Triangulation by Ear Clipping

David Eberly, Geometric Tools, Redmond WA 98052
https://www.geometrictools.com/

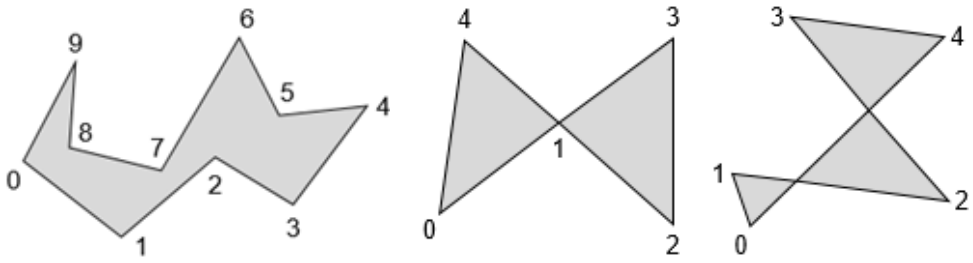
Created: November 18, 2002
Last Modified: August 3, 2024

Contents

1 Introduction

A classic problem in computer graphics is to decompose a simple polygon into a collection of triangles whose vertices are only those of the simple polygon. By definition, a simple polygon is an ordered sequence of n points, $\boldsymbol{V}_0$ through $\boldsymbol{V}_{n-1}$. Consecutive vertices are connected by an edge $\langle \boldsymbol{V}_i, \boldsymbol{V}_{i+1} \rangle$, $0 \le i \le n-2$, and an edge $\langle \boldsymbol{V}_{n-1}, \boldsymbol{V}_0 \rangle$ connects the first and last points. Each vertex shares exactly two edges. The only place where edges are allowed to intersect are at the vertices. A typical simple polygon is shown in figure 1.

Figure 1. The left polygon is simple. The middle polygon is not simple because vertex 1 is shared by more than two edges. The right polygon is not simple because the edge $\langle 0, 4 \rangle$ is intersected by two other edges at points that are interior to edge $\langle 0, 4 \rangle$.



simple nonsimple nonsimple

If a polygon is simple, the interior bounded region is always to one side as you traverse the edges. I assume that the polygon is counterclockwise ordered so that as you traverse the edges, the interior is to your left. The vertex indices in figure 1 for the simple polygon correspond to a counterclockwise order.

The decomposition of a simple polygon into triangles is called a *triangulation* of the polygon. Any triangulation of a simple polygon of n vertices has $n-2$ triangles. Various algorithms have been developed for triangulation, each characterized by its asymptotic order as n grows without bound. The simplest algorithm, called *ear clipping*, is the algorithm described in this document. The order is $O(n^2)$. Algorithms with better asymptotic orders exist but are more difficult to implement. Horizontal decomposition into trapezoids followed by identification of monotone polygons that are themselves triangulated is an $O(n \log n)$ algorithm [2, 3]. An improvement using an incremental randomized algorithm produces an $O(n \log^* n)$ where $\log^* n$ is the iterated logarithm function [5]. This function is effectively a constant for very large n that you would see in practice, so for all practical purposes the randomized method is linear time. An $O(n)$ algorithm exists in theory [1] but is quite complicated; to my knowledge, no implementation is publicly available for this algorithm.

All the figures in the document were drawn using Mathematica [6].

2 Ear Clipping

An *ear of a polygon* is a triangle formed by three consecutive vertices $\boldsymbol{V}_{i_0}$, $\boldsymbol{V}_{i_1}$, and $\boldsymbol{V}_{i_2}$ for which $\boldsymbol{V}_{i_1}$ is a *convex vertex*. At such a vertex, the interior angle at the vertex is smaller than π radians and the line segment from $\boldsymbol{V}_{i_0}$ to $\boldsymbol{V}_{i_2}$ lies completely inside the polygon. The ear has the property that no vertices of the

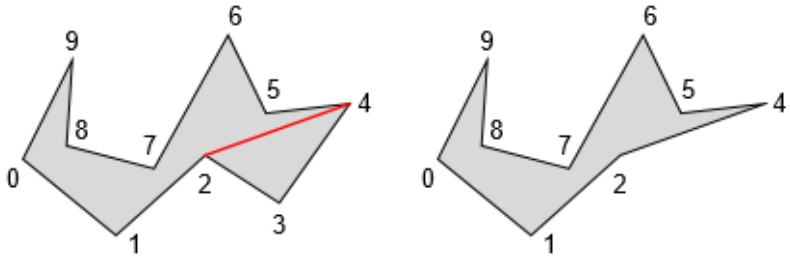
polygon are contained in the triangle other than the three vertices of the triangle. The line segment $\langle \boldsymbol{V}_{i_0}, \boldsymbol{V}_{i_2} \rangle$ is referred to as a *diagonal* of the polygon. The vertex $\boldsymbol{V}_{i_1}$ is called the *ear tip*. A triangle consists of a single ear, although you can place the ear tip at any of the three vertices. A polygon of four or more sides always has at least two nonoverlapping ears [4]. This suggests a recursive approach to the triangulation. If you can locate an ear in a polygon with $n \geq 4$ vertices and remove it, the remaining polygon has $n-1$ vertices and the process is repeated. A straightforward implementation of this will lead to an $O(n^3)$ algorithm.

With some careful attention to details, the ear clipping can be done in $O(n^2)$ time. The first step is to store the polygon as a doubly linked list so that you can quickly remove ear tips. Construction of this list is an $O(n)$ process. The second step is to iterate over the vertices and find the ears. For each vertex $\boldsymbol{V}_i$ and corresponding triangle $\langle \boldsymbol{V}_{i-1}, \boldsymbol{V}_i, \boldsymbol{V}_{i+1} \rangle$, test all other vertices to see if any are inside the triangle; the indexing is modulo n, so $\boldsymbol{V}_n = \boldsymbol{V}_0$ and $\boldsymbol{V}_{-1} = \boldsymbol{V}_{n-1}$. If none are inside, you have an ear. If at least one is inside, you do not have an ear. The actual implementation I provide tries to make this somewhat more efficient. It is sufficient to consider only reflex vertices in the triangle containment test. A *reflex vertex* is one for which the interior angle formed by the two edges sharing it is larger than π radians. The data structure for the polygon maintains four doubly linked lists simultaneously using an array for storage rather than dynamically allocating and deallocating memory in a standard list data structure. The vertices of the polygon are stored in a cyclical list, the convex vertices are stored in a linear list, the reflex vertices are stored in a linear list and the ear tips are stored in a cyclical list.

