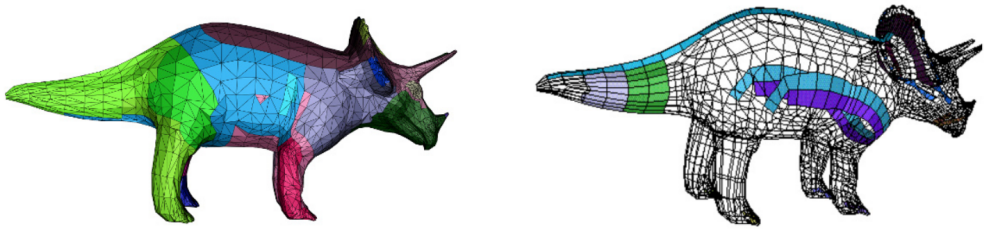
Convex Polygon Triangulation

Comp 221 - Final Project

Christian Lentz & Nolan Meyer

12/9/2022



1 Introduction: What is Triangulation?

Polygon Triangulation is a classic problem and key task in computational geometry. In general, a triangulation is the partition of some set of points or a simple polygon into a set of disjoint, or non-overlapping triangles. In this paper, we will emphasize the triangulation of a simple convex polygon, in which we look to find the set of disjoint triangles whose union is the convex polygon. This may seem like a trivial task, but there are a number of details that make triangulation a difficult problem to compute.

Some of these details include the distinction between the triangulation of a point set versus a polygon, as well as the distinction between various classes of polygons. With that, there are also a number of important questions to ask: What is this triangulation for? Do we wish to find one triangulation, or to enumerate all possible triangulations? For our purpose, is there a best or worst triangulation? Finally, how do the answers to these questions affect the approach needed to compute, and the effort that is required to do this computation?

2 Polygon Types

The best place to start is to define what is meant by a polygon that is simple versus complex, as well as the distinction between a polygon that is convex and concave. A simple polygon encloses a single interior space (i.e. no holes) and does not have intersecting sides. A complex polygon is a polygon that encloses a single interior space, but may have holes and/or intersect itself.

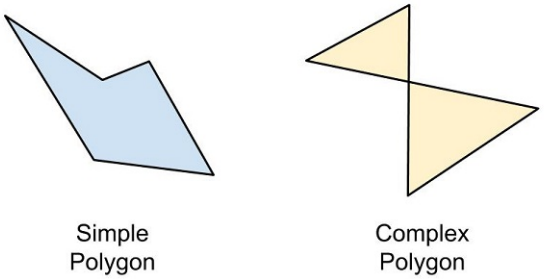


Figure 1: Examples of a simple polygon and a complex polygon

For our purpose, we will focus on simple polygons. More specifically, we will focus on a class of simple polygons known as convex polygons. Convex polygons are those which do not include any edges that point inward towards the center of the polygon, meaning that any two points p and q within the polygon cannot be connected with a chord such that the chord leaves the polygon. From this, it is clear that a convex polygon does not contain any interior angles which are greater than 180 degrees. On the other hand, a concave polygon does contain one or much such interior angles. For example, the polygon on the left is concave, while the polygon on the right is convex.

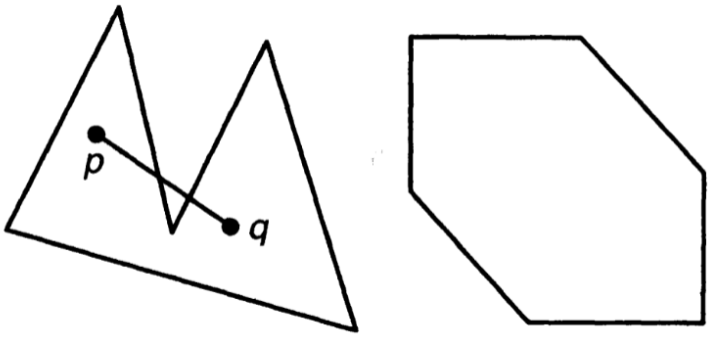


Figure 2: An example of a concave polygon (left) and a convex polygon (right)

It is important to understand these distinctions moving forward.

3 Point Sets versus Convex Hull

In order to perform triangulation, a set of points is required as an input. This point set could be a random assortment of points in a plane, or it could represent the vertices of a polygon. Triangulating a random point set using a convex polygon triangulation technique will require an additional pre-processing step: finding the convex hull of the points. The convex hull of a set of points is the smallest convex polygon that encloses all of the points. Once the convex hull is found, the problem simplifies into convex polygon triangulation.

