INFO-F420: Project report
Compatible triangulations of polygons

Laurie Van Bogaert

14 December 2020

1 Introduction

For the course of computational geometry, students are asked to do a project. The present document is the final project report. The subject that I choose is: **Compatible triangulations of polygons**.

2 Why this problem ?

The thematic for the projects this year is about the open problems in computational geometry. One of them is about compatible triangulations. On the website of the open project the problem is stated as determining if two sets of points of the same size and in the general position possess compatible triangulations[1]. The project here is about a variant of this problem where the triangulation is done on simple polygons in place of sets of points.

Finding the compatible triangulations has practical application in 2D animations and shape morphing [2]. Therefore, this subject can be used to give entertaining visual results.

3 Theoretical considerations

First it could be interesting to recall some definitions:

- **Simple polygons**: is a single closed area delimited by a polygon chain with no edges intersecting others. Therefore, it can not contain any holes.
- **Triangulation**: is the subdivision of a polygon into a maximal set of triangles that does not intersect each others. Every simple polygon with n vertices possess at least one triangulation and it is always composed of $n-2$ triangles. [3]
- **Compatible triangulations**: The definition given by Aichholzer et al. is that two triangulation T_1 and T_2, over a set of points S_1 and S_2 respectively, are compatible if and only if there exists a bijection φ between the two sets points (S_1 and S_2) and that for every triangle ijk from T_1, that do not contain any other points of S_1, a triangle $\varphi(i)\varphi(j)\varphi(k)$ exists in T_2 and does not contains any point of S_2. A property of compatible triangulation is that the external face is conserved.
- **Compatible triangulation of polygons**: The definition can be described nearly as the same than above except that the two triangulations are done on two Polygon P_1 and P_2. It can be proven that two polygons don't always admit a compatible triangulation. However, a compatible triangulation can always be found using $\Theta(n^2)$ additional points (called Steiner points) inside of each polygon.

4 Implementation

The project is divided in 3 parts.

4.1 User Input

First, the user have to create a polygon by selecting new points with his mouse. Then he have to press the next button to close the polygon. At this points, the program check if there is any edge crossing. Also, the rest of the code suppose that the polygon follow the left-turn convention in computational geometry. Therefore , if the program detect it to be right turn it will make it left turn by inverting the points and the arrays. Next, to create the second polygons, the points of the first polygons are moved. Once again the program check that there is no edge-crossing. Since the second polygon is build by moving the point of the first polygon, it gives a first correspondence between the points of the two polygons.

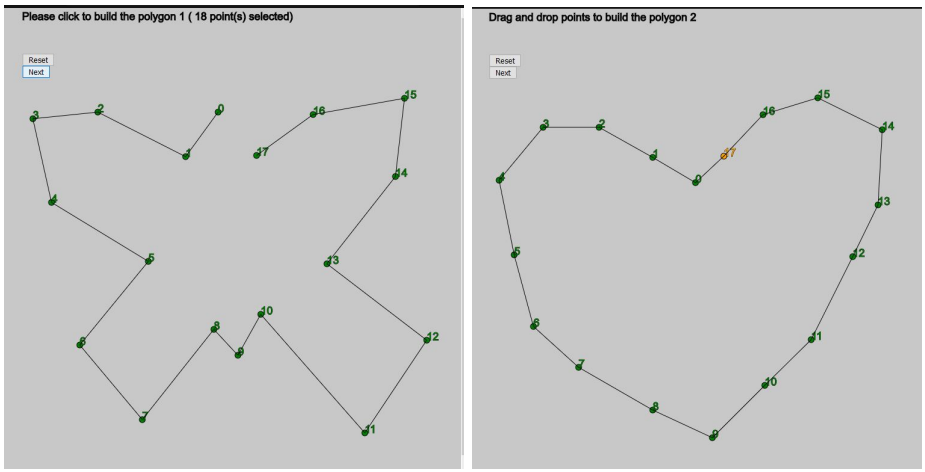

Figure 1: An example of input given by the user. With the first polygon at the left and the second one at the right

4.2 Algorithm for compatible triangulations

First, the number of possible triangulations for a convex polygon of size n is equal to the (n-2)th Catalan number: $C_{n-2} = \frac{(2n-4)!}{(n-1)!(n-2)!}, \qquad n \geq 3.$ [5]. For non-convex simple polygons, the number of combinations is smaller, as there are vertices that are not visible by others.

The polygons may be triangulated without using Steiner point by using the following brute force algorithm (see pseudo-code algorithm 1 in the next page). This recursive algorithm will research compatible triangulations between the first polygon and the second polygon.

The idea here is to explore the tree of possible enumerations of triangles to form a triangulation. It is know that a polygon of size n is composed of exactly $n-2$ triangles [3]. So there is $(n-2)!$ different enumerations of triangles. The number $(n-2)!$ is actually far above the number of possible triangulations for a convex polygon of size n. Is this due to the fact that a triangulation is a set of triangles, so we didn't care about order unlike a enumeration. Therefore, in the worst case scenario where we have to explore the entire tree, the complexity is quite bad and follow a factorial factor.

In the algorithm 1, a triangle abc and is considered as valid if it:

- is left-turn in the first polygon. The polygons are supposed to be left turn so a triangle that contains an edge of the polygon is left turn if the triangle is in the polygon.
- is left-turn in the second polygon
- does not contains a vertex of the first polygon.
- does not contains a vertex of the second polygon.
- does not cross an edge of the first polygon or the valid triangles already computed.
- does not cross an edge of the second polygon or the valid triangles already computed.

Algorithm 1 A brute force algorithm

```
1: procedure findCompatibleTriangulations(triangles , Polygon)
2:     Edge (a,b) ← Polygon.getEdge(0)
3:     for Vertex c in Polygon with c ≠ a and c ≠ b do
4:         if Triangle abc is valid for Polygon1 and Polygon2 then
5:             NewTriangles ← copy(Triangles)
6:             add Triangle abc to NewTriangles
7:             NewPolygon ← copy(Polygon)
8:             for Edge ij in triangle abc do
9:                 if Edge ij is an edge of Polygon then
10:                    remove ij from NewPolygon
11:                else
12:                    add ij to NewPolygon
13:            if NewPolygon still have edges then
14:                NewTriangles ← findCompatibleTriangulations(NewTriangles,NewPolygon)
15:                if NewTriangles ≠ null then
16:                    return NewTriangles
17:            else
18:                return NewTriangles
19:    return null
```

If the procedure above does not gives a valid result, it can be repeated using another point correspondences between the two sets of points S_1 and S_2 at most $n-1$ times. The vertex $x_i \in S_1$ will be linked to the the vertex $\begin{cases} x_{i+t} \in S_2 & \text{if} \quad i+t < n \\ x_{i+t-n} \in S_2 & \text{if} \quad i+t \geq n \end{cases}$ with t being the t^{th} try.

However, it would not work for every pair of polygon as it is known that not every pair of polygons possess compatible triangulations [6]. Nonetheless, it should deal with the simplest cases.

4.3 Showing result

Finally, the compatible triangulation is showed on the screen with a morphing animation between the two polygons. The triangles are traced with red lines and coloured like in the paper [2].

An example of it can been seen at figure 2 where the first and last images correspond to the first and the second polygon and the other pictures are intermediary images taken from the animation.

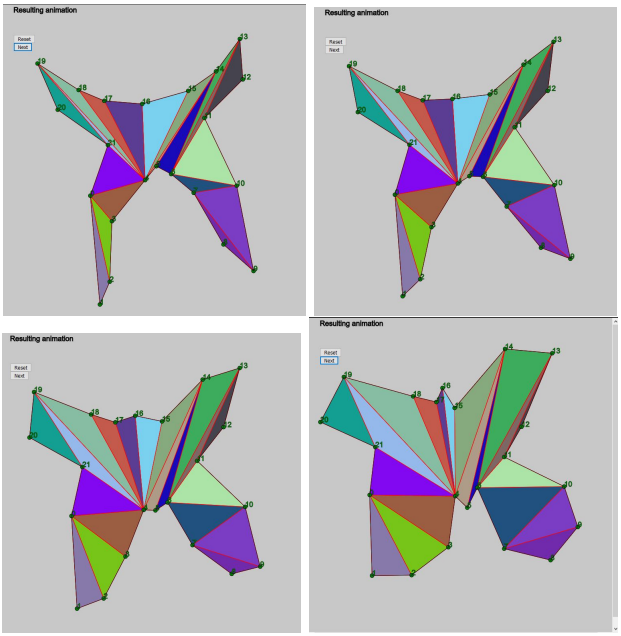

Figure 2: Animation generated with the program when a compatible triangulation is found

5 Discussion and improvements

The program seems to work well on simple cases but the algorithm is far from optimised. For example, it will be more effective to use a plane-sweep algorithm to check for intersections. However, the goal here is to experiment with the problematic so an optimised algorithm is not needed.

Moreover, there is room for improvements. For example, Steiner points can be used to find compatible triangulations for the cases where the program can not find one. In fact, Aronov et al. presents different constructions to achieve it [6].

Furthermore, real applications of compatible triangulations often care for the quality of the triangulation and try to avoid elongated triangles. This element was not considered in the present approach.

References

[1] E. D. Demaine, J. S. B. Mitchell, and J. O'Rourke. (2017) Problem 38: Compatible triangulations. [Online]. Available: http://www.science.smith.edu/~jorourke/TOPP/P38.html#Problem.38

[2] Z. Liu, L. Zhou, H. Leung, and H. P. Shum, "High-quality compatible triangulations and their application in interactive animation," *Computers & Graphics*, vol. 76, pp. 60 – 72, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0097849318301018

[3] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, "Polygon triangulation," in *Computational Geometry: Algorithms and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 45–61. [Online]. Available: https://doi.org/10.1007/978-3-540-77974-2_3

[4] O. Aichholzer, F. Aurenhammer, