**Objective**

To understand the process on how to build Alpha Shapes using techniques from [4], first published in 1983. In addition to alpha shapes, Delaunay Triangulations and Voronoi Diagrams are covered.

**Introduction**

Euclid’s contribution to the field of geometry, especially the Euclidian construction was the fundamental rock that started all. The Euclidian construction consists of an algorithm and a proof[3]. It’s unambiguous, correct and terminating. For the next 2000 years, even though geometry was advancing, the analysis of algorithm was in decline[3]. The consequence was the introduction of “proof by contradiction” which made the life easier for mathematicians [3].

Geometry and the analysis of algorithms may be explained as computational geometry. Indeed, analysis and design of algorithms in the 1970s opened a door for Computational Geometry [1]. In early computational geometry, algorithms were slow or difficult to understand and implement [1].

The field has advanced and become useful for many other domains including computer graphics, computer-aided design and manufacturing, geographic information systems and robotics [1][7].

One of the most important structures in the field of computational geometry are convex hulls [2], which can be though as a set of nails in an space S, such that when a rubber band is release, it will take the form of a convex hull [1][2] as shown in figure 1. In other words, a convex hull is the minimum convex set that contains S [1]. This report will not go over convex hulls, which were covered in class. However, alpha shapes are a generalization of convex hulls.

Before going over alpha shapes, a small background information about Voronoi diagrams and Delaunay graphs will be introduced, with a bigger concentration in Delaunay graphs.
Voronoi Graphs

One of the questions in [1] is “the post office problem”. How to place a new site/office given existing ones. An easy way to visualize this problem is thinking of Starbucks. They start planning for a new store and need to find out where to place it. An example is shown in figure 2 [2]. They will have information related to the distance that people are willing to travel for coffee, the maximum capacity for each store with some minimum profit requirements. All of these variables may determine a desired distance for a new store. Voronoi diagram is the solution to this problem.

The formalization of the problems starts with the basic distance formula shown below:

$$\text{dist}(p,q) := \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

Letting P be the sets of points from $p_1$ to $p_n$, and the dist formula shown above, the definition of Voronoi diagrams can be formalized.
A Voronoi diagram is defined as the subdivision of the plane into \(n\) cells, one for each site \(P\). In addition, the property that a point \(q\) lies in the cell corresponding to a site \(p_i\) if and only if, the \(\text{dist}(q, p_i) < \text{dist}(q, p_j)\) for each \(p_j \in P\) with \(j \neq i\) [1]. With points \(p\) and \(q\) a bisector of \(p\) and \(q\) is defined as the perpendicular bisector of the line segment \(pq\) [1]. This bisectors divides them into two half planes. Therefore, a half plane \(h(p, q)\) corresponds to the half that contains \(p\) and vice versa[1].

With this in mind, The cell of Voronoi diagram \(\text{Vor}(P)\) can be seen as the intersections of \(n-1\) with at most \(n-1\) vertices and at most \(n-1\) edges. One can query if \(r\) is in any of this half planes such that \(r \in h(p, q)\) iff \(\text{dist}(r, p) < \text{dist}(r, q)\). Therefore, the following observation is shown below:

\[
V(p_i) = \bigcap_{1 \leq j \leq n, j \neq i} h(p_i, p_j)
\]

Three additional theorems of importance will not be shown with proof. However, the proofs are found in [1].

Figure 2 – The Post Office Problem[2]
Theorem 7.2: Let P be a set of n point sites in the plane. If all the sites are collinear then Vor(P) consist of n-1 parallel lines. Otherwise, Vor(P) is connected and its edges are either segments of half-lines.

Theorem 7.3: for \( n \geq 3 \), the number of vertices in the Voronoi diagram of a set of n point sites in the plane is at most \( 2n-5 \) and the number of edges is at most \( 3n-6 \).

Theorem 7.4 : For the Voronoi diagram Vor(P) of a set of points P the following holds:

(i) A point q is a vertex of Vor(P) if and only if its largest empty circle \( C_p(q) \) contains three or more sites on its boundaries

(ii) The bisector between sites \( p_i \) and \( p_j \) defines an edge of Vor(P) if and only if there is a point q on the bisector such that \( C_p(q) \) contains both \( p_i \) and \( p_j \) on its boundary but no other site.