Once the initial lists for reflex vertices and ears are constructed, the ears are removed one at a time. If $\boldsymbol{V}_i$ is an ear that is removed, then the edge configuration at the adjacent vertices $\boldsymbol{V}_{i-1}$ and $\boldsymbol{V}_{i+1}$ can change. If an adjacent vertex is convex, a quick sketch will convince you that it remains convex. If an adjacent vertex is an ear, it does not necessarily remain an ear after $\boldsymbol{V}_i$ is removed. If the adjacent vertex is reflex, it is possible that it becomes convex and possibly an ear. After the removal of $\boldsymbol{V}_i$, if an adjacent vertex is convex you must test whether it is an ear by iterating over over the reflex vertices and testing for containment in the triangle of that vertex. There are $O(n)$ ears. Each update of an adjacent vertex involves testing whether or not it is an ear, a process that is $O(n)$ per update. The total removal process is $O(n^2)$.

The simple polygon of figure 1 is used to illustrate the algorithm. The vertex locations are $\boldsymbol{V}_0 = (3, 48)$, $\boldsymbol{V}_1 = (52, 8)$, $\boldsymbol{V}_2 = (99, 50)$, $\boldsymbol{V}_3 = (138, 25)$, $\boldsymbol{V}_4 = (175, 77)$, $\boldsymbol{V}_5 = (131, 72)$, $\boldsymbol{V}_6 = (111, 113)$, $\boldsymbol{V}_7 = (72, 43)$, $\boldsymbol{V}_8 = (26, 55)$, and $\boldsymbol{V}_9 = (29, 100)$. The initial list of convex vertices is $C = \{0, 1, 3, 4, 6, 9\}$, the initial list of reflex vertices is $R = \{2, 5, 7, 8\}$ and the initial list of ears is $E = \{3, 4, 6, 9\}$. The ear at vertex 3 is removed, so the first triangle in the triangulation is $T_0 = \langle 2, 3, 4 \rangle$. Figure 2 shows the ear-clipped polygon.

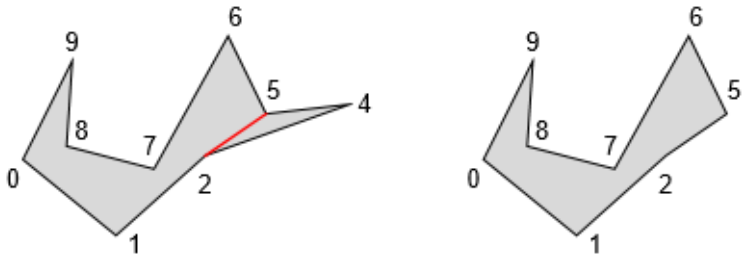
Figure 2. The right polygon shows the ear $\langle 2, 3, 4 \rangle$ clipped from the left polygon.



The adjacent vertex 2 was reflex and still is. The adjacent vertex 4 was an ear and remains so. The reflex list R remains unchanged, but the ear list is now $E = \{4, 6, 9\}$ (removed vertex 3).

The ear at vertex 4 is removed. The next triangle in the triangulation is $T_1 = \langle 2, 4, 5 \rangle$. Figure 3 shows the ear-clipped polygon.

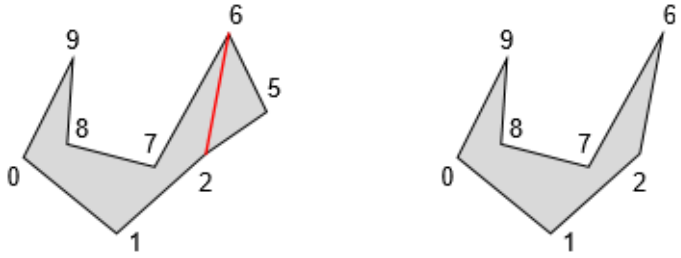
Figure 3. The right polygon shows the ear $\langle 2, 3, 4 \rangle$ clipped from the left polygon.



The adjacent vertex 2 was reflex and still is. The adjacent vertex 5 was reflex but is now convex. It is tested and found to be an ear. The vertex is removed from the reflex list, $R = \{2, 7, 8\}$. It is also added to the ear list, $E = \{5, 6, 9\}$ (added vertex 5, removed vertex 4).

The ear at vertex 5 is removed. The next triangle in the triangulation is $T_2 = \langle 2, 5, 6 \rangle$. Figure 4 shows the ear-clipped polygon.

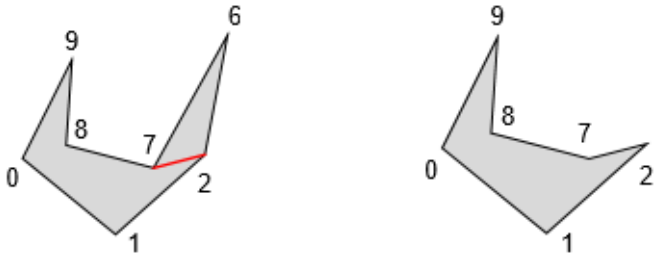
Figure 4. The right polygon shows the ear $\langle 2, 5, 6 \rangle$ clipped from the left polygon.



The adjacent vertex 2 was reflex but is now convex. Although difficult to determine from the figure, the vertex 7 is inside the triangle $\langle 1, 2, 6 \rangle$, so vertex 2 is not an ear. The adjacent vertex 6 was an ear and remains so. The new reflex list is $R = \{7, 8\}$ (removed vertex 2) and the new ear list is $E = \{6, 9\}$ (removed vertex 5).

The ear at vertex 6 is removed. The next triangle in the triangulation is $T_3 = \langle 2, 6, 7 \rangle$. Figure 5 shows the clipped polygon.

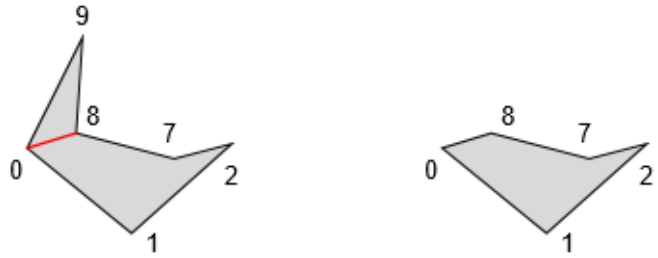
Figure 5. The right polygon shows the ear $\langle 2, 6, 7\rangle$ clipped from the left polygon.



The adjacent vertex 2 was convex and remains so. It was not an ear, but now it becomes one. The adjacent vertex 7 was reflex and remains so. The reflect list R does not change, but the new ear list is $E = \{9, 2\}$ (added vertex 2, removed vertex 6). The ear list is written this way because the new ear is added first before the old ear is removed. Before removing the old ear, it is still considered to be first in the list. The remove operation sets the first item to be that next value to the old ear rather than the previous value.

The ear at vertex 9 is removed. The next triangle in the triangulation is $T_4 = \langle 8, 9, 0\rangle$. Figure 6 shows the clipped polygon.

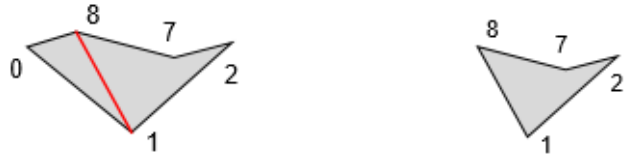
Figure 6. The right polygon shows the ear $\langle 8, 9, 0\rangle$ clipped from the left polygon.



The adjacent vertex 8 was reflex but is now convex and in fact an ear. The adjacent vertex 0 was convex and remains so. It was not an ear but has become one. The new reflex list is $R = \{7\}$ and the new ear list is $E = \{0, 2, 8\}$ (added 8, added 0, removed 9, the order shown is what the program produces).

The ear at vertex 0 is removed. The next triangle in the triangulation is $T_5 = \langle 8, 0, 1\rangle$. Figure 7 shows the clipped polygon.

Figure 7. The right polygon shows the ear $\langle 8, 0, 1\rangle$ clipped from the left polygon.



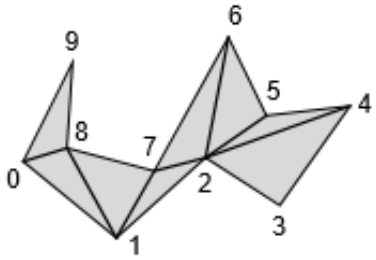
Both adjacent vertices 8 and 1 were convex and remain so. Vertex 8 remains an ear and vertex 1 remains not an ear. The reflex list remains unchanged. The new ear list is $E = \{2, 8\}$ (removed vertex 0).

Finally, the ear at vertex 2 is removed. The next triangle in the triangulation is $T_6 = \langle 1, 2, 7 \rangle$. Figure 8 shows the clipped polygon.

Figure 8. The right polygon shows the ear $\langle 1, 2, 7 \rangle$ clipped from the left polygon.

At this time there is no need to update the reflect or ear lists because we detect that only three vertices remain. The last triangle in the triangulation is $T_7 = \langle 7, 8, 1 \rangle$. The full triangulation is show in Figure 9.

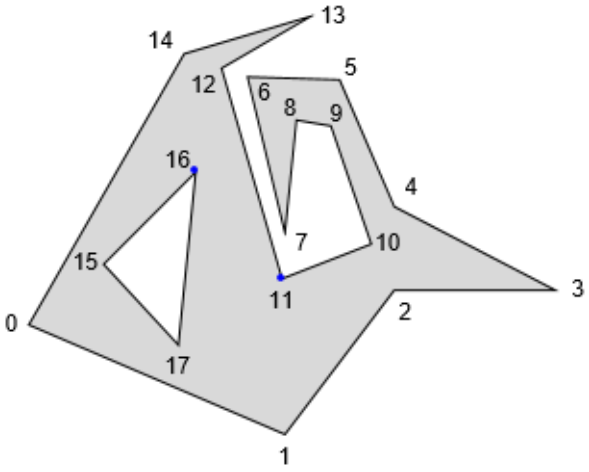
Figure 9. The full triangulation of the original polygon.



3 Polygons with a Hole

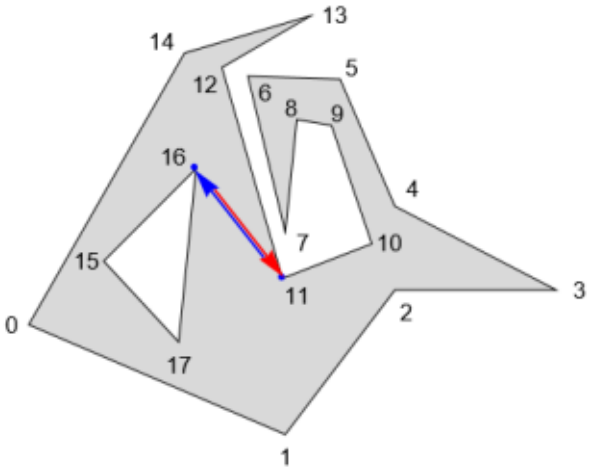
The ear-clipping algorithm may also be applied to polygons with holes. First, consider a polygon with one hole as shown in figure 10. It consists of an *outer polygon* and an *inner polygon*. The ordering of the outer vertices and the inner vertices must be opposite. If the outer vertices are counterclockwise ordered, then the inner vertices must be clockwise ordered.

Figure 10. A polygon with a hole.



The vertices $\boldsymbol{V}_{11}$ and $\boldsymbol{V}_{16}$ are mutually visible, colored blue in figure 10. We can convert this to the topology of a simple polygon by introducing two coincident but directed edges connecting the blue-colored vertices: $\langle \boldsymbol{V}_{11}, \boldsymbol{V}_{16} \rangle$ and $\langle \boldsymbol{V}_{16}, \boldsymbol{V}_{11} \rangle$. I will refer to this pair as a *bridge* between the outer polygon and inner polygon. Figure 11 shows the two new edges, one drawn in blue and one drawn next to it in red.

Figure 11. The polygon hole is removed by introducing two coincident but directed edges that "cut" open the polygon. Small half-arrows are attached to the edges to show their directions. The bridge has outer-polygon vertex $\boldsymbol{V}_{11}$ and inner-polygon vertex $\boldsymbol{V}_{16}$.



The original outer polygon has vertices

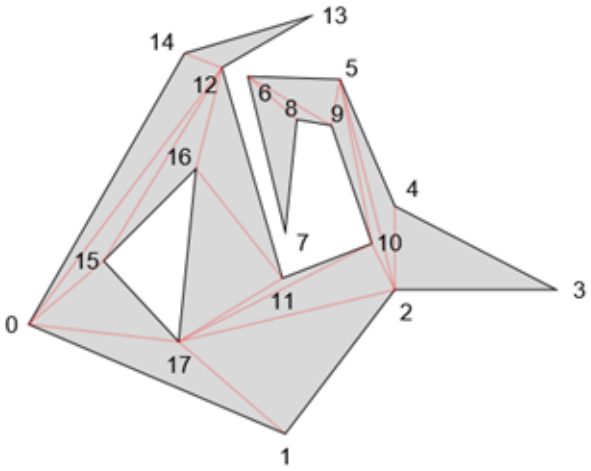
$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\} \tag{1}$$

and the original inner polygon has vertices $\{15, 16, 17\}$. After inserting the bridge, the new outer polygon is

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 16, 17, 15, 16, 11, 12, 13, 14\} \tag{2}$$

The new outer polygon is what I call *pseudosimple* to emphasize that two of the edges are coincident but are in opposite directions. This polygon can be triangulated by ear clipping, shown in figure 12.

Figure 12. The triangulation of the polygon with a hole shown in figure 10.



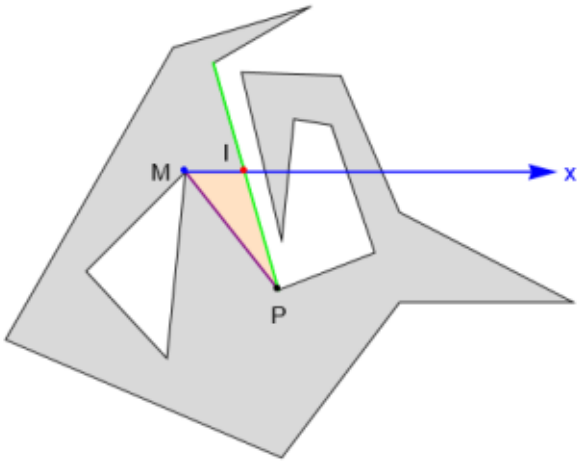
4 Finding Mutually Visible Vertices

Visually, we can see in figure 10 that vertices V_{11} and V_{16} are mutually visible. In fact, there are more such pairs, one vertex from the outer polygon and one vertex from the inner polygon. We need an algorithm that will find a pair of mutually visible vertices.

One algorithm is the following. Search the vertices of the inner polygon to find the one with maximum x-value. In figure 10, this is V_{16}. Imagine an observer standing at this vertex, looking in the positive x-direction. They will see (1) an interior point of an edge or (2) a vertex, an interior point of an edge being the most probable candidate. If a vertex is visible, then we have a mutually visible pair.

Suppose that the closest visible point in the positive x-direction is an interior edge point, as illustrated in figure 13.

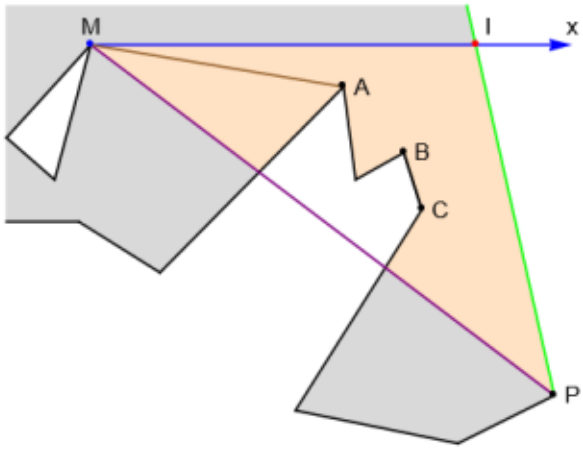
Figure 13. The closest visible point is an interior edge point.



Let $\boldsymbol{M}$ be the origin of the ray, which in the example is $\boldsymbol{V}_{16}$. The ray $\boldsymbol{M} + t(1, 0)$ is shown in blue. The closest visible point is shown in red, call this point $\boldsymbol{I}$. The edge on which the closest point occurs is drawn in green. The endpoint of the edge that has maximum x-value is labeled $\boldsymbol{P}$. The point $\boldsymbol{P}$ is the candidate for mutual visibility with $\boldsymbol{M}$. The line segment connecting them is drawn in purple. The triangle $\langle \boldsymbol{M}, \boldsymbol{I}, \boldsymbol{P} \rangle$ is drawn with an orange interior.

In figure 13, $\boldsymbol{P}$ is indeed visible to $\boldsymbol{M}$. Generally, it is possible that other edges of the outer polygon cross the line segment $\langle \boldsymbol{M}, \boldsymbol{P} \rangle$ in which case $\boldsymbol{P}$ is not visible to $\boldsymbol{M}$. Figure 14 shows such a situation.

Figure 14. A case where $\boldsymbol{P}$ is not visible to $\boldsymbol{M}$.



The gray color denotes the region between the outer and inner polygons. The orange region is also part of

the interior. In figure 13, the entire interior of triangle $\langle \boldsymbol{M}, \boldsymbol{I}, \boldsymbol{P} \rangle$ is part of the outer-inner interior. In figure 14, the outer polygon cuts into the triangle, so only a subset of the triangle interior is part of the outer-inner interior.

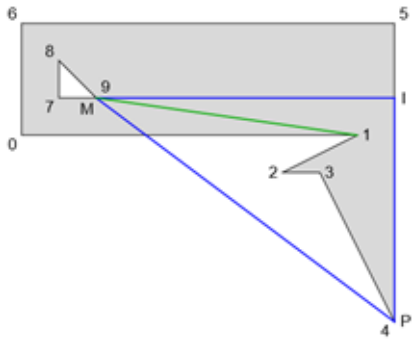
Four vertices of the outer polygon occur inside triangle $\langle \boldsymbol{M}, \boldsymbol{I}, \boldsymbol{P} \rangle$. Generally, if vertices occur inside this triangle, at least one must be reflex. And of all such reflex vertices, one must be visible to $\boldsymbol{M}$. In figure 14, three reflex vertices of the outer polygon occur inside the triangle; they are labeled $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$. The reflex vertex $\boldsymbol{A}$ is visible to $\boldsymbol{M}$, because it minimizes the angle between $\langle \boldsymbol{M}, \boldsymbol{I} \rangle$ and $\langle \boldsymbol{M}, \boldsymbol{R} \rangle$ for all reflex vertices $\boldsymbol{R}$ inside the triangle $\langle \boldsymbol{M}, \boldsymbol{I}, \boldsymbol{P} \rangle$.

The algorithm is summarized as:

1. Search the inner polygon for vertex $\boldsymbol{M}$ of maximum x-value.
2. Intersect the ray $\boldsymbol{M} + t(1, 0)$ with all directed edges $\langle \boldsymbol{V}_i, \boldsymbol{V}_{i+1} \rangle$ of the outer polygon for which $\boldsymbol{M}$ is to the left of the line containing the edge. Moreover, the intersection tests need only involve directed edges for which $\boldsymbol{V}_i$ is below (or on) the ray and $\boldsymbol{V}_{i+1}$ is above (or on) the ray. This is essential when the polygon has multiple holes and one or more holes have already had bridges inserted. Let $\boldsymbol{I}$ be the closest visible point to $\boldsymbol{M}$ on the ray. The implementation keeps track of this by monitoring the t-value in search of the smallest positive value for those edges in the intersection tests.
3. If $\boldsymbol{I}$ is a vertex of the outer polygon, then $\boldsymbol{M}$ and $\boldsymbol{I}$ are mutually visible and the algorithm terminates.
4. Otherwise, $\boldsymbol{I}$ is an interior point of the edge $\langle \boldsymbol{V}_i, \boldsymbol{V}_{i+1} \rangle$. Select $\boldsymbol{P}$ to be the endpoint of maximum x-value for this edge.
5. Search the reflex vertices of the outer polygon, not including $\boldsymbol{P}$ if it happens to be reflex. If all of these vertices are strictly outside triangle $\langle \boldsymbol{M}, \boldsymbol{I}, \boldsymbol{P} \rangle$, then $\boldsymbol{M}$ and $\boldsymbol{P}$ are mutually visible and the algorithm terminates.
6. Otherwise, at least one reflex vertex lies in $\langle \boldsymbol{M}, \boldsymbol{I}, \boldsymbol{P} \rangle$. Search for the reflex $\boldsymbol{R}$ that minimizes the angle between $\langle \boldsymbol{M}, \boldsymbol{I} \rangle$ and $\langle \boldsymbol{M}, \boldsymbol{R} \rangle$; then $\boldsymbol{M}$ and $\boldsymbol{R}$ are mutually visible and the algorithm terminates.

Mark Spoor (17 August 2021) had suggested to use a distance approach rather than a minimum-angle approach, recommending that you choose the reflex vertex closest to $\boldsymbol{M}$. However, Emmett Lalish (24 October 2023) provided a counterexample as shown in figure 15.

Figure 15. A polygon for which the minimum-angle reflex vertex is not the same as the minimum-distance reflex vertex. The MIP-triangle is shown in blue and the bridge is shown in green.

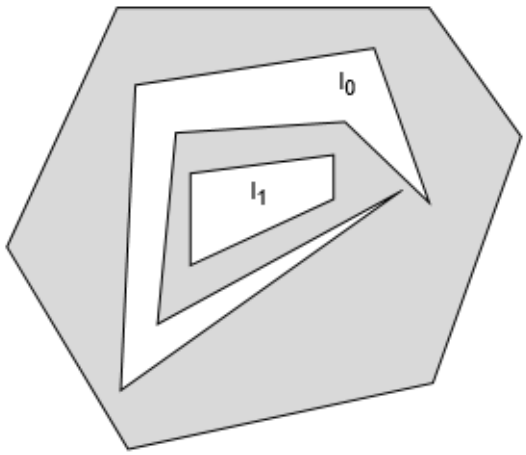


The outer polygon vertices are $\boldsymbol{V}_0 = (0,5)$, $\boldsymbol{V}_1 = (9,5)$, $\boldsymbol{V}_2 = (7,4)$, $\boldsymbol{V}_3 = (8,4)$, $\boldsymbol{V}_4 = (10,0)$, $\boldsymbol{V}_5 = (10,8)$, and $\boldsymbol{V}_6 = (0,8)$. The inner polygon vertices are $\boldsymbol{V}_7 = (1,6)$. $\boldsymbol{V}_8 = (1,7)$. $\boldsymbol{V}_9 = (2,6)$. The various points of interest are $\boldsymbol{M} = \boldsymbol{V}_9$, $\boldsymbol{I} = (10,6)$, and $\boldsymbol{P} = \boldsymbol{V}_4$. The triangle $\langle \boldsymbol{M}, \boldsymbol{I}, \boldsymbol{P} \rangle$ contains the reflex vertices $\boldsymbol{V}_1$ and $\boldsymbol{V}_3$. The minimum-angle reflex vertex is $\boldsymbol{V}_1$ and the minimum-distance reflex vertex is $\boldsymbol{V}_3$. The distance from $\boldsymbol{V}_9$ to $\boldsymbol{V}_1$ is $\sqrt{50} \doteq 7.071068$ and the distance from $\boldsymbol{V}_9$ to $\boldsymbol{V}_3$ is $\sqrt{40} \doteq 6.324555$. The vertices $\boldsymbol{M}$ and $\boldsymbol{V}_3$ are not mutually visible, so the bridge is $\langle \boldsymbol{V}_1, \boldsymbol{V}_9 \rangle$.

5 Polygons with Multiple Holes

A polygon may have multiple holes (inner polygons). The assumptions are that they are all strictly contained by the outer polygon and none of the inner polygons overlap. Figure 16 shows such a polygon.

Figure 16. An outer polygon with two inner polygons.



The figure makes it clear that none of the vertices of inner polygon I_1 are visible to the outer polygon vertices. However, some vertices of inner polygon I_0 are visible to outer polygon vertices. The outer polygon and inner polygon I_0 can be combined into a pseudosimple polygon. This polygon becomes the new outer polygon and I_1 is combined with it to form another pseudosimple polygon. The final pseudosimple polygon is triangulated by ear clipping.

Given multiple inner polygons, the one containing the vertex of maximum x-value of all inner polygon vertices is the one chosen to combine with the outer polygon. The process is then repeated with the new outer polygon and the remaining inner polygons.

The algorithm for combining an outer polygon and a single inner polygon was presented previously with 6 steps. It is important to pay attention to the description of step 2. Intersections of $\boldsymbol{M} + t(1,0)$ are computed only for directed edges $\langle \boldsymbol{V}_i, \boldsymbol{V}_{i+1} \rangle$ such that $\boldsymbol{V}_i$ is below (or on) the ray and $\boldsymbol{V}_{i+1}$ is above (or on) the ray). In particular, if the directed edge is a bridge from a previous combination of the outer polygon with an inner polygon, the pseudosimple polygon has directed edges $\langle \boldsymbol{V}_i, \boldsymbol{V}_{i+1} \rangle$ and $\langle \boldsymbol{V}_{i+1}, \boldsymbol{V}_i \rangle$. If one of these edges satisfies the constraints to be a candidate for intersection, then the other edge cannot be a candidate.

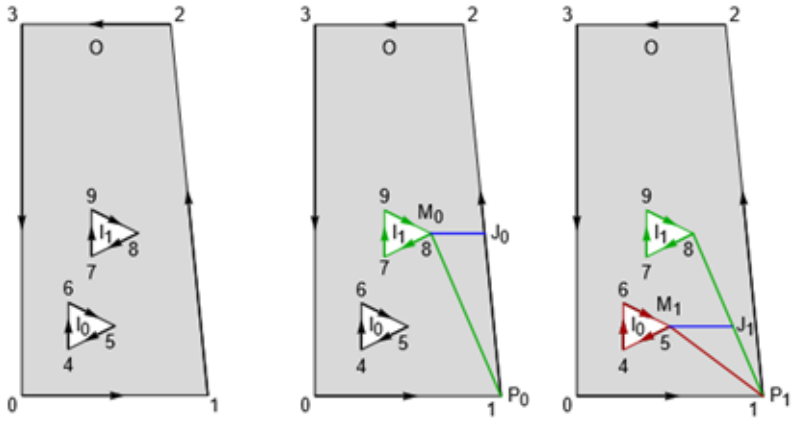
A couple of technical support incidents had been received that indicated a problem with the algorithm when an outer polygon vertex occurs in 2 or more bridges. The first reporter (Tomer Shalev, 29 November 2019) asked whether the bridges must be sorted, but he did not provide an example for which the Geometric Tools code failed. I suspect that he implemented the algorithm from the description in the previous version of this document. Moreover, he mentioned that the implementation is sensitive to small degenerate cases, where two inner polygons touch at a vertex (because of floating-point rounding errors). The Geometric Tools code is designed to allow rational arithmetic for the computations to avoid the rounding errors. A precondition for the input polygons is that they are simple polygons that do not share vertices. Each inner polygon must be strictly inside the outer polygon and two inner polygons cannot overlap. Ensuring these requires preprocessing the vertices and polygons before triangulation. Alternatively, Emmett Lalish has an extension of the ideas of this document that is designed to be used with floating-point arithmetic: https://github.com/elalish/manifold.

The second reporter (Mario Binder, 24 January 2023) said that the bridges must be sorted. An example was provided in the form of sketches rather than actual data sets for which the code failed. He implemented the algorithm himself rather than using the Geometric Tools code. implemented the algorithm themselves rather than using the Geometric Tools code. Perhaps my analysis is incorrect, but I believe the algorithm does not require the explicit sorting of bridges. The following examples are from the second reporter. It is instructive to walk through these examples to see the self-sorting nature of the bridge construction. For simplicity of labeling, only the indices j for vertices $\boldsymbol{V}_j$ are shown. The notation used here is from figure 13. The rays have direction $(1, 0)$.

5.1 Example 1

An outer polygon $O = \{0, 1, 2, 3\}$ and two inner polygons, $I_0 = \{5, 4, 6\}$ and $I_1 = \{8, 7, 9\}$, are shown in figure 17. The vertices are $\boldsymbol{V}_0 = (0, 0)$, $\boldsymbol{V}_1 = (4, 0)$, $\boldsymbol{V}_2 = (3.2, 8)$, $\boldsymbol{V}_3 = (0, 8)$, $\boldsymbol{V}_4 = (1, 1)$, $\boldsymbol{V}_5 = (2, 1.5)$, $\boldsymbol{V}_6 = (1, 2)$, $\boldsymbol{V}_7 = (1.5, 3)$, $\boldsymbol{V}_8 = (2.5, 3.5)$ and $\boldsymbol{V}_9 = (1.5, 4)$.

Figure 17. An example of combining the outer polygon with the inner polygons. The combination of O and I_1 is named $O \oplus I_1$. The combination of $O \oplus I_1$ and I_0 is named $(O \oplus I_1) \oplus I_0$.



(a) original configuration (b) $O \oplus I_1$ (c) $(O \oplus I_1) \oplus I_0$

$\boldsymbol{V}_8$ has the maximum x-value of all the inner polygon vertices. Therefore, inner polygon I_1 is combined with O first. Figure 17(b) shows that the ray origin is $\boldsymbol{M}_0 = \boldsymbol{V}_8$. The ray intersects the directed edge $\langle \boldsymbol{V}_1, \boldsymbol{V}_2 \rangle$ at a point $\boldsymbol{J}_0$. This implies $\boldsymbol{P}_0 = \boldsymbol{V}_1$, which is chosen as the O vertex visible to the I_1 vertex $\boldsymbol{M}_0$. The bridge has directed edges $\langle \boldsymbol{V}_1, \boldsymbol{V}_8 \rangle$ (outer-to-inner) and $\langle \boldsymbol{V}_8, \boldsymbol{V}_1 \rangle$ (inner-to-outer). The new outer polygon is

$$O \oplus I_1 = \{0, 1, 2, 3\} \oplus \{8, 7, 9\} = \{0, 1, 8, 7, 9, 8, 1, 2, 3\} \tag{3}$$

This polygon has 9 elements that are stored in an array. It is important to observe that $\boldsymbol{V}_1$ of the bridge occurs twice in new outer polygon. However, the first occurrence is at the outer-polygon array index 1 and the second occurrence is at the outer-polygon array index 6. The differing outer-polygon array indices lead to a self-sorting algorithm of multiple bridges that share the same outer-polygon vertex.

The inner polygon I_0 is now combined with $O \oplus I_1$. Figure 17(c) shows that the ray origin is $\boldsymbol{M}_1 = \boldsymbol{V}_5$. The ray intersects the directed edge $\langle \boldsymbol{V}_1, \boldsymbol{V}_8 \rangle$. Note that $\boldsymbol{V}_5$ is *to the left of directed edge* $\langle \boldsymbol{V}_1, \boldsymbol{V}_8 \rangle$. The ray intersects the other directed edge $\langle \boldsymbol{V}_8, \boldsymbol{V}_1 \rangle$, but $\boldsymbol{V}_5$ is *to the right of directed edge* $\langle \boldsymbol{V}_8, \boldsymbol{V}_1 \rangle$. Such directed edges are deemed not visible to the ray. An implementation must discard these edges when computing intersections between the ray and the outer polygon edges. [Mr. Binder provided a sketch of this case where the ray was shown to intersect the directed edge $\langle \boldsymbol{V}_1, \boldsymbol{V}_2 \rangle$. But this directed edge is not the closest to $\boldsymbol{M}$ measured along the ray; the directed edge $\langle \boldsymbol{V}_1, \boldsymbol{V}_8 \rangle$ is the closest directed edge measured along the ray.]

The ray intersects the directed edge $\langle \boldsymbol{V}_1, \boldsymbol{V}_8 \rangle$ at a point $\boldsymbol{J}_1$. This implies $\boldsymbol{P}_1 = \boldsymbol{V}_1$, which is chosen as the $O \oplus I_1$ vertex visible to the I_0 vertex $\boldsymbol{M}_1$. The bridge has directed edges $\langle \boldsymbol{V}_1, \boldsymbol{V}_5 \rangle$ (outer-to-inner) and $\langle \boldsymbol{V}_5, \boldsymbol{V}_1 \rangle$ (inner-to-outer), and it occurs *before* bridge $\langle \boldsymbol{V}_1, \boldsymbol{V}_8$ as you traverse around $\boldsymbol{V}_1$ in clockwise order. The new outer polygon is

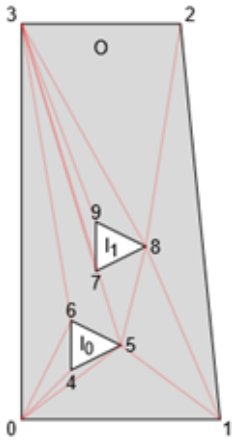
$$(O \oplus I_1) \oplus I_0 = \{0, 1, 8, 7, 9, 8, 1, 2, 3\} \oplus \{5, 4, 6\} = \{0, 1, 5, 4, 6, 5, 1, 8, 7, 9, 10, 11, 2, 3\} \tag{4}$$

This pseudosimple polygon is then triangulated using the ear clipping algorithm to produce triangles involving all $\boldsymbol{V}_j$ for $0 \le j \le 13$. The triangles are

$$\langle 0, 1, 5 \rangle,\ \langle 0, 4, 6 \rangle,\ \langle 0, 5, 4 \rangle,\ \langle 0, 6, 3 \rangle,\ \langle 1, 2, 8 \rangle,\ \langle 1, 8, 5 \rangle,\ \langle 2, 3, 8 \rangle,\ \langle 3, 6, 5 \rangle,\ \langle 3, 5, 7 \rangle,\ \langle 3, 7, 9 \rangle,\ \langle 3, 9, 8 \rangle,\ \langle 5, 8, 7 \rangle \tag{5}$$

Figure 18 shows the triangulation.

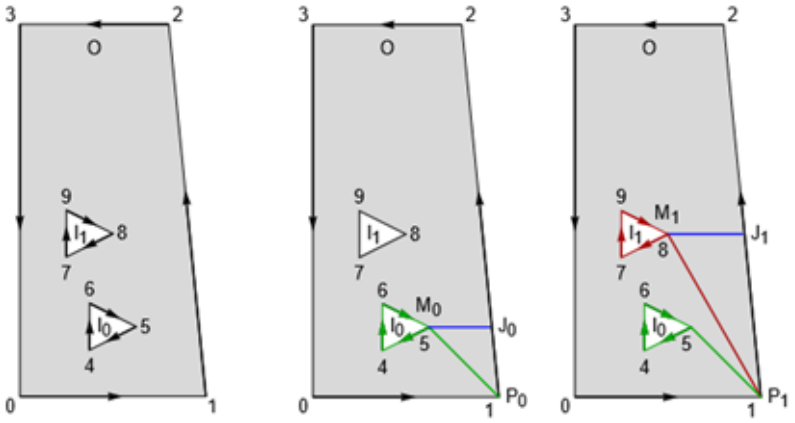
Figure 18. The triangulation of the pseudosimple polygon $(O \oplus I_1) \oplus I_0$.



5.2 Example 2

An outer polygon $O = \{0, 1, 2, 3\}$ and two inner polygons, $I_0 = \{5, 4, 6\}$ and $I_1 = \{8, 7, 9\}$, are shown in figure 19. The vertices are $\boldsymbol{V}_0 = (0, 0)$, $\boldsymbol{V}_1 = (4, 0)$, $\boldsymbol{V}_2 = (3, 10)$, $\boldsymbol{V}_3 = (0, 10)$, $\boldsymbol{V}_4 = (1.5, 1)$, $\boldsymbol{V}_5 = (2.5, 1.5)$, $\boldsymbol{V}_6 = (1.5, 2)$, $\boldsymbol{V}_7 = (1, 3)$, $\boldsymbol{V}_8 = (2, 3.5)$ and $\boldsymbol{V}_9 = (1, 4)$.

Figure 19. An example of combining the outer polygon with the inner polygons. The combination of O and I_0 is named $O \oplus I_0$. The combination of $O \oplus I_0$ and I_1 is named $(O \oplus I_0) \oplus I_1$.



(a) original configuration (b) $O \oplus I_0$ (c) $(O \oplus I_0) \oplus I_1$

$\boldsymbol{V}_5$ has the maximum x-value of all the inner polygon vertices. Therefore, inner polygon I_0 is combined with O first. Figure 19(b) shows that the ray origin is $\boldsymbol{M}_0 = \boldsymbol{V}_5$. The ray intersects the directed edge $\langle \boldsymbol{V}_1, \boldsymbol{V}_2 \rangle$ at a point $\boldsymbol{J}_0$. This implies $\boldsymbol{P}_0 = \boldsymbol{V}_1$, which is chosen as the O vertex visible to the I_0 vertex $\boldsymbol{M}_0$. The bridge has directed edges $\langle \boldsymbol{V}_1, \boldsymbol{V}_5 \rangle$ (outer-to-inner) and $\langle \boldsymbol{V}_5, \boldsymbol{V}_1 \rangle$ (inner-to-outer). The new outer polygon is

$$O \oplus I_0 = \{0, 1, 2, 3\} \oplus \{5, 4, 6\} = \{0, 1, 5, 4, 6, 5, 1, 2, 3\} \tag{6}$$

This polygon has 9 elements that are stored in an array. It is important to observe that $\boldsymbol{V}_1$ of the bridge occurs twice in new outer polygon. However, the first occurrence is at the outer-polygon array index 1 and the second occurrence is at the outer-polygon array index 6. The differing outer-polygon array indices lead to a self-sorting algorithm of multiple bridges that share the same outer-polygon vertex.

The inner polygon I_1 is now combined with $O \oplus I_0$. Figure 17c shows that the ray origin is $\boldsymbol{M}_1 = \boldsymbol{V}_8$. The ray intersects the directed edge $\langle \boldsymbol{V}_1, \boldsymbol{V}_2 \rangle$ at a point $\boldsymbol{J}_1$. This implies $\boldsymbol{P}_1 = \boldsymbol{V}_1$, which is chosen as $O \oplus I_0$ vertex visible to the I_1 vertex vertex $\boldsymbol{M}_1$. The bridge is $\langle \boldsymbol{V}_1, \boldsymbol{V}_8 \rangle$. The self-sorting nature of the algorithm (based on the outer-polygon array indices) ensures that the bridge $\langle \boldsymbol{V}_1, \boldsymbol{V}_5 \rangle$ occurs before the bridge $\langle \boldsymbol{V}_1, \boldsymbol{V}_8 \rangle$ in the traversal of the outer-polygon array indices. The new outer polygon is

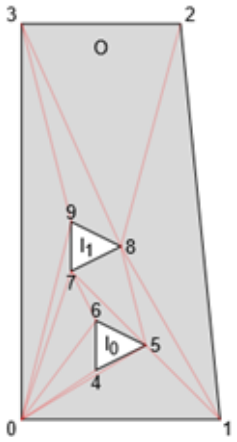
$$(O \oplus I_0) \oplus I_1 = \{0, 1, 5, 4, 6, 5, 1, 2, 3\} \oplus \{8, 7, 9\} = \{0, 1, 5, 4, 6, 5, 1, 8, 7, 9, 8, 1, 2, 3\} \tag{7}$$

This pseudosimple polygon is then triangulated using the ear clipping algorithm to produce triangles involving all $\boldsymbol{V}_j$ for $0 \le j \le 13$. The triangles are

$$\langle 0, 1, 5 \rangle,\ \langle 0, 4, 6 \rangle,\ \langle 0, 5, 4 \rangle,\ \langle 0, 6, 7 \rangle,\ \langle 0, 7, 9 \rangle,\ \langle 0, 9, 3 \rangle,\ \langle 1, 2, 8 \rangle,\ \langle 1, 8, 5 \rangle,\ \langle 2, 3, 8 \rangle,\ \langle 3, 9, 8 \rangle,\ \langle 5, 7, 6 \rangle,\ \langle 5, 8, 7 \rangle \tag{8}$$

Figure 20 shows the triangulation.

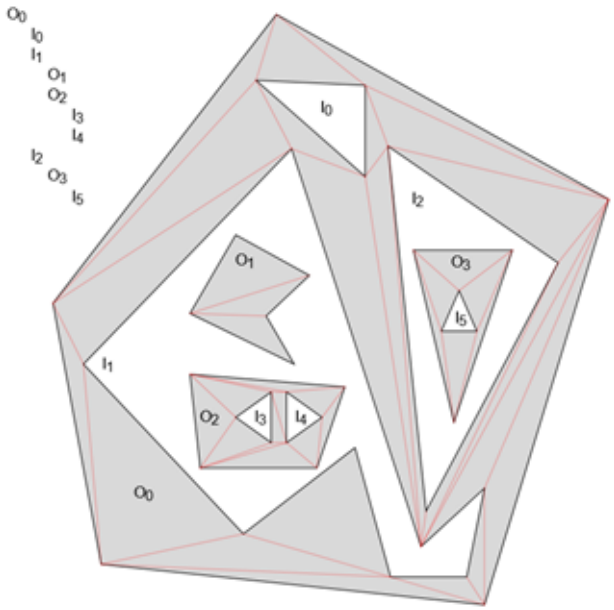
Figure 20. The triangulation of the pseudosimple polygon $(O \oplus I_0) \oplus I_1$.



6 Hierarchies of Polygons

The inner polygons themselves can contain outer polygons with holes, leading to trees of nested polygons. Figure 21 shows a tree of nested polygons that has been triangulated using ear clipping.

Figure 21. A triangulated tree of nested polygons. The upper-left portion of the image shows the tree data structure with child nodes indented to the right from their parent nodes. Siblings are aligned vertically.



The root of the tree corresponds to the outermost outer polygon. The children of the root are the inner polygons contained in this outer polygon. Each grandchild (if any) is a subtree whose root corresponds to an outer polygon that is strictly contained in the parent inner polygon and whose own children are inner polygons. A tree of polygons may be processed using a breadth-first traversal.

The pseudocode for processing the tree is shown in Listing 1.

Listing 1. Pseudocode for processing a tree of nested polygons.

```
struct PolygonTree
{
    Polygon p;
    array<PolygonTree> children;
};

array<Triangle> triangles;
PolygonTree tree = <some tree of polygons>;
queue<PolygonTree> Q;
Q.insertRear(tree);
while (not Q.empty()) do
{
    PolygonTree outerNode = Q.removeFront();
    numChildren = outerNode.children.quantity();
    if (numChildren == 0)
    {
        // The outer polygon is a simple polygon with no nested inner polygons.
        triangles.append(GetEarClipTriangles(outerNode.p));
    }
    else
    {
        // The outer polygon contains inner polygons.
        for (i = 0; i < numChildren; i++)
        {
            PolygonTree innerNode = outerNode.children[i];
            array<Polygon> innerPolygons;
            numGrandchildren = innerNode.children.quantity();
            for (j = 0; j < numGrandchildren; j++)
            {
                innerPolygons.append(innerNode.p);
                Q.insertFront(innerNode.children[j]);
            }
        }

        Polygon combined = CombineToPseudoSimple(outerNode.p,innerPolygons);
        triangles.append(GetEarClipTriangles(combined));
    }
}
```

The function CombineToPseudoSimple encapsulates the algorithm described previously for finding two mutually visible vertices for an outer and an inner polygon

References

[1] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6:485–524, 1991.

[2] B. Chazelle and J. Incerpi. Triangulation and shape complexity. *ACM Trans. on Graphics*, 3:135–152, 1984.

[3] A. Fournier and D.Y. Montuno. Triangulating simple polygons and equivalent problems. *ACM Trans. on Graphics*, 3:153–174, 1984.

[4] G.H. Meisters. Polygons have ears. *Amer. Math. Monthly*, 82:648–651, 1975.

[5] R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Computational Geometry: Theory and Applications*, 1(1):51–64, 1991.

[6] Wolfram Research, Inc. *Mathematica 13.2.1.* Wolfram Research, Inc., Champaign, Illinois, 2023.