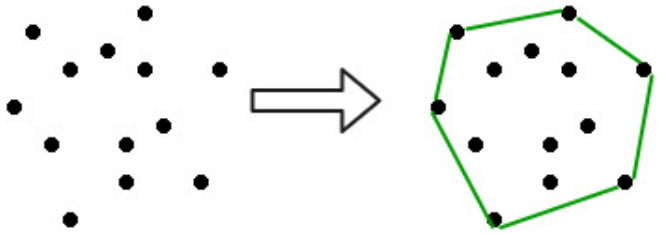


Figure 3: Finding the convex hull of a set of points

In both cases, we will require the set of points representing the polygon to follow two rules:

1. Adjacent vertices are connected by an edge
2. The set is ordered counter-clockwise

For example, the polygon below could be represented as:

$$V = \{v_a, v_b, v_c, v_d, v_e, v_f\}$$

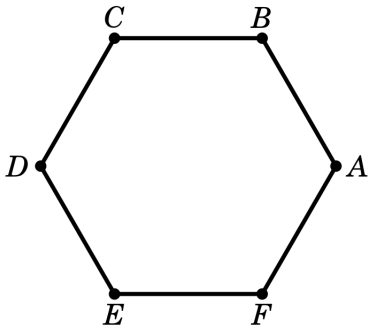


Figure 4: Converting from a convex polygon to an ordered set of points

As we are focusing only on 2D triangulation, the points will be represented by their (x, y) coordinates in the plane.

4 Applications in Computer Graphics

Now that we have formally defined the relevant objects and techniques, it is now possible to discuss some of the applications of this problem. One of the most important practical applications of triangulation is in the field of Computer Graphics. Much of this work is grounded in Computational Geometry, which is the field of Computer Science that is concerned with the development of efficient algorithms to solve and compute geometric problems.

Computer graphics has to do with the production of images for use within film, computer animation, video games, 3D models, image processing and numerous other media and applications. Among the many disciplines of science that are necessary for work in computer graphics, Computational Geometry and triangulation turn out to be extremely important.

Although we are focusing on 2D triangulation, it is also possible to triangulate 3D objects and

shapes. This is a more difficult and computationally intensive task. However, the 3D problem relies on a mapping to two dimensions, and thus relies on algorithms that are extremely similar to those used for 2D triangulation.

One example that illustrates the close relationship between the 2D and 3D triangulation problem is that a (clever) triangulation of a set of points in 2D can actually be used to represent elevation/depth and 3D images. Algorithms used to do this must make choices about which is the best triangulation for the task. Often, this means allowing the width of the triangles to be no less than a given threshold.

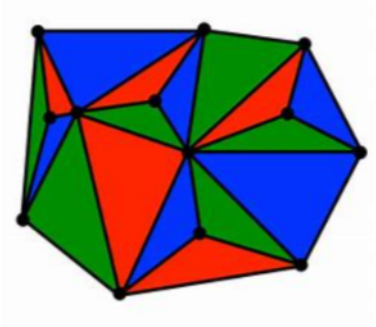


Figure 5: A 2D triangulation which encodes elevation

5 Applications in Mathematics

Some of the more abstract and theoretical uses of polygon triangulation are across various fields of mathematics and computer science, such as graph theory and combinatorics. There are a number of interesting and sometimes unexpected results that we can prove about polygon triangulation.

5.1 Graph Theory

Triangulation can be related to chorded cycles and graph coloring. A chorded cycle is a cycle on n vertices with non-intersecting chords (also known as diagonals) which connect non-consecutive points along the cycle. A graph coloring is any coloring of the vertices of a graph such that no two vertices which are adjacent have the same color. It can be shown inductively that any chorded cycle with $n \geq 3$ is three-colorable. This can be done by proving that the graph can be colored in three colors or fewer. In doing this, it is crucial to show that the chorded cycle with the maximum number of chords is also three-colorable. Such a graph is the triangulation of the convex polygon formed by the cycle on n vertices. In general, it can be said that a triangulation of a n-sided convex polygon (with $N \geq 3$) is a maximal planar graph and is three-colorable.

5.2 Combinatorics

An important question to ask about polygon triangulation is how many such triangulations exist for a given convex polygon on n vertices. It turns out that these can be counted by a famous counting sequence known as the Catalan numbers, a result first shown by Leonhard Euler.

Let h_n be the number of triangulations of a convex polygon on $n+1$ sides where $n-2$ non-intersecting diagonals are used to to split the polygon into $n-1$ disjoint triangular regions. Euler proved that h_n satisfies the recurrence relation:

$$h_n = \sum_{k=1}^{n-1} h_k h_{n-k}, (n \geq 2)$$
$$h_1 = 1$$

The insight here is that each time a chord is added to a triangulation, a right polygon (h_k) and left polygon (h_{n-k}) are created, and are related in a multiplicative way. The solution to the recurrence relation is:

$$h_n = \frac{1}{n}\binom{2n-2}{n-1}, (n \in \mathbb{N})$$

This solution is found via a generating function, and is simplified to the form of the Catalan Numbers using Newton's Binomial Theorem. Thus, polygon triangulations can be counted using the Catalan numbers.

6 Common Implementations and Efficiency

6.1 Seidel's Algorithm

Seidel's algorithm is an incremental randomized algorithm first developed in the early 1990s by Raimund Seidel. It is one of the best polygon triangulation algorithms known to date for two reasons. First, compared to some other (slightly more efficient) triangulation algorithms, Seidel's algorithm is more practical and easier to implement. Second, its expected run-time is very close to the lower time bound of triangulation, making it usable in the real world.

Seidel's algorithm can be summarized in three steps:

1. Decompose the polygon into trapezoids (trapezoidation) by considering every permutation by which we can use $n-2$ non-intersecting and non-horizontal line segments of the polygon.
2. Decompose the trapezoids into monotone polygons (if necessary).
3. Triangulate the monotone polygons.

The image below illustrates the process by which Seidel's algorithm uses trapezoidation to partition a polygon into smaller and disjoint polygons that are easier to triangulate.

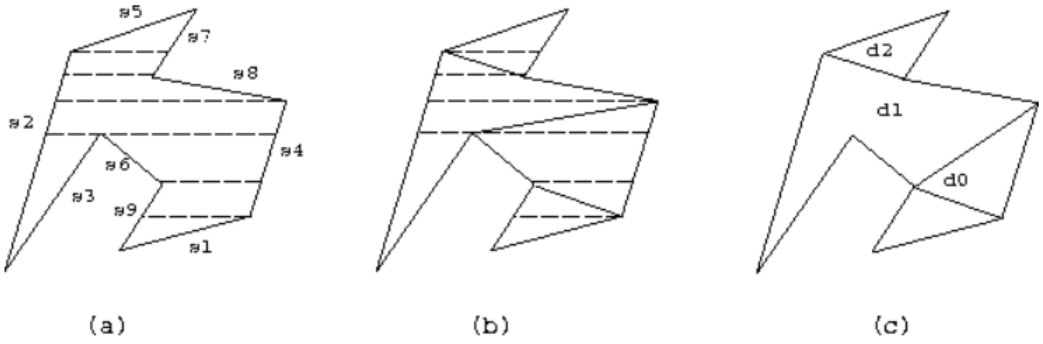


Figure 6: Steps of Seidel's algorithm visualized

6.1.1 Seidel's Efficiency

By analysing each part of Seidel's algorithm, we can determine its Big-Oh time complexity.

- Decompose the polygon into trapezoids - proven by Seidel to take $O(n \log^* n)$ time
- Decompose the trapezoids into monotone polygons - linear time, $O(n)$
- Triangulate the monotone polygons - linear time, $O(n)$

Thus, the time complexity for Seidel's algorithm is $O(n \log^* n)$ time. In practice however, it is almost linear run-time for simple polygons of n vertices.

6.2 The Ear Clipping Theorem

Convex polygon triangulation via ear clipping is based on an important insight known as *The Two Ears Theorem*, which states that any simple polygon with $n \geq 4$ sides has at least two ears. An ear is defined as a triangular region that is within the polygon, which has two sides that are part of the polygon, and one side which is within the polygon. The algorithm follows these general steps:

1. Find an ear
2. Remove the Ear
3. Repeat this process until the base case ($n = 4$)
4. Split the four sided shape into two triangles and return

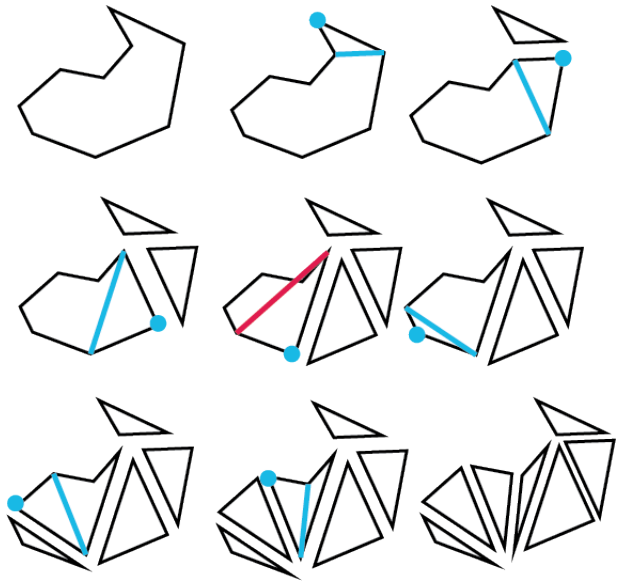


Figure 7: Depiction of ear clipping. Valid diagonals shown in blue, invalid in red.

6.2.1 Ear clipping Efficiency

The most basic and straightforward approach to implementing ear clipping can be done in $O(n^3)$ time. However, with slight modifications a $O(n^2)$ solution exists, and was discovered Godfried Toussaint.

In essence, the algorithm loops over all of the vertices in the polygon to find a potential ear (three consecutive vertices), and then must check to see if the segment connecting the "first" and "third" vertices is a valid diagonal (as shown above). Both of these steps take $O(n)$ time and are nested loops within each other, leading to $O(n^2)$ run-time.

Although ear clipping is considered to be one of the easier implementations of polygon triangulation, there are a number of faster known algorithms.

7 Our Implementation

Our implementation of convex polygon triangulation uses the Ear Clipping method, and follows much of the logic that is outlined in this paper. The implementation is designed to follow a general process that is interactive with the user:

1. Generate a random set of points in the plane.
2. Find the convex hull of those points and remove the rest.
3. Pass those points to the Ear Clipping algorithm as defined above in section three.
4. Perform ear clipping.
5. Return a list of the triangles formed.

In order to make this work, our version makes use of a few other well known algorithms, including *Selection Sort* and *Quick Hull*. It is useful to note that more efficient sorting algorithms may be useful as the number of randomly generated points increases.

7.1 Pseudocode

Here we will outline high-level and in-depth pseudocode for ear clipping. Pseudocode for *Selection Sort* and *Quick Hull* will not be included.

7.1.1 High-level

```
Algorithm EarClipping(V)
// V is the set of vertices representing a polygon
// T is the list of triangulations
1.  Create a circularly linked list, vertices, of the n vertices in V
2.  While there are more than 3 vertices in V
3.      Locate a potential ear, {v_{i-1}, v_i, v_{i+1}}
4.      Check if {v_{i-1}, v_{i+1}} is a valid diagonal
5.          If it is, add {v_{i-1}, v_i, v_{i+1}} to T
6.          Remove v_i from the list of vertices
7.      Iterate through the rest of the points in V
8.  Return T
```

7.1.2 In-Depth (closer to actual code)

```
Algorithm EarClipping(V)
// V is the set of vertices representing a polygon
// T is the list of triangulations
1.    let vertices = a circularly linked list of the n vertices in V
2.    let node = vertices.first()
3.    while vertices.size() > 3 do
4.        let v0 = node.previous.data
5.        let v1 = node.data
6.        let v2 = node.next.data

      //Check if diagonal intersects any other edge in polygon
7.        let is_diagonal = CheckDiagonal(v0, v2)
8.        if is_diagonal do
9.            T.add({v0, v1, v2})
10.           vertices.remove(node.data)

11.       let node = node.next
12.   return T
```

8 Conclusion

This paper was aimed at providing an in depth look into the geometric problem of triangulation. Throughout, emphasis was placed on 2D convex polygons, as well as specific implementations to compute, including Seidel's Algorithm and the Ear clipping Algorithm. With that, we have also highlighted the key definitions, terminology and techniques. These include the distinction between polygon types and the triangulation of point sets versus polygonal shapes. We discussed the applications of triangulation in computer graphics, and described the ways in which this problem appears or is useful in graph theory and combinatorics. Finally, we provided high-level and in-depth pseudocode. All of this together should highlight the importance and usefulness of algorithmic tasks such as polygon triangulation, which on the surface may seem abstract or not entirely useful in the real world, but in fact offer a number of useful and practical applications.

References

[1] Richard A. Brualdi. *Introductory combinatorics*. Pearson, 2018.

[2] David Eberly. Triangulation by ear clipping - geometric tools. https://www.geometrictools.com/Documentation/TriangulationByEarClipping.pdf, Jul 2022.

[3] Bartosz Kajak. Improved algorithms for ear-clipping triangulation. *UNLV Theses, Dissertations, Professional Papers, and Capstones*, 2011.

[4] Gang Mei, John C Tipper, and Nengxiong Xu. Ear-clipping based algorithms of generating high-quality polygon triangulation. http://export.arxiv.org/pdf/1212.6038.

[5] Atul Narkhede and Dinesh Manocha. Fast polygon triangulation based on seidel's algorithm. https://gamma.cs.unc.edu/SEIDEL/.

[6] Yuvraj Nigade and Pushkar Newaskar. Polygon triangulation, triangulation algorithm. http://www.polygontriangulation.com/2018/07/triangulation-algorithm.html.

[7] Nils Olovsson. Ear clipping triangulation. http://www.all-systems-phenomenal.com/articles/ear_clipping_triangulation/index.php, Mar 2021.

[8] STEVEN S. SKIENA. *Algorithm Design Manual*. SPRINGER, 2021.

[9] Sunny Srinidhi. Circular double linked list implementation in java. https://blog.contactsunny.com/tech/circular-double-linked-list-implementation-in-java, Jan 2020.

[10] Wikipedia. Polygon triangulation. https://en.wikipedia.org/wiki/Polygon_triangulation#Related_objects_and_problems, Oct 2022.