserting algorithm which is easy to implement and can enhance the final quality. This presented algorithm was implemented and compared with the ear clipping technique which is simple to carry out and long-standing in the triangulation history. These two algorithms were tested on various polygons in 2D and an analysis concerning the quality, speed and the standard deviation of the output triangles` size is done.

Keywords: Triangulation, simple polygon, quality, ear clipping, diagonal inserting.

Contents

1 Introduction

Polygon triangulation is one of the most interesting and useful branch subjects in computational geometry. Triangulation is usually used as pre-processing step in computer graphics, terrain modelling and such similar fields. In the modelling, plenty of polygons are used to build the objects from the real world; however the triangles are always the basic and efficient components for rendering. Therefore, designing a simple and efficient algorithm for triangulation has always been very important.
A new triangulation algorithm is introduced in this report. The idea is aimed at designing a diagonal inserting algorithm which is easy to implement and has a high output quality. Ear clipping is an introductory triangulation algorithm and simpler to realise than other existing diagonal inserting algorithms. Therefore, a comparison was made between the new algorithm and ear clipping [2]. The time complexity will not be the primary property to compare, because the new algorithm will be performed in a straightforward approach and the ear clipping is not famous as its efficiency.

1.1 Aim

The aim of this paper is to compare two diagonal inserting triangulation algorithms based on the quality of the output. One algorithm is designed, the other one is well known as ear clipping. Different kinds of the polygons are tested for comparison.

1.2 Questions at issue

- Which of the two triangulation algorithms produce higher quality of output results on the same polygon?
- What are the properties of the new algorithm?

2 Theoretical background

This chapter is aimed at giving the reader some basic and important knowledge of Computational Geometry. In this paper some ambiguous definitions are clarified and all the *theorems* used here have been proved, for details and proofs see [1].

2.1 Triangulation of polygon

In computational geometry, a *triangulation* is a partition of a geometric domain, Such as a point set, polygon, or polyhedron, into simplices that meet only at shared faces [4]. Triangulating a simple polygon is a basic task in computational geometry and also plays an important role in some other computations. That is why finding a triangulation algorithm that with high quality and low time consumption is so popular and significant in nowadays. And there are many efficient algorithms that have been published. Some algorithms based on the diagonal inserting, like ear clipping which run in O(n^4) time was discovered by Meisters [2]. But it has been improved with the complexity of O(n^2), because of an efficient algorithm for finding an ear in linear time was discovered by Godfried Toussaint et al [5]. And it is a simple algorithm for understanding and implementation. The algorithms based on Delaunay triangulation [8] can get high quality triangulations but are hard to implement. Lewis and Robinson described a recursive algorithms based on Delaunay triangulation takes O(n^2) time[9]. For the purpose of higher triangulation quality, some algorithms are produced by using Steiner points. And Ruppert [10] gave an algorithm that can make the triangles in the output have angles in the interval (140°, 180°). The two algorithms are compared in this paper are all building upon the diagonal inserting. And all the polygons used in the testing are two-dimensional without holes.

2.2 Triangulation quality

The output quality is commonly used in the polygon triangulation algorithm evaluation. For some applications, the minimum interior angle of a triangle of the computed triangulation is as large as possible which defines quality [11]. Therefore, the mean of the all the triangles' minimum interior angles can be used as a criterion to judge the outputs quality and this is the so-called "Average angle". And the low quality triangulation always has many sharp triangles with small minimum interior angle in the triangulated polygon. See figure 1.

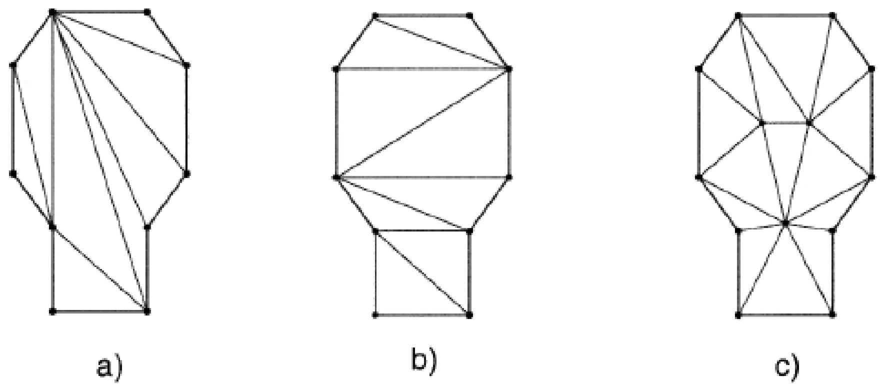


Figure 1: a) Low quality triangulation; b), c) High quality triangulation [11].

2.3 Basic definitions and theorems

2.3.1 Simple Polygon

A *simple polygon* is a closed region of the plane enclosed by a simple cycle of straight line segments [8]. If a polygon is self-intersecting, it is called *non-simple polygon*. A *simple closed polygonal chain* is one in which consecutive segments intersect only at their endpoints and the first point coincides with the last one [10]. The boundary of a *simple polygon* is a *simple closed polygonal chain*. (See figure 2). The line segments in the chain named *edges* and their endpoints are called *vertices*. A *reflex vertex* is one for which the interior angle is greater than 180 degrees [2]. A *convex vertex* is one for which the interior angle is smaller than 180 degrees [2].

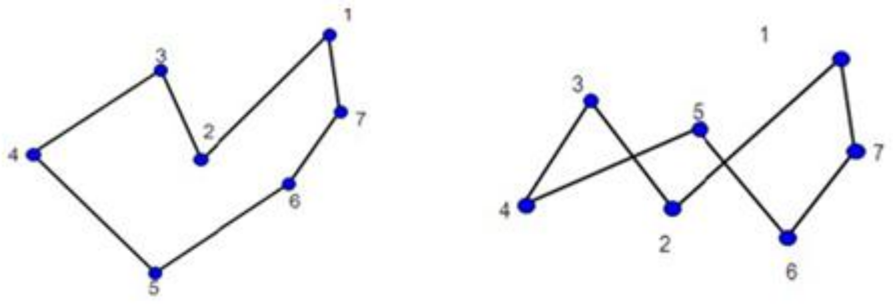


Figure 2: A simple polygon(left) and a non-simple polygon(right) with Vertices{1,2,3,4,5,6,7}.

2.3.2 Visibility

Line segment *XY* does not cross any obstacles so that x and y are *visible* to each other (See Figure 3) [7]. Then endpoint Y is one *visibility* to object X and vice versa. In this paper, the high/low visibility refers to the number of visible vertices of the object. Here the two adjacent points are not considered as visibilities to the object vertex.

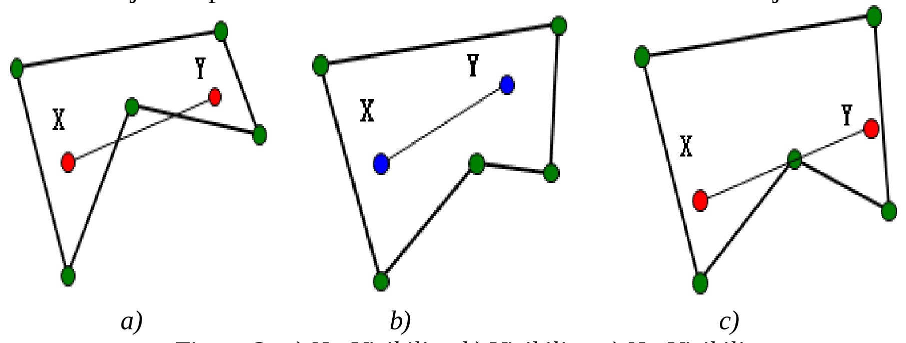


Figure 3: a) No-Visibility, b) Visibility, c) No-Visibility.

2.3.3 Diagonal

A *diagonal* of a polygon P is a line segment inside the polygon whose endpoints are vertices and which otherwise does not touch the boundary of the polygon [6]. And here we define two *diagonals* are *non-crossing* if their intersection point is a subset of their endpoints. [1] *Diagonal inserting algorithm* means adding as many diagonals as possible to partition a polygon.

2.3.4 Ear

In the computational geometry jargon, an *ear* of a polygon is a triangle formed by three consecutive vertices V_{i0}, V_{i1}, V_{i2} for which no other vertices of the polygon are inside the triangle. And the line segment $V_{i0}V_{i2}$ should be a *diagonal* of this polygon, at this time we can call the vertex V_{i1} *an ear tip* [2]. (See Figure 4)

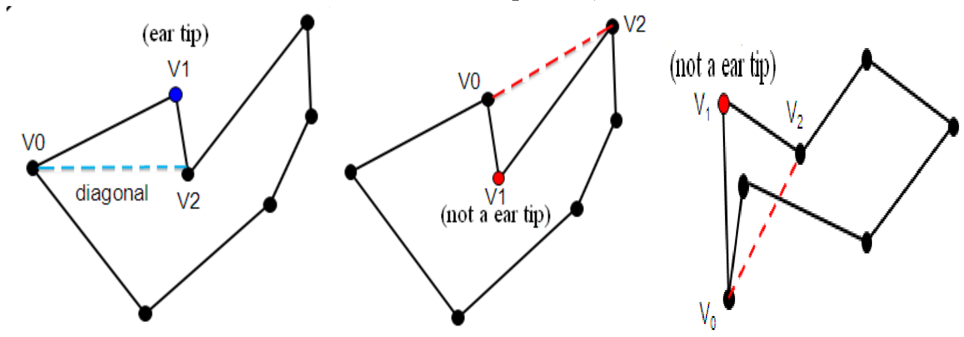


Figure 4: In the left image, $\triangle V_0V_1V_2$ is an ear and $V_1$is an ear tip. In the middle and right images $\triangle V_0V_1V_2$ are not ear because of V_0 V_2 are not diagonals.

2.3.5 Theorems

Theorem 1(Jordan Curve Theorem): *Every simple closed plane curve divides the plane into two components* [1].

Theorem 2(Triangulation): *Every polygon P of n vertices may be partitioned into triangles by the addition of (zero or more) diagonals* [1].

Theorem 3(Number of Diagonals): *Every triangulation of a polygon P of n vertices uses (n-3) diagonals and consists of (n-2) triangles* [1].

Theorem 4(Two Ears Theorem): *Every polygon P of n$\geqslant$4 vertices has at least two non-overlapping ears [5].*

3 Generating random simple polygons

To make the comparison of two algorithms exactly, plenty of controllable polygons are needed for the testing. And the method presented below can produce the polygons. It is known that a simple polygon is a polygonal chain that has no-intersection without the start and end vertices. Therefore, the task is to create a polygonal chain randomly. This work is done in several steps. Firstly lots of points are generated with random coordinates. The shape of the polygons can be controlled by the random coordinates and that is very important for the further testing. After that, compute a bounding circle that can enclose all that points and get the center. Split the points into two parts by the horizontal axis of the centre point .See Figure 5.

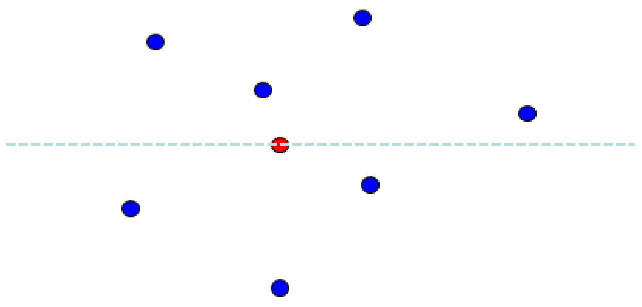


Figure 5: The red point is the centre point was found and the blue points are created with the random coordinates. The blue points are split into two parts by the dotted line which is the horizontal axis of centre point.

After the partitioning of points, calculate the angles between the positive X axis and those which were indicated by the random points and the centre point. Then sort all of them angularly about the centre point. See figure 6. Compare to Graham`s convex hull algorithm [1], all the sorted points will be linked with index.

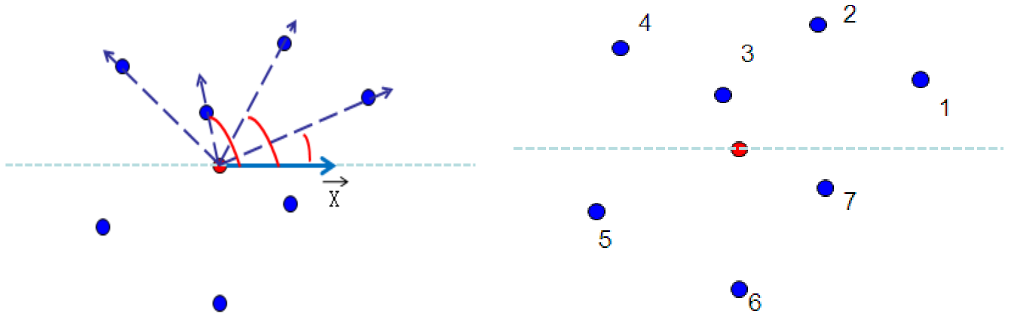


Figure 6: In the left image the angles are calculated, in the right image the vertices are sorted by the angles.

The polygonal chain is generated by linking the sorted vertices. Connect the head and tail of the chain, then one *simple polygon* is coming out (See Figure7). And all the simple polygons that used to test the two algorithms were generating in this way. As the representation in Figure 7, all the simple polygons are counter clockwise ordered and all vertices` indices are assigned at the sorting step. By *Theorem 1*, we know a polygon divides the plane into exterior and interior. And the interior refers to the part inside the polygonal chain.

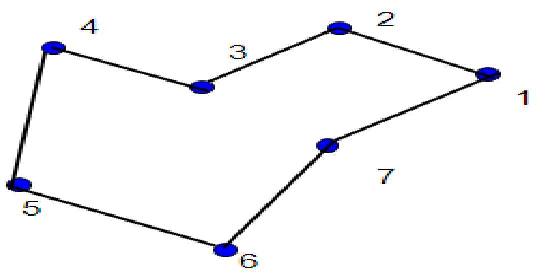


Figure 7: A simple polygon generated randomly.

4 Description of the algorithms

This section explains these two comparative algorithms in details. The first algorithm is known as "Ear Clipping" or "Ear Removal". The other one is designed from the theoretical analysis and then implemented in practical.

4.1 Ear Clipping

"Ear Clipping" is an example algorithm that is explained by Joseph O`Rourke [1] in his book. This algorithm has been realized in many different ways. By *Theorem 4*, a straightforward and recursive implementation can be done. However, this approach will lead a $O(n^3)$ time complexity and the "ear tip finding" step takes $O(n^2)$ due to the vertices' status updating. Joseph O`Rourke sketches a method for improving this technique to $O(n^2)$. And the ear finding step has been reduced to O(n) technically. The final order of ear clipping is $O(n^2)$. The following statements will present how this algorithm works.

4.1.1 Finding ear tip and triangulation

According to the definition, we can check a vertex whether an *ear tip* in this way. For any vertex V_i and corresponding triangle<V_{i-1}, V_i, V_{i+1}>. If line segment (V_{i-1}, V_{i+1}) is a diagonal of the polygon, then triangle<V_{i-1}, V_i, V_{i+1}> is an *ear* and V_i is an *ear tip*. Otherwise it is not. In other words, detect the segment of two adjacent vertices whether a diagonal can judge a vertex is an ear tip or not. Use this method to iterate over the vertices and record the status .This initialization process will take O(n) time.

After that we start to cut off the ears. Different to the straightforward way, the ear status will renew for the two adjacent vertices instead of all after one *ear* is cut off. Because the removal of an ear just affects its adjacent vertices V_{i-1}, V_{i+1}. And this change saves O(n) time. Put the independent updating and cutting steps together makes this algorithm with $O(n^2)$ time complexity . Figure 8 shows the pseudo code of this approach. [1]

```
Initialize the ear trip status of each vertex
While n>3 do
        Locate an ear tip $V_i$.
        Delete $V_i$.
        Update the ear trip status of $V_{i-1}$ and $V_{i+1}$
```

Figure 8: Ear Clipping Algorithm [1]

The following example will present a simple polygon triangulation by using ear removal. The polygon was generated as the way represented above. The initial list of ear tips is E= {1, 3, 5, 6, 7, 9, 10} .Follow the pseudo code, the ear at vertex 1 is removed and T_1= {10, 1, 2} is the first triangle in the triangulation. The list of *non-crossing diagonal* is D= {(2, 10)}. Then update status of V_{10} and V_2. Vertex 10 is still an ear tip and vertex 2 still is not. And the new ear tips list is E= {3, 5, 6, 7, 9, 10}. See Figure 9.

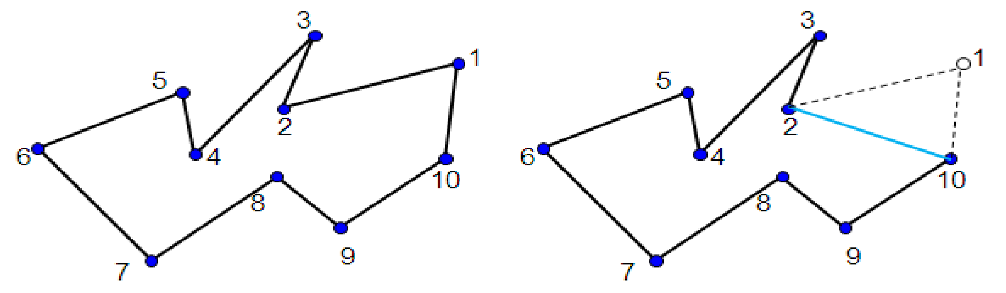
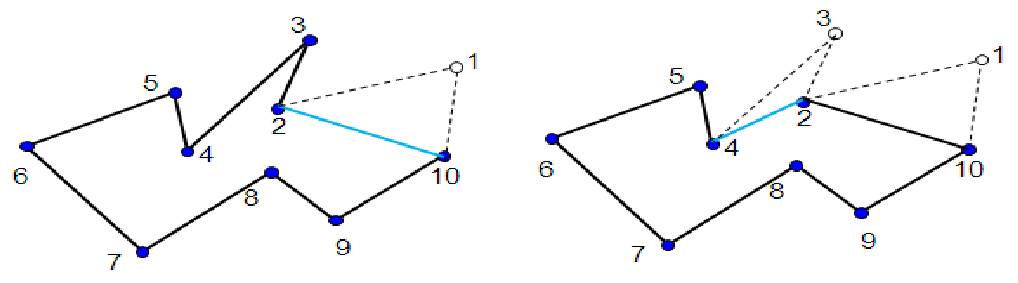


Figure 9: The right image shows the ear tip 1 is removed from the left simple polygon. And the diagonal <2, 10> is drawn.

The ear tip V_3 is removed. The triangles list T= {(10, 1, 2), (4, 3, 2)} and Diagonal List D= {(2, 10), (2, 4)}. Then update the status of adjacent vertices V_2 and V_4. V_2 was not an ear tip but has become one now and V_4 still is not. So the new ear tips list becomes E= {5, 6, 7, 9, 10, 2}. The new ear is added at the end of the list that is because the vertex is checking one by one in the counter clockwise order. See Figure10.



a) b)

Figure 10: a) and b) The process of ear tip 3 removing. And the diagonal <2, 4> is drawn.

Then, the recursive operation is done until there are just 3 ear tips left in the list E. See Figure 11.

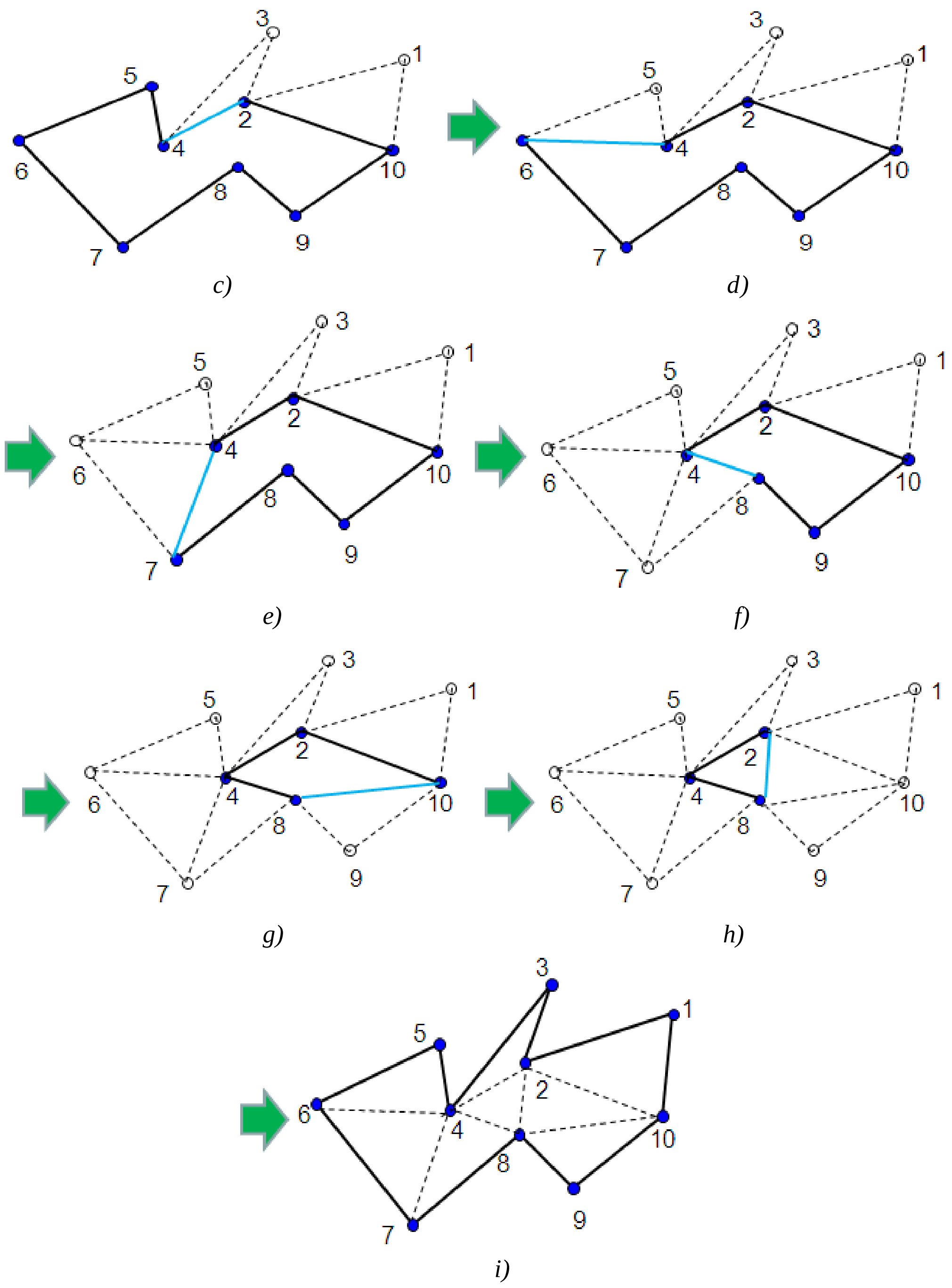


Figure 11: c) - i).The process of Ear clipping algorithm. i) triangulated polygon. T= {2, 4,8}, D={(2,10),(2,4),(4,6),(4,7),(4,8),(8,10),(2,8)} and the triangles list T={(2,1,10),(4,3,2),(6,5,4),(6,4,7),(7,4,8),(8,9,10),(2,10,8),(4,2,8)}

The Full triangulation of the original simple polygon is show in image 8. After the triangulation all the diagonals and triangles are stored in arrays D and T for the further computation.

4.2 New Algorithm

The new algorithm performs the diagonal inserting operation based on the vertices` visibilities. And in this part the *high/low visibility* refers to the number of visible vertices of the object. The primary intention is to divide a big simple polygon into two smaller individual sub-polygons from the middle and do recursion with the sub-polygons. As experience, the reflex points always have a high visibility and distribute in the middle of a polygon. These two features can be used to find the mid part of polygon. In the original thought, this algorithm contains four steps. First, compute the visibilities for all the vertices. Second, find the vertex with the highest visibility. Third, connect the highest visibility vertex with its visible vertex which having "second" highest visibility. Fourth, Split the polygon into two pieces by the line segment found in step three and do recursion with the sub-polygons until three vertices left in the polygonal chain. The visibilities computing and updating are described in the following parts.

4.2.1 Compute The Visibilities.

All the operations in this algorithm are based on the visibilities. The vertices\` visibilities are computed in a straightforward way as checking all the other points to detect whether they are visible. And if a target vertex fulfils two requirements, it is counted as one visibility for the object vertex. First, the target vertex should be in object\`s field of view. Second; it should be clearly seen without any blocking. In figure 12, the space between vectors V_0V_1 and V_0V_9 is the visible area of object V_0 (the field of view) and α called the *view angle* of V_0. The following example illustrates how the visibility of vertex 0 is computed.

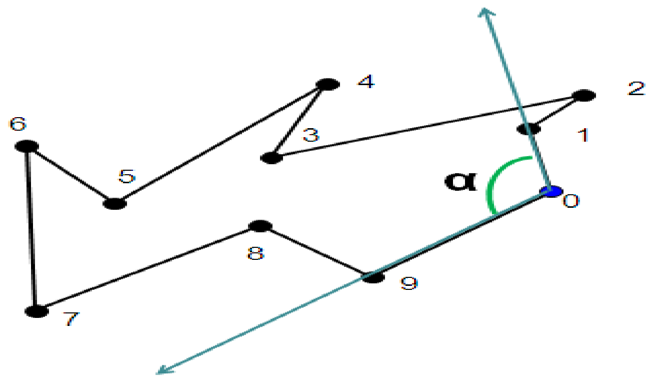


Figure 12:The polygon with Vertices{0,1,2,3,4,5,6,7,8,9}. Vertex 0 is the object and the area in angle α is the visible region.

As figure 12 says the vertex 2 is out of V_0\`s field of view, so it is omitted. Continue checking V_3 with the two requirements. V_3 is in the visible area of V_0 and V_0V_3 has no intersection with other vertices or edges. Therefore, V_3 is one visibility of V0. Update the field of view and resume the checking work from V_4. See figure 13.

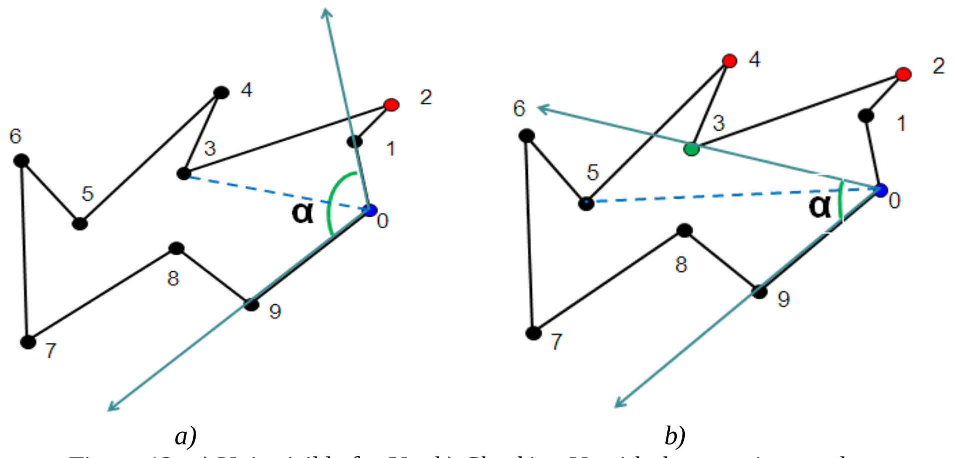


Figure 13: a) V_3 is visible for V_0. b) Checking V_5 with the new view angle

The figure 13 (right) shows V_4 is out of the field of view. Skip it and continue checking V_5. V_5 fulfils the requirements, so it is visible for V_0. The visible region is updated and the next vertex will be checked. See Figure 14.

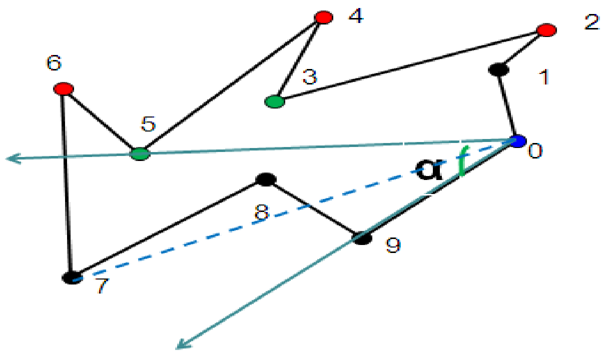


Figure 14: Vertex 6 out of the field of view and V_0V_7 intersects with V_8V_9.

Vertex 7 in the visible area but blocked by the edge V_8V_9, so it is not a visible target for Object V_0.Then check Vertex 8, see figure 15.

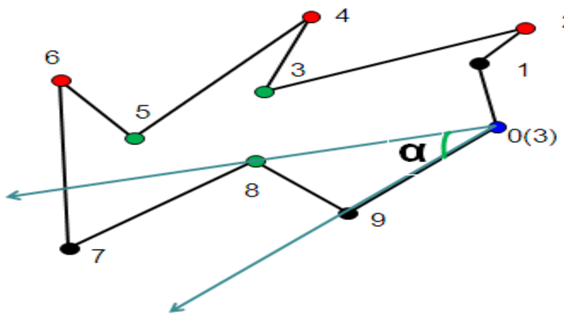
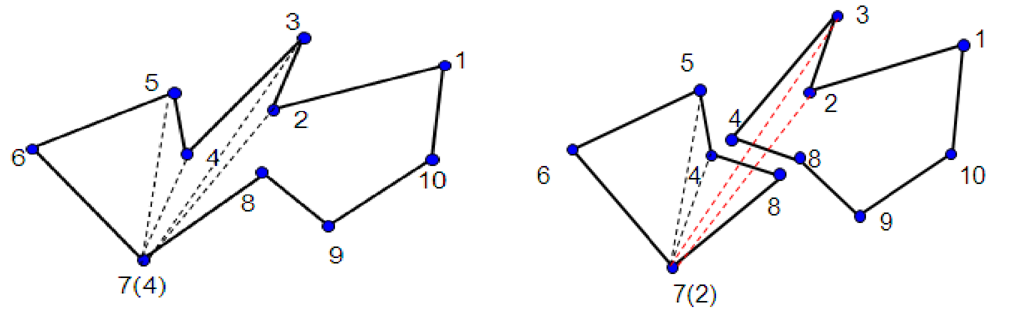


Figure 15: The number in the parenthesis refers to the visibility of V_0. The light green points are visible for V_0.

The computation will end if not any target vertices in V_0`s field of view. The visible vertices list is <$V_3$, $V_5$, $V_8$> so the visibility of $V_0$ is 3. All the vertices` visibilities will be computed like the process described above. Although some techniques and “tricks” are used to speed up the visibilities computation, the initialization process still take over $O(n^2)$ time. Testing each vertices for one object vertex costs over O(n) time and repeating this process for all the object vertices takes totally more than $O(n^2)$.

4.2.2 Update The Visibility and triangulation.

After each division of a polygon, all the visibilities need to be recomputed. For the purpose of speeding up, the new visibilities will not be calculated like the initialization way and the *single circular list* data structure is used. The *single circular list* is a singly linked-list with the tail node points to the head. Store the visible vertices in a circular list for each object vertex when the visibilities are computed for the first time. Then the visibilities can be updated by checking the vertices in the list. Delete the vertices that are not visible from the list after a polygon division and the number of remains is the new visibility for the object vertex. The visibility updating of vertex 7 is shown in Figure 16.



a) *b)*

Figure 16: a) The polygon before division. b) The polygon after division. The blue dotted lines mean visible, the red ones mean invisible.

Look at image (a), the visibility of V_7 is 4 after the initialization and the visible vertices are <V_2, V3, V4, V5>. Those vertices are stored in list L_7= {V2, V3, V4, V5}. The whole polygon first decomposed by diagonal V_4V_8 as showing in image (b). In the sub-polygon {8, 4, 5, 6, 7}, the new visibility of V_7 is computed by checking the elements in list L_7. As the image (b) shown V_2 and V_3 are blocked by the new edge V_4V_8 so they are not visible for V_7 anymore. Delete V_2 and V_3 from L_7 and L_7 = {V4, V5}. Therefore, the new visibility of V_7 is 2. Use this method to update the visibilities can save lots of running time and more efficiently. But this step is still time consuming. Because updating one vertex visibility takes O(1) in the best case and O(n) in the worst case, repeating this process for the polygons that have three more vertices takes at most O(n) time. In totally, this approach costs at least O(n). This rough time complexity analysis about the visibilities computing and updating clearly shows that this new algorithm is slow and inefficient. Figure 17 shows the pseudo code of this new algorithm.

```
Initialize the visibility of each vertex for polygon
function(polygon)
    a ← vertex with highest visibility of Polygon.
    b ← vertex with "second" highest visibility of Polygon.
    divide polygon into two sub-polygons by ab.
    polygonA ← a sub-polygon of polygon.
    polygonB ← another sub-polygon of polygon.
    numberA ← the vertices number of polygon A.
    numberB ← the vertices number of polygon B.
    if(numberA>3)
        update the visibilities of polygonA.
        function(polygon A).
    if(numberB>3)
        update the visibilities of polygonB.
        function(polygon B).
```

Figure 17: Pseudo code of the new algorithm

The following example presents how the new algorithm works. The polygon is the one used in "Ear Clipping".
Firstly, initialize the visibilities of the vertices using the method described above. After initialization, look for the vertex with the highest visibility and set it as the head. When the visibilities are the same, the head can be anyone of them. See figure 18.

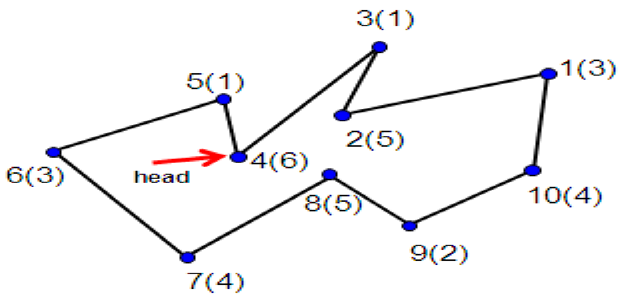


Figure 18: The visibilities are in the parenthesis, V_4 is the head.

Next step is finding the vertex having the “second” highest visibility. The visibility of V_2 and V_8 are both 5. Then the absolute difference values of two sub-chains' length are compared. As the figure 19 showing, the absolute difference values of V2 is 6 and V8 is 2. For the purpose of mid cutting, the vertex has small difference value is the “second” head. The process shows in Figure 19.

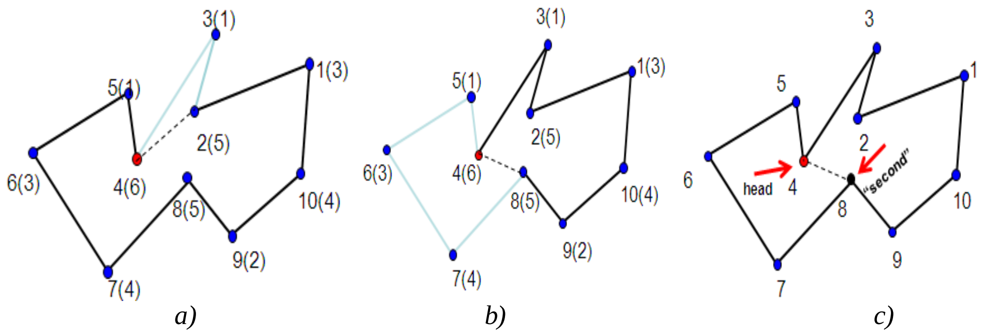


Figure 19: a) and b) The absolute difference values of the black chains and the light green chains in image are compared. c) First and second heads.

Follow the pseudo code; split the polygon in two by V_4V_8. See figure 20.

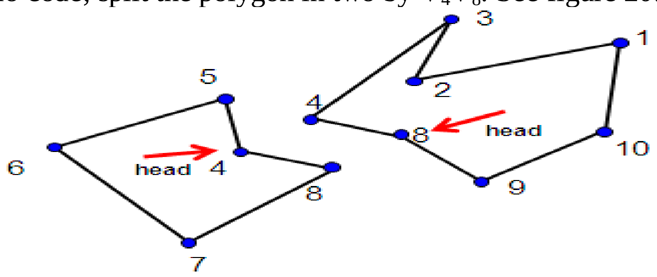
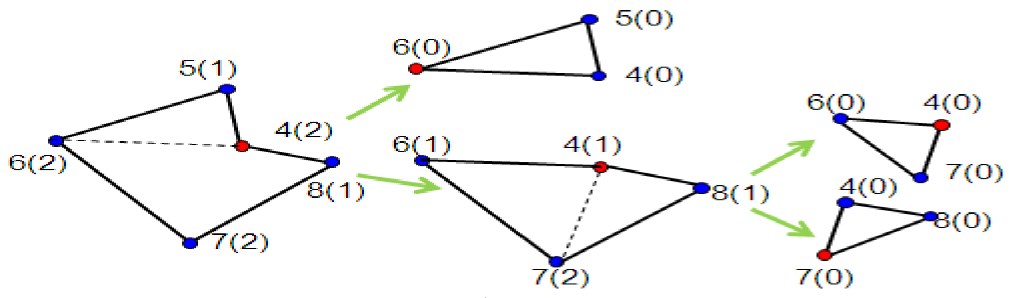
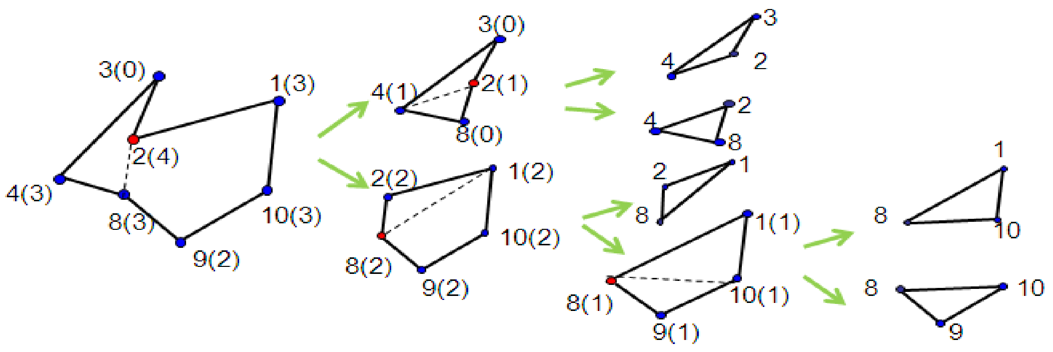


Figure 20 : Polygon<4,5,6,7,8> and <8,9,10,1,2,3,4> are the new sub-polygons.

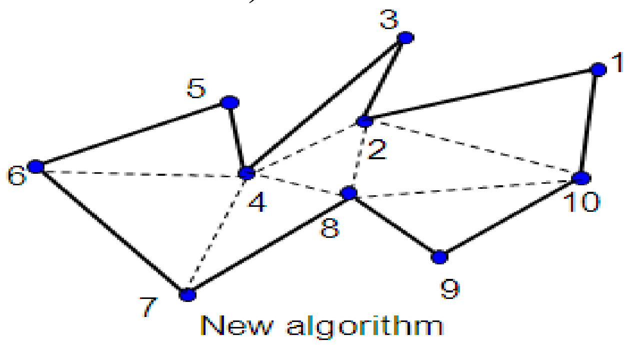
Update the visibilities of sub-polygons and do recursive operations on those which vertices number more than three. The left steps show in Figure 21.



a)



b)



c)

Figure 21: a) and b) The process of triangulation. c) The final result after decomposing.

The two algorithms got the same results with the particular polygon. But the new algorithm takes more running time than ear clipping.

5 Implementation

This section describes the implementation of the two algorithms.

5.1 Previous Work.

The two algorithms to be compared were written in C. And the array, circular list data structures and the sequence container "vector" are used in the program.
All the random coordinates are stored in two-dimensional arrays and generated in an alterable interval, in this aspect, they are controllable. The coordinates are presented with integers rather than with floats. This will make us avoid the float value round-off error and easy to debug. After the points` generation, a random simple polygon is created by sorting and linking the points. In order to insert and delete the vertices more easily, the doubly linked circular list is used to store the polygonal chain. Since the vertices of a polygon are counter clockwise ordered, the circular list permits the operations on the polygon more intuitive. That is the reason for using this structure. Another circular list structure is used in the new algorithm to store the visible vertices of each object point. The basic cell of the double list represents a vertex, with fields for its coordinates, pointers to the adjacent vertices, a unique index and the visibility etc. And in the single list, one node just contains the index of a visible vertex and the pointer to the next node. According to *Theorem 3*, the number of the diagonals and triangles are constant over all the triangulations. Therefore, the diagonals and triangles are stored in the separate arrays with length (n-3) and (n-2); n is the vertices number of a polygon.

5.2 Triangulating with ear clipping.

The first pass is to initialize the status of all vertices. The vertices in the doubly-linked circular list will be checked one by one after the initialization. If a vertex is an ear tip, clip the ear and update the status of the adjacent vertices. Then delete this vertex from the doubly linked list, and store the diagonal and ear triangle in two arrays. The arrays are used to draw the outputs and compute the area. Continue clipping until three vertices left in the doubly circular list. When the triangulation finished, the area of each triangle, the polygon area and the standard deviation of the triangles` area will be computed with the array which stores the triangles. All the information will show on the screen for comparing.

5.3 Triangulating with new algorithm.

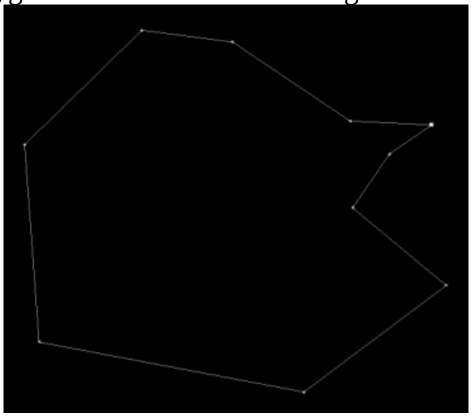
The singly linked list used in this algorithm is aimed to increase the algorithm`s efficiency. Each vertex in the double circular list has an own single circular list that is used to compute the new visibilities.
The first step of this algorithm is to initialize the visibilities and build the single lists for each vertex. Looking for the vertices with the highest and "second" highest visibilities in the double circular list then split the list in two with the found vertices. If the new doubly list contains more than three vertices, update its visibilities. Otherwise, the recursion stops on the given list and a triangle is produced. All the diagonals and triangles found in the triangulation also are stored in arrays with length (n-3) and (n-2); n is the vertices number of a polygon.

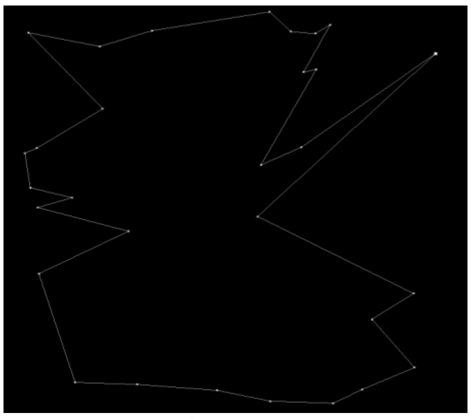
6 Results

This section includes the resulting images and some statistics graphs.

6.1 Randomly generated polygons.

The images represent two types of polygons that were tested, the first four images are round polygons and the last two are long.


a)


b)

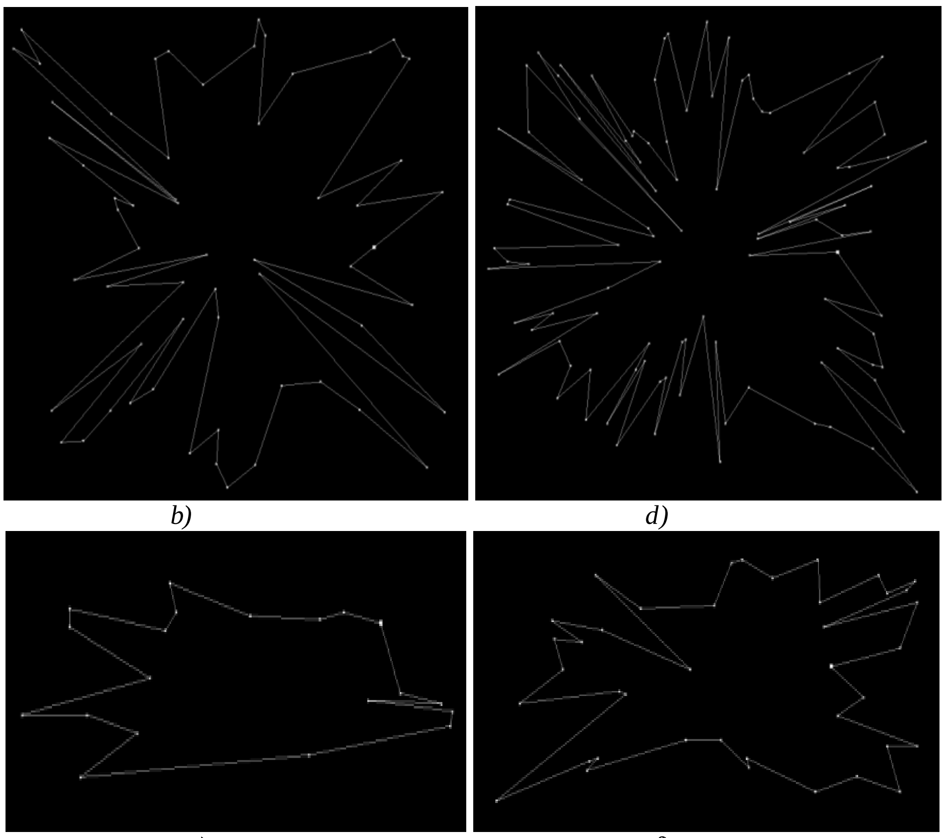


Figure 22: a) 10 vertices, b) 30 vertices, c) 60 vertices, d) 100 vertices, e) 20 vertices, f) 40 vertices.

6.2 Resulting images.

In this section, the tables show the information of polygons and the measured values. The “Mean size” used in the tables refers to the mean of the triangles’ area, “Standard deviation” refers to the standard deviation of the triangles’ area and the “Average angle” refers to the average of the triangles’ minimum interior angles. The images of results are displayed on pairs where the left image is ear clipping triangulation and the right image is the new algorithm triangulation. The normal distribution graphs show the distribution of the triangles’ area. Two types of polygons with different vertices were tested and some more complete results are shown in appendix 1.

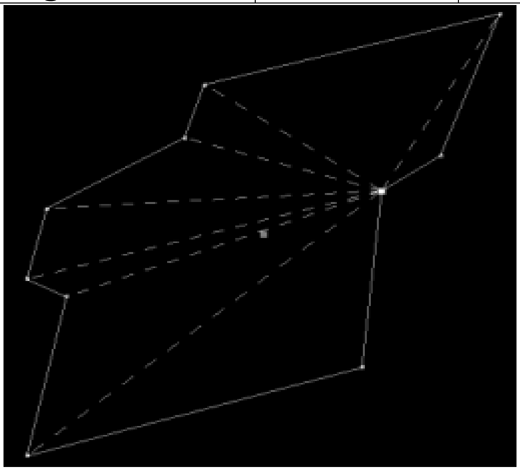
6.2.1 Testing on the round polygons.

For this type polygons, the “Standard deviations” and “Average angle” were measured when polygons contain from 10 to 55 vertices (with step 5), and repeated 20 times for each number of vertices. The comparisons of the standard deviation are shown in figure 26.

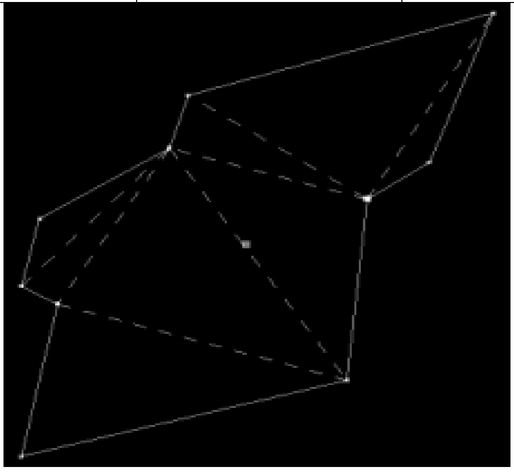
The round polygon with 10 vertices was tested. The vertices were generated in the interval [-20, 20].

Vertices Number: 10 **point:** $\{(X,Y) \mid -20 \le X \le 20, -20 \le Y \le 20,\}$				
Algorithms	Polygon area	Mean size	Standard deviation	Average angle
Ear Clipping	315	39.38	25.81	19.25

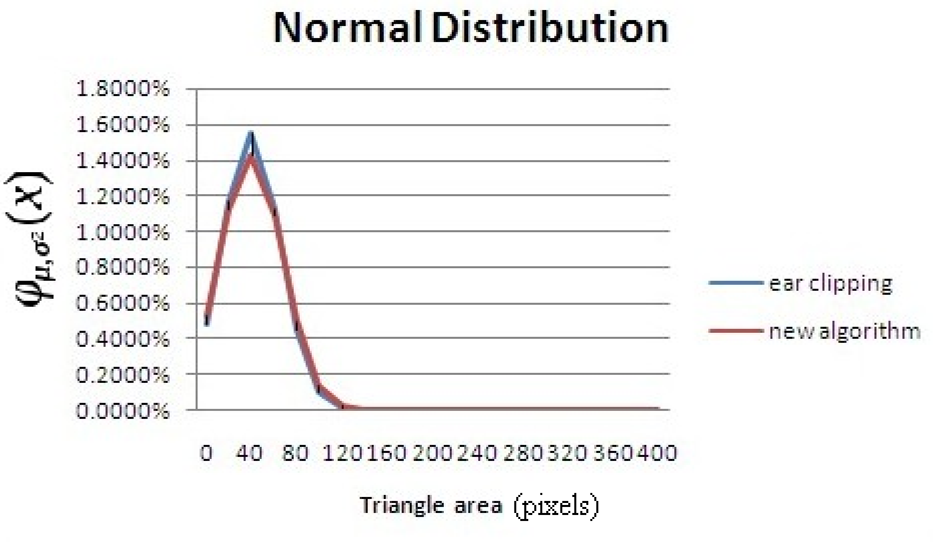
New Algorithm	315	39.38	28.05	27.38



a)



b)



c)

Figure 23: a) The ear clipping triangulated polygon, b) The new algorithm triangulated polygon. c) The normal distribution of triangles` area.

.

The round polygon with 30 vertices was tested. The vertices were generated in the interval [-50, 50].

Vertices Number: 30	**point:** $\{(X,Y)\vert -50 \le X \le 50, -50 \le Y \le 50,\}$			
Algorithms	Polygon area	Mean size	Standard deviation	Average angle
Ear Clipping	4128.5	147.446	163.63	10.29
New Algorithm	4128.5	147.446	148.91	15.04

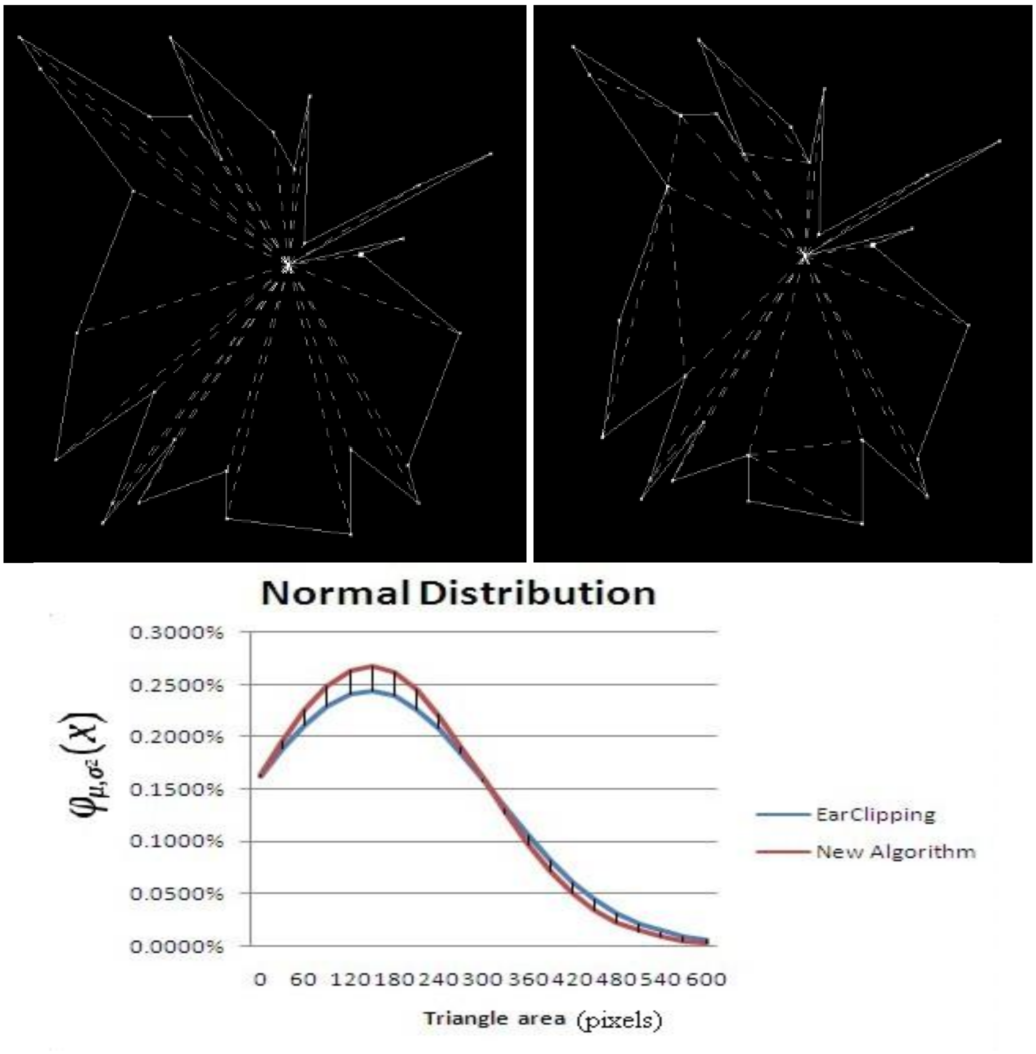
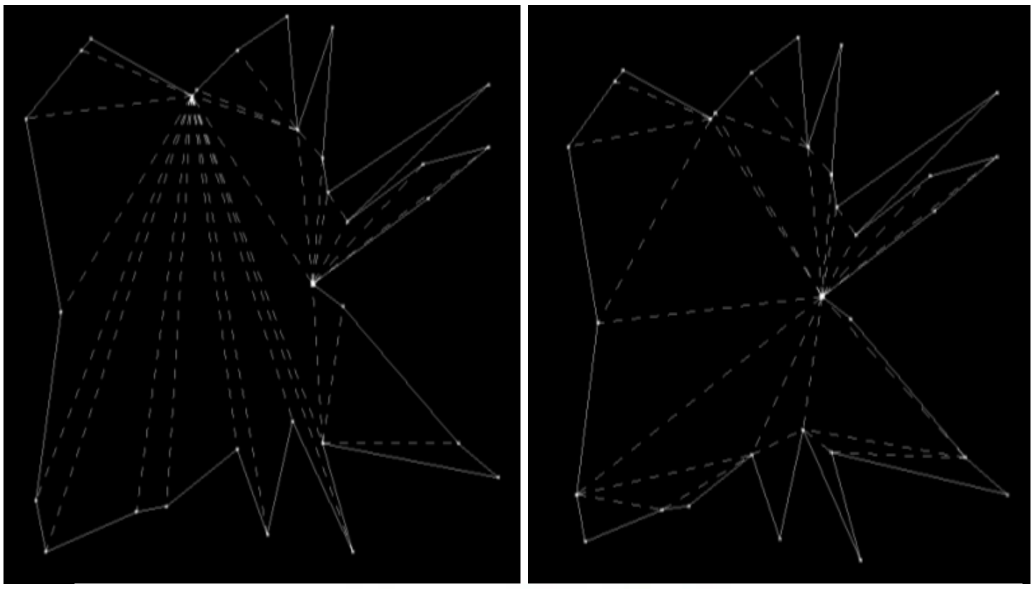


Figure 24: a) The ear clipping triangulated polygon, b) The new algorithm triangulated polygon. c) The normal distribution of triangles` area.

The round polygon with 30 vertices was tested. The vertices were generated in the interval [-50, 50].

Vertices Number: 30	**point:** $\{(X,Y) \mid -50 \le X \le 50, -50 \le Y \le 50,\}$			
Algorithms	Polygon area	Mean size	Standard deviation	Average angle
Ear Clipping	5015	179.11	165.64	12.53
New Algorithm	5015	179.11	233.05	19.11



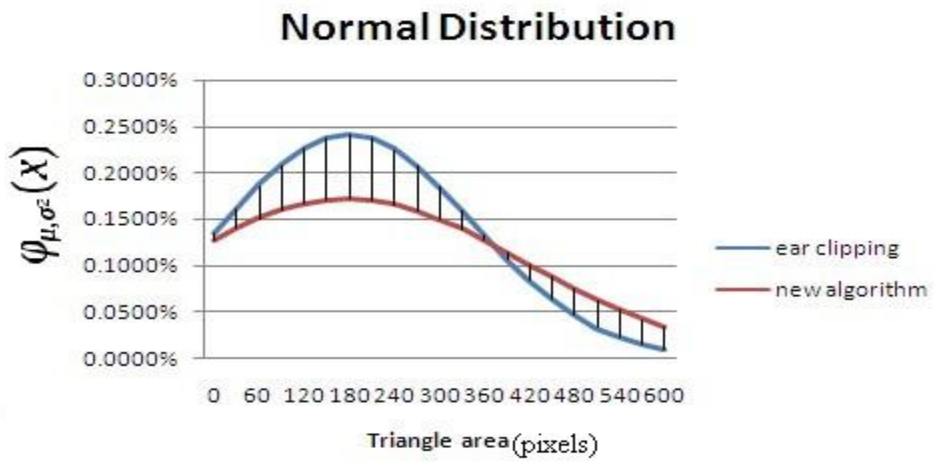


Figure 25: a) The ear clipping triangulated polygon, b) The new algorithm triangulated polygon. c) The normal distribution of triangles` area.

Figure 26 shows an outline in percent of the algorithms got the lower standard deviation after 200 polygons were tested. It is obvious that ear clipping works better than the new algorithm on this kind of polygons. Ear clipping has Over 65% probability to get the lower standard deviation.

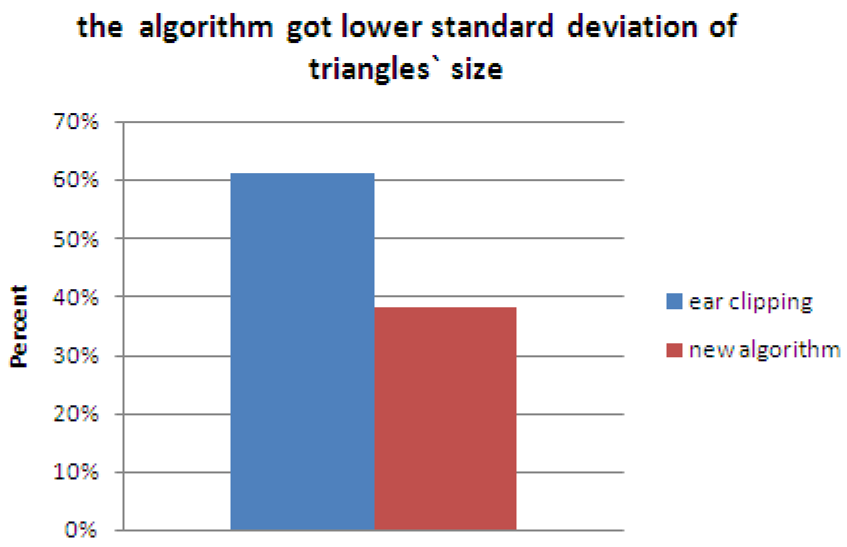


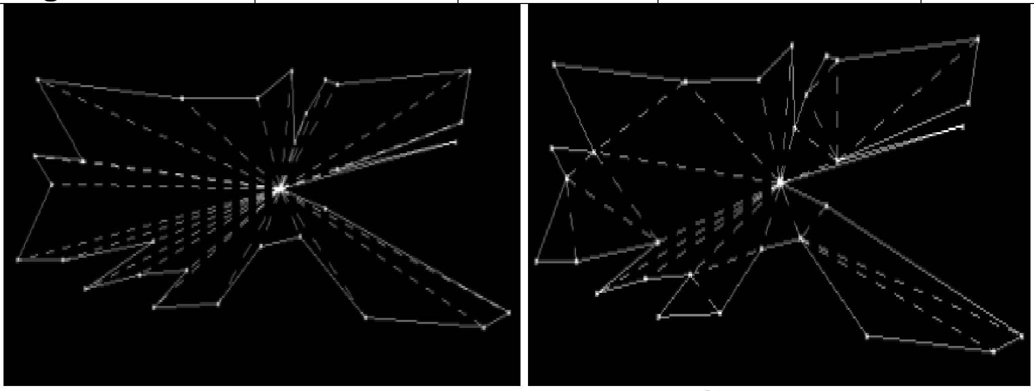
Figure 26: Diagram showing in percent of the algorithms got the lower standard deviation

6.2.2 Testing on the long polygons.

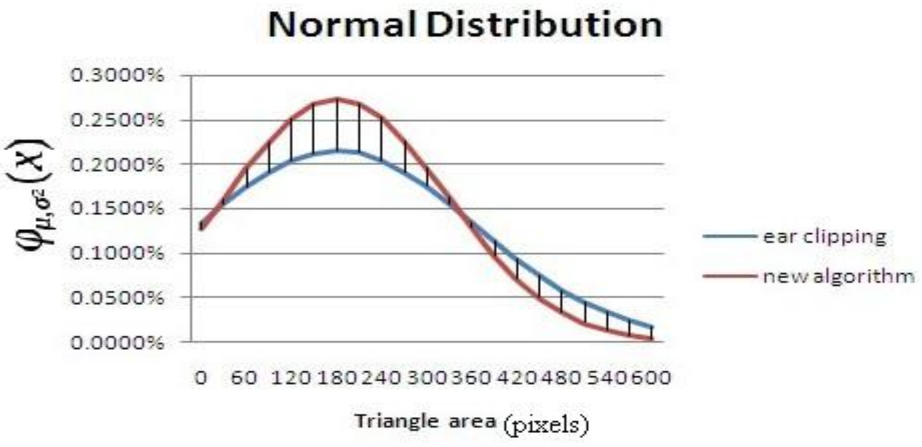
For the long polygons, the "Standard deviations" and "Average angle" were measured when polygons contain from 10 to 55 vertices (with step 5), and repeated 20 times for each number of vertices. The comparisons of the standard deviation are shown in figure 29.

The long polygon with 30 vertices was tested. The vertices were generated in the interval $\{(X,Y)|-100 \le X \le 100, -30 \le Y \le 30,\}$

Vertices Number: 30	**point:** $\{(X,Y)\|-100 \le X \le 100, -30 \le Y \le 30,\}$			
Algorithms	Polygon area	Mean area	Standard deviation	Average angle
Ear Clipping	5066	180.93	185.50	9.23
New Algorithm	5066	180.93	146.03	16..45



a) b)



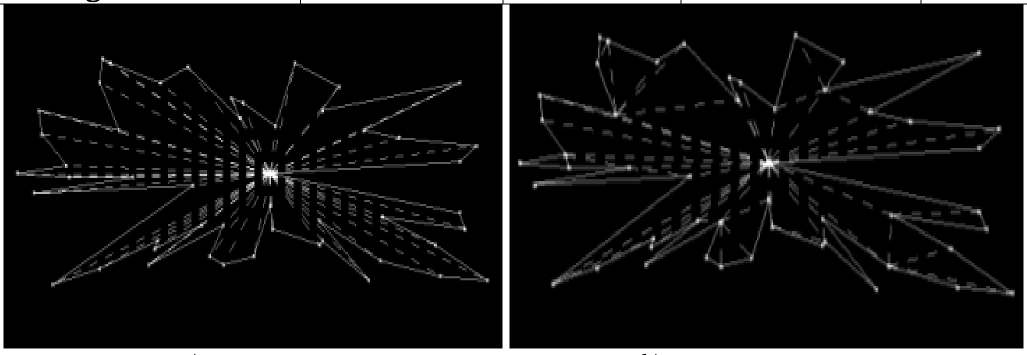
c)

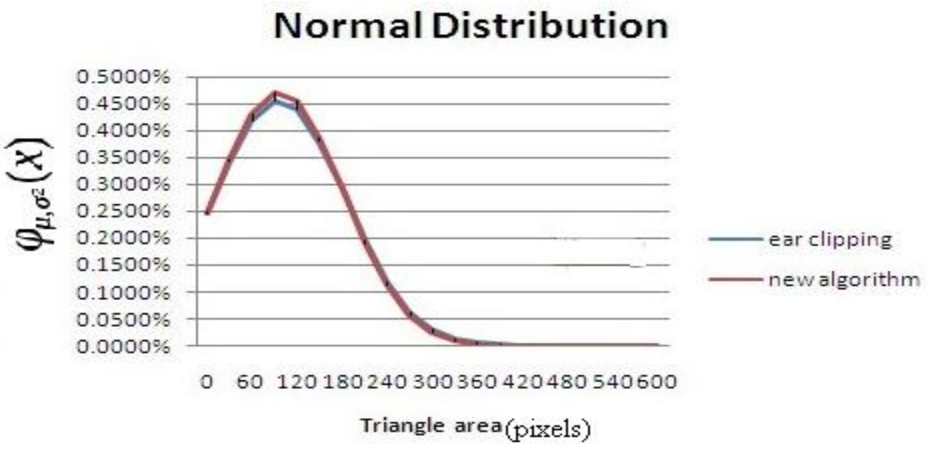
Figure 27: a) The ear clipping triangulated polygon, b) The new algorithm triangulated polygon. c) The normal distribution of triangles` area.

The long polygon with 50 vertices was tested. The vertices were generated in the interval $\{(X,Y)|-20 \le X \le 20, -20 \le Y \le 20,\}$

Vertices Number: 50	**point:** $\{(X,Y)\|-20 \le X \le 20, -20 \le Y \le 20,\}$			
Algorithms	Polygon	Mean	Standard	Average

	area	area	deviation	angle
Ear Clipping	4634	96,54	87.24	8.56
New Algorithm	4634	96,54	84.23	14.62





c)

Figure 28: a) The ear clipping triangulated polygon, b) The new algorithm triangulated polygon. c) The normal distribution of triangles` area.

Figure 29 indicates that the new algorithm has Over 55% probability to get the lower standard deviation than ear clipping on this type of polygons.

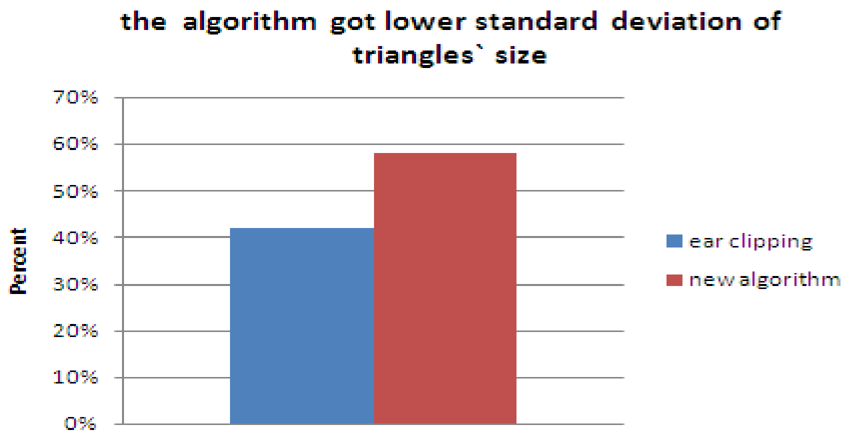


Figure 29: Diagram showing in percent of the algorithms got the lower standard deviation

7 Discussion

7.1 The polygon images.

The figures of the random polygons indicate that the polygon generation algorithm used in this report is not good at producing the polygon with large vertices number. The shapes are almost the same when the number of vertices is over 60. That gives the reason why the testing on polygons with lots of vertices is not done in this report. All the polygons used for testing are within 60 vertices. The shape of the polygons can be controlled by altering the interval of the coordinates. However, the polygons were chosen before the testing every time.

7.2 The resulting images.

In the original thought, the standard deviations of triangles` area were considered as the criterion for the quality comparison. On this condition, the two diagrams clearly show that Ear clipping works better on the round polygons and the new algorithm is fit for the long polygons. That is because the new algorithm`s middle cutting operation could cause a big area difference on the round polygons. For instance, look at the round polygon in figure 30.

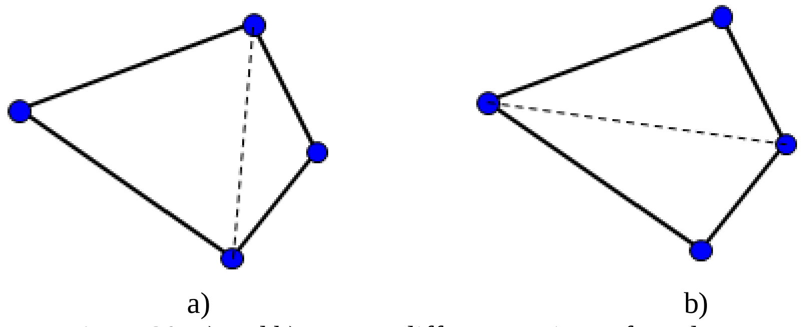


Figure 30: a) and b) are two different cuttings of a polygon.

In the implementation, in order to reduce the sharp triangles, the new algorithm cuts a polygon like image a) shows. And ear clipping can cut the polygon in both ways. It is obvious that two triangles have a big area difference in image a), small difference in b). That is why the new algorithm has more chances to get a high standard deviation of the triangles` area. But this will not happen on the long polygons. Look at the long polygon cutting in figure 31.

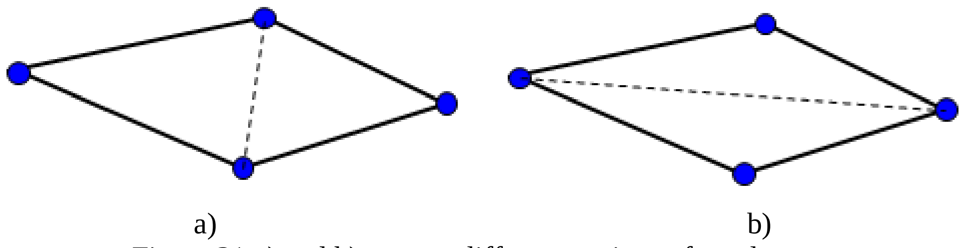


Figure 31:a) and b) are two different cuttings of a polygon.

The long polygon will reduce the difference between two triangles. That is the reason why the new algorithm works better on long polygon.

However, in fact overall the results and the statistics in the tables clearly indicated that the new algorithm decreased the number of sharp triangles in both two types of polygons. In this aspect, the quality rose obviously and the original purpose was reached. And in the figure 23 and 25, although the standard deviation of new algorithm is high, the output qualities are still better on visual. Of course there is not any miscalculation. The criterion used to judge the output quality is doubted. After a discussion, the reason is found. That is because we obtain high standard deviation of triangles` area when there are two classes sub groups of triangles such that some are very small and some are big. However, the standard deviation of each group is small. That is what happened with the new algorithm. Therefore the standard deviation of triangles` area was a bad measure for the triangulation quality. The problem of making the comparison in triangle groups was not treated in this paper but should be done in the future work.

Then the mean of the output triangles' minimum interior angles was used as the new quality evaluation criteria. The "Average angle" columns in the tables indicate that the new algorithm got the lower values than ear clipping on both types of polygons. It reflects the new algorithm produced higher triangulations quality as the matter of fact.

The time complexity of ear clipping is O(n^2) and the new algorithm is over O(n^3) in a rough analysis. In this aspect, the new algorithm needs to be improved. And many structures are used in the new algorithm and it has lots of operations on lists and arrays, so it is a little harder to implement than ear clipping.

The new algorithm is not stable in the testing. Sometimes it decomposed the polygon in the middle as expected but sometimes it does not. Because the vertices with highest or "second" highest visibilities are not always distribute around the middle of the polygon. Some other conditions should be set to make the middle cutting more accurately.

8 Conclusion

A new algorithm for simple polygon triangulation is introduced in this paper. It can reduce the sharp triangles in the triangulated polygon and produce higher output quality than ear clipping on both types of tested polygons. The new algorithm has a big time complexity and is harder to implement than ear clipping. That is because it implemented in a straightforward way and the visibility computing step costs lots of running time. The random polygon generation algorithm described in this paper has some problems. First, the polygons shapes do not change too much when the vertices number over 60. Second, sometimes lots of vertices are collinear. Both the polygon generation algorithm and the new triangulation algorithm could be improved in the future works. And this study point out that the standard deviation is not a good measure for the triangulation quality.

References

[1] Joseph O,Rourke."Computational Geometry In C second edition", Cambridge University Press, 1998.

[2] David Eberly (1998). "Triangulation by Ear Clipping". Geometric Tools, LLC. http://www.geometrictools.com/

[3] G.H. Meisters, Polygons have ears, Amer. Math. Monthly, vol. 82, pp. 648–651, 1975.

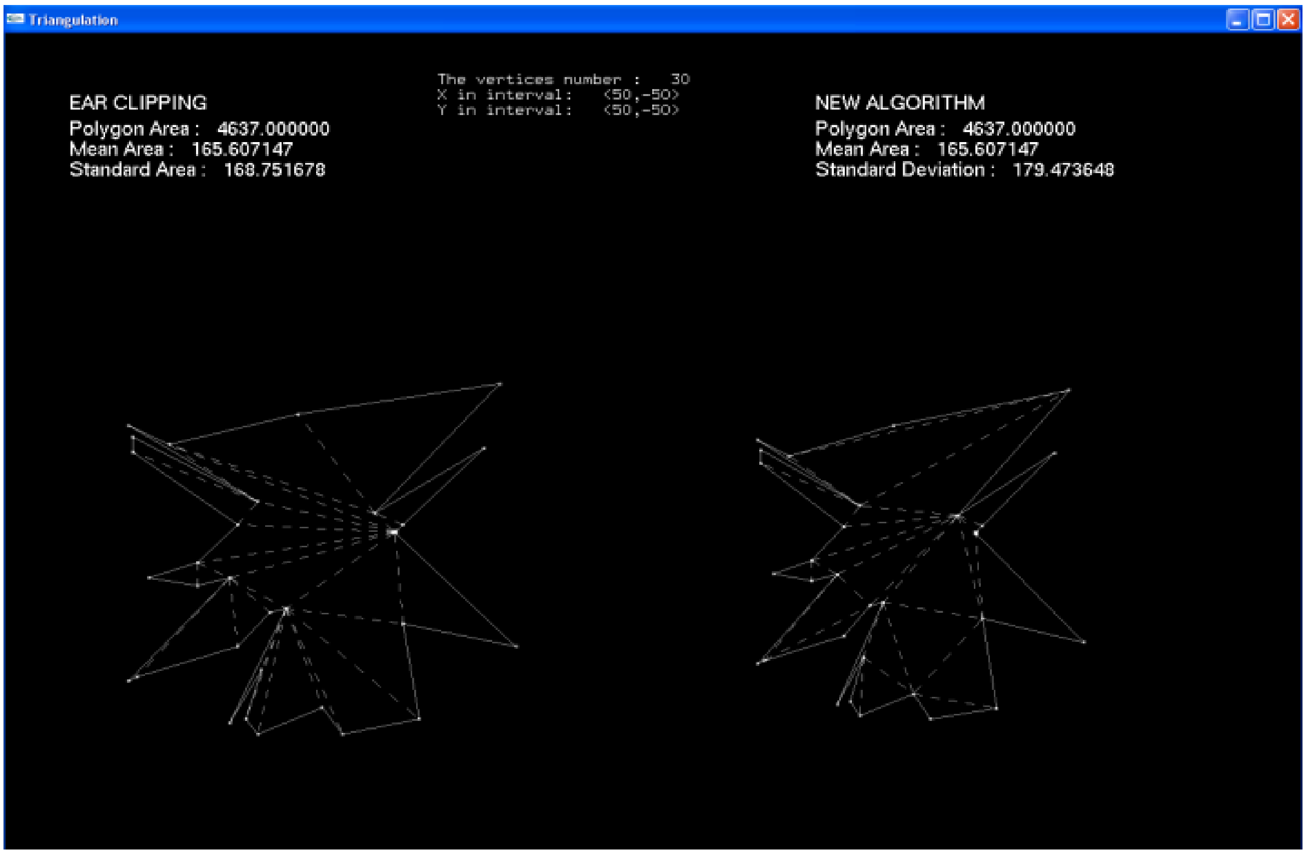
[4] Jacob E. Goodman, Joseph O`Rourke, "Handbook of discrete and computational geometry second edition", chapter 25, Marshall Bern, "Triangulation and mesh generation", pp.563-582. 2004. ISBN 1-58488-301-4

[5] ElGindy, H., Everett, H., and Toussaint, G. T., (1993) "Slicing an ear using prune-and-search," *Pattern Recognition Letters*, **14**, (9):pp.719-722.

[6] Jacob E. Goodman, Joseph O`Rourke, "Handbook of discrete and computational geometry second edition", chapter 26, Joseph O`Rourke and Subhash Suri, "Polygons", pp.583-606. 2004. ISBN 1-58488-301-4

[7] Jacob E. Goodman, Joseph O`Rourke, “Handbook of discrete and computational geometry second edition”, chapter 28, Joseph O`Rourke, “Visibility”, pp.643-663.583-606. 2004. ISBN 1-58488-301-4
[8] L. DE FLORIANI, E. PUPPO, An On-Line Algorithm for Constrained Delaunay Triangulation, *Graphical Models and Image Processing*, **3** (1992), pp.290-300.
[9] B. A. LEWIS, J. S. ROBINSO, Triangulating of planar regions with applications. *Compute. J.*, **4** (1979), pp.324-332
[10] Ruppert, Jim (1995). "A Delaunay refinement algorithm for quality 2-dimensional mesh generation". *Journal of Algorithms* **18** (3): ppt.548–585.
[11] Marko Lamot and Borut Zalik. “A Contribution to Triangulation Algorithms for Simple Polygons”. Journal of Computing and Information Technology-CIT 8,**4** (2000), ppt.319-331,

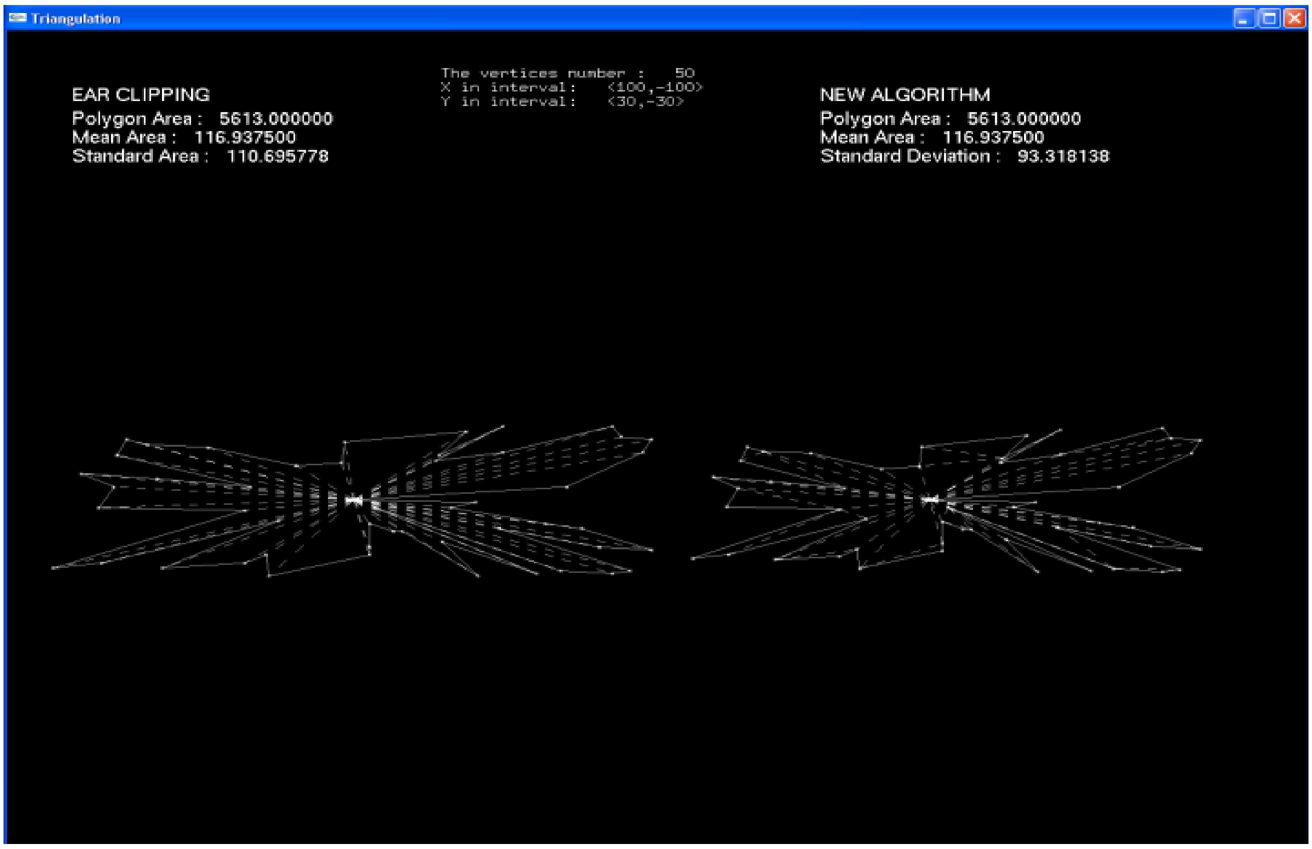
Appendix 1: Complete results



```
The vertices number :   30
X in interval:   <50,-50>
Y in interval:   <50,-50>
```

```
EAR CLIPPING
Polygon Area :   4637.000000
Mean Area :   165.607147
Standard Area :   168.751678
```

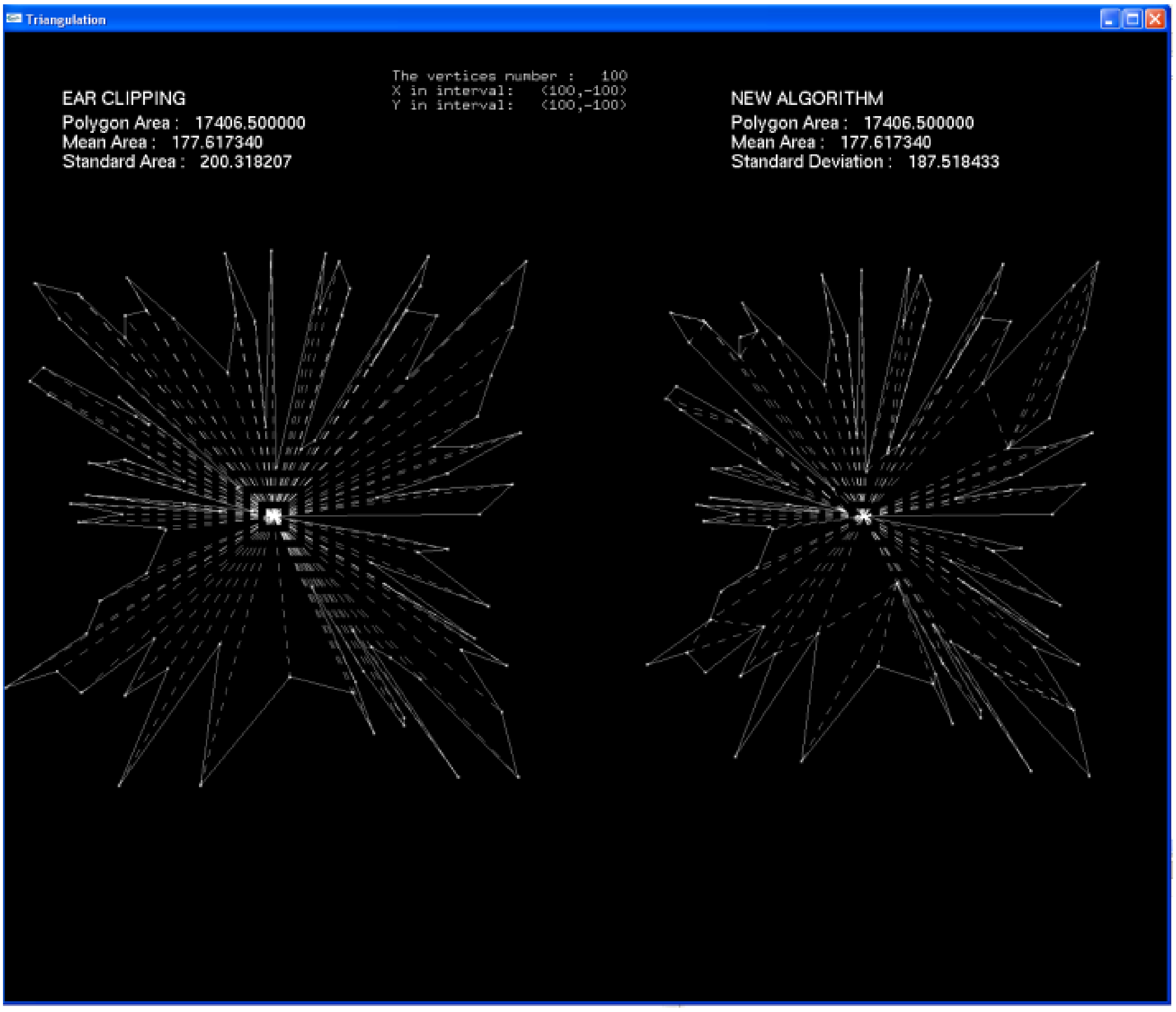
```
NEW ALGORITHM
Polygon Area :   4637.000000
Mean Area :   165.607147
Standard Deviation :   179.473648
```



```
The vertices number :   50
X in interval:   <100,-100>
Y in interval:   <30,-30>
```

```
EAR CLIPPING
Polygon Area :   5613.000000
Mean Area :   116.937500
Standard Area :   110.695778
```

```
NEW ALGORITHM
Polygon Area :   5613.000000
Mean Area :   116.937500
Standard Deviation :   93.318138
```



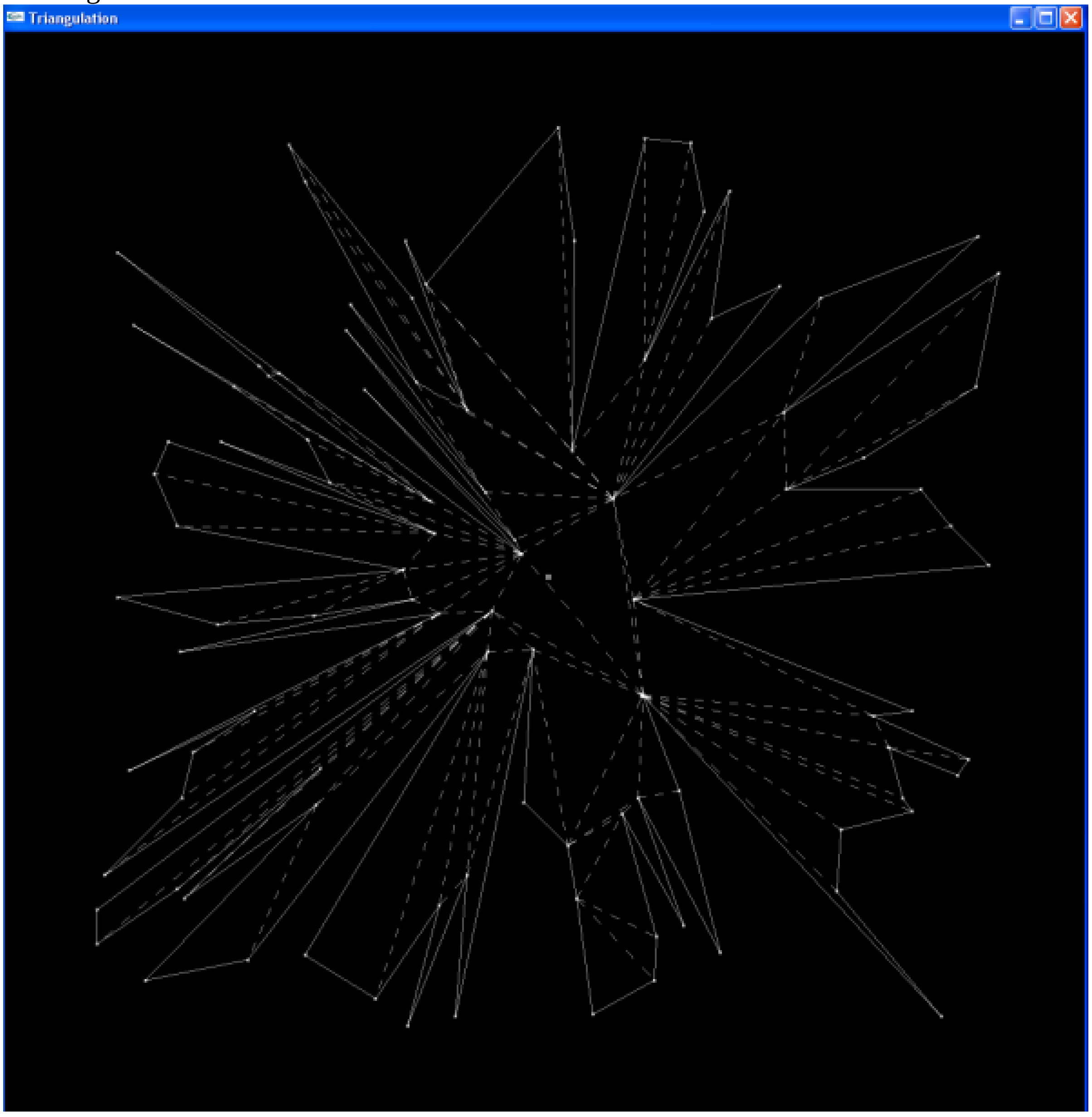
Triangulation

The vertices number : 100
X in interval: (100,-100)
Y in interval: (100,-100)

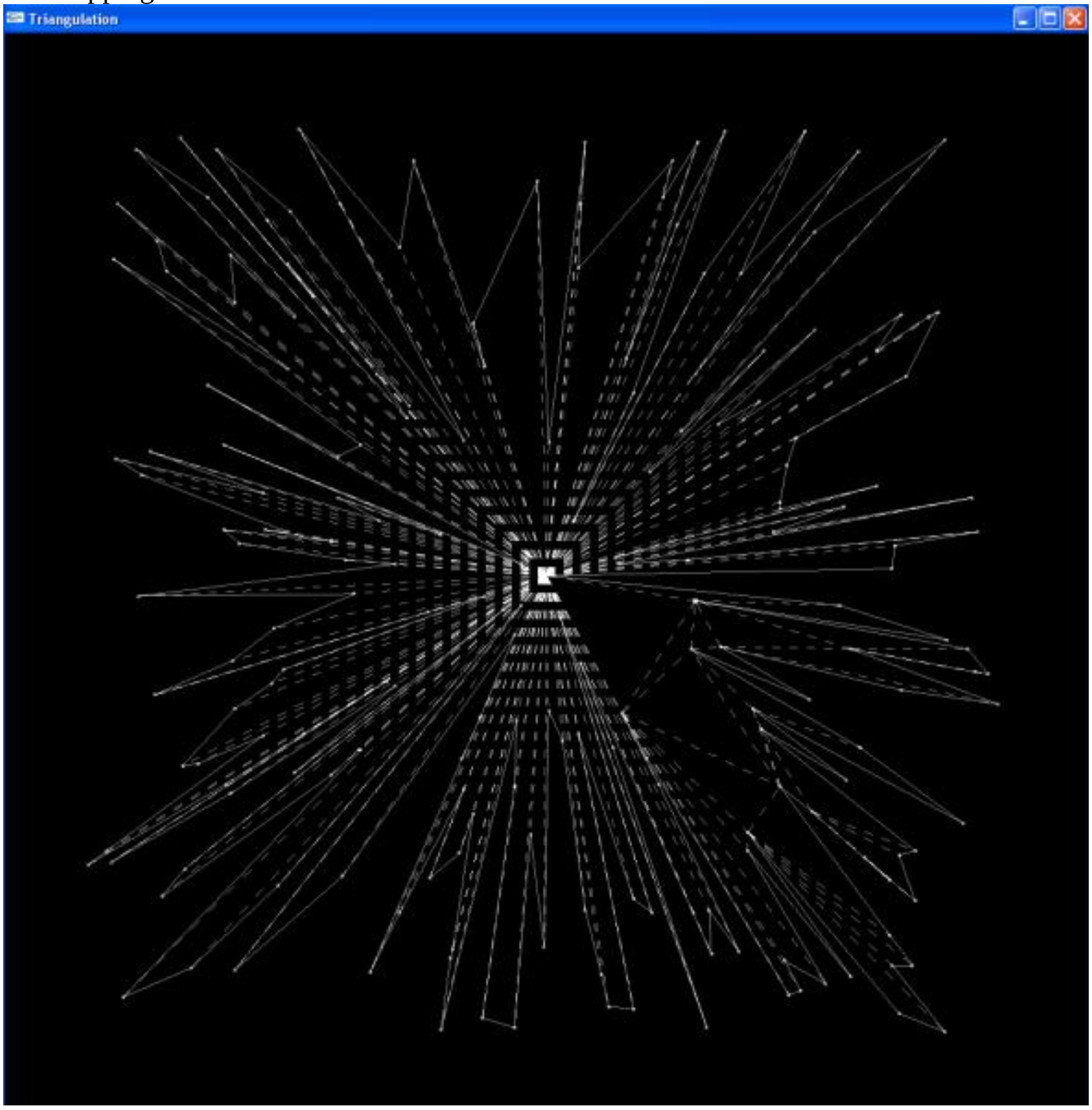
EAR CLIPPING
Polygon Area : 17406.500000
Mean Area : 177.617340
Standard Area : 200.318207

NEW ALGORITHM
Polygon Area : 17406.500000
Mean Area : 177.617340
Standard Deviation : 187.518433

Polygon with 100 vertices
New algorithm.



Polygon with 200vertices
Ear Clipping.


Triangulation