Finally, the Voronoi diagrams runs in \( O(n \log n) \) using Fortune’s algorithm. It’s important to notice that improving the algorithm to linear time is not feasible because it requires sorting which is known to be \( \Omega(n \log n) \)[1]. The pseudo-code is shown in listing 1.
let \*(z) be the transformation \*(z) = (zx,zy + d(z)), where d(z) is a parabola with minimum at z
let T be the "beach line"
let Rp be the region covered by site p.
let Cpq be the boundary ray between sites p and q.
let p1,p2,...,pm be the sites with minimal y-coordinate, ordered by x-coordinate

create initial vertical boundary rays

while not IsEmpty(Q) do
  p ← DeleteMin(Q)
  case p of
    p is a site in \*(V):
      find the occurrence of a region \*(Rq) in T containing p, bracketed by Crq on the left and Cqs on the right
      create new boundary rays and with bases p
      replace * (Rq) with \*(s) in T
      delete from Q any intersection between Crq and Cqs
      insert into Q any intersection between Crq and
      insert into Q any intersection between and Cqs
    p is a Voronoi vertex in * (V):
      let p be the intersection of Cqr on the left and Crs on the right
      let Cuq be the left neighbor of Cqr and
      let Csv be the right neighbor of Crs in T
      create a new boundary ray if qy = sy,
      or create if p is right of the higher of q and s,
      otherwise create
      replace Cqr, * (Rr),Crs with newly created Cqs in T
      delete from Q any intersection between Cuq and Cqr
      delete from Q any intersection between Crs and Csv
      insert into Q any intersection between Cuq and Cqs
      insert into Q any intersection between Cqs and Csv
      record p as the summit of Cqr and Crs and the base of Cqs
      output the boundary segments Cqr and Crs
  endcase
endwhile
output the remaining boundary rays in T

Delaunay graphs

Delaunay graphs deals with triangulation. What is the most effective way to triangulate a set of points[1]. Delaunay, as it will be shown below has some very special properties that enables “the cell tower” problem or alpha shapes (in my case) to be performed quickly and with correctness.

The Cell phone tower, as shown in Figure 3, allows asking a basic question. Given a Delaunay triangulation, which is the closest neighboring tower given the positing of $X_b$ and $X_c$ [6].

Let’s start by some basic definitions of Delaunay triangulations.

Theorem 9.1: Let $P$ be a set of $n$ point in the plane, not all collinear and let $k$ denote the number of points in $P$ that lie on the boundary of the convex hull of $P$. Then any triangulation of $P$ has $2n-2-k$ triangles and $3n-3-k$ edges.  

A basic concept needed to create Delaunay graphs is “edge flipping.” Edge flipping states that for a triangulation $T$ is said to be optimal if $A(T) \geq A(T’)$ where $A(T) := (\alpha_1, \alpha_2, \ldots \alpha_3)$ and $A(T’) := (\alpha’_1, \alpha’_2, \ldots \alpha’_3)$ ($\alpha$ are angle-vector.)[2]. In addition, this is constraint to $\alpha_i = \alpha’_j$ for all $j < i$ , otherwise $\alpha_i > \alpha’_j$ are required as well. [1].
Without a formal proof, the concept edge flipping needs to be explain. This can be performed as shown in figure 4. The question remains: How to know if edge flipping is needed. In [1] Lemma 9.4 states that an edge $p_i p_j$ is illegal if and only if $p_i$ lies inside the interior of $p_k$. This property will be useful for the alpha shapes. If the four points in figure 5 form a convex quadrilateral and do not lie on a common circle, then either $p_i p_j$ or $p_k p_l$ is an illegal edge [1]. The algorithm is shown in listing 2.
LEGALIZE_EDGE($p_r, p_ip_j, T$)

1. if $p_ip_j$ is illegal

2. then Let $p_ip_jp_k$ be the adjacent triangle to $p_ip_jp_i$ along $p_ip_j$

3. Replace $p_ip_j$ with $p_ip_k$

4. LEGALIZE_EDGE($p_r, p_ip_k, T$)

5. LEGALIZE_EDGE($p_r, p_kp_j, T$)

Listing 2: Edge Flipping [2]

In general, the Delaunay graph of $P$ is an embedding of the dual graph of the Voronoi diagram as shown previously in figure 3.

The topic of Delaunay graphs and Voronoi diagrams are much more detailed that is covered here (e.g [1]). Nevertheless, this is enough to go into alpha shape construction.

**Alpha Shapes**

If given a set of point, is it possible to describe a shape? Alpha Shape is a generalization of the convex hull. When $\alpha = \infty$ then is a convex hull. Alpha shapes. I will be concentrating in negative alpha hulls. Figure 6 shows the positive and the alpha hulls output after process. The reason for this, other than the process is fairly similar, one can see at the naked eye the prefer one. The negative alpha hull describes the letter A in figure 6.
When thinking about alpha shapes, the best definition that can describe the problem is one of many found in [2]:

*the α-hull (α<0) of given set of points in S, can be defined as the complement of the union of all open discs of radius not less that -1/α which contains no point of S [4].*

It’s possible to fill this report with many observations and proof found in [2] but I rather finished this with a step-by-step process of creating an α-shape. Listing 3 gives a high level algorithm.

**AlphaShape(Given a set points S and alpha value)**

1. Construct Delaunay Triangulation of S
2. Determine Alpha Extreme points of S
3. Determine Alpha-Neighbors of S
4. Output Alpha Shape

**Listing 3: High Level Alpha Shape Algorithm**
The first step calls to construct the Delaunay triangulation. This algorithm can be found in [2] and it is shown in Listing 4 with some small modification for better understanding. Note that if different size of $\alpha$ want to be tested, point 1 can be skipped.

**DELAUNAYTRIANGULATION(P)**

1. Build bounding triangle with points $p_1p_2p_3$ to contained all points in $P$
2. $T \leftarrow$ Bouding Triangle
3. Randomize set $P$ (* For faster performance * )
4. for $r \leftarrow 1$ to $n$
5. do
6. find a triangle in $p_ip_jp_k \in T$ containing $p_r$
7. if $p_r$ lies in the interior of the triangle $p_ip_jp_k$
8. then add edges from $p_r$ to three vertices of $p_ip_jp_k$
9. LEGALIZEEDGE($p_r$, $p_ip_j$, $T$)
10. LEGALIZEEDGE($p_r$, $pjp_k$, $T$)
11. LEGALIZEEDGE($p_r$, $p_kp_i$, $T$)
12. else
13. Add edges from $p_r$ to $p_k$ and the third vertex $p_l$ belonging to the other triangle incident to $p_ip_j$
14. LEGALIZEEDGE($p_r$, $pipl$, $T$)
15. LEGALIZEEDGE($p_r$, $pipj$, $T$)
16. LEGALIZEEDGE($p_r$, $pjpk$, $T$)
17. LEGALIZEEDGE($p_r$, $pkpi$, $T$)
18. Discard (*after for loop*) bounding triangle in $T$.

**Listing 4: Delaunay Triangulation Algorithm**

In general, this algorithm is fairly straightforward. The bounding triangle is created and later discarded just for to be able to calculate the triangulations. One can see that in line 7, The “if” statement is trying to find out if $p_r$ is in the interior of the triangle. It’s not so clear how to know if the point is in the interior or not.
The legalization of the edges can be further explained for the purposes of finding the extreme points in step 2 of listing 3. First, let’s find out how to create a circle from a triangle, which will be needed. This circle has radius \(-1/\alpha\). In listing 5, this circle will play a major role.

With the knowledge of points a, b, c, and d and some properties shown in figure 7, we can construct a circle. The first step is to find the midpoints \(M_b\) and \(M_c\). Finding the vector from c to a can be easily accomplished by \((-y,x) = (y,x)\). After having found the midpoints, one can imagine two vectors intersecting each other (bisector) at point p. To find this point, which will be the center of the alpha shape circle, one must find \(t\) which is the length from b axis to p. Other than \(t\), all the values are known. Therefore, \(t\) can be found using the following equation:

\[ (V_c \cdot t + (M_c-M_b)) \cdot (c-a) = 0 \]

\[ M_c = \frac{(a+b)}{2} \]
\[ M_b = \frac{(a+c)}{2} \]
\[ V_c = (b-a)^\perp \]

\[ p = M_c + V_c \cdot t \]
\[ p - M_b \cdot (c-a) = 0 \]
\[ (V_c \cdot t + (M_c-M_b)) \cdot (c-a) = 0 \]

Figure 7: Circle Creation[5][6]

After constructing a circle, the question remains. Is p and q points an alpha edge? Listing 5 and figures 8 and 9 show how to answer if p and q are extreme points (alpha edges.) Note that the check must be done to the right and left. Alpha edges can be found in both checks. Listing 5 shows only the left check. The right check is the same just with the D circle to the right and swap: \( a \rightarrow b, b \rightarrow a, c \rightarrow d, d \rightarrow c \)[6].
Is_{\text{AlphaEdge\_LEFT}}(T, \alpha_c)

1. If \(a \& \& b\) points form a circle
2. Construct two circles one to the left and one to the right with points \(c\) and \(d\).
3. If \(c\) is a corner of the bounding triangle return YES.
4. If \(\alpha_c \leq \frac{1}{2}|ab|\) or \((ca \cdot cb) \leq 0\) return NO
5. If \(db \cdot da > 0\) then return YES
6. If \(db \cdot da < 0\) and \(\alpha_c > D_c\) then return YES (*\(D_c\) is the circle to the left*)
7. return NO.

Listing 5: AlphaEdge[6]

Figure 8: Finding an alpha edge when \(db \cdot da > 0\)
Finally, once all the alpha-edges (extreme edges) are found, the Delaunay edges have the information to connect all of those edges.

**Conclusion**

The experience of this project has sparked an interest on Delaunay triangulation and computational geometry as a whole. My next step is to build a java program that finds alpha shapes displaying step by step, similar to demo provided by [] that finds Delaunay triangulation with graphical output.

A final remark about alpha shapes does not seem to be of big interest, which can be concluded from the amount of papers working with alpha shapes. There are a few variations of alpha shapes, including weighted alpha shapes.
References

6. Notes and handouts from Dr. Victor Milenkovic

1 [The next three theorems are Verbatim from book [2]] (chapter 7)


3 Fortune Algorithm in C can be found at : http://ect.bell-labs.com/who/sjf/voronoi.tar

4 [The next three theorems are Verbatim from book [2]] (chapter 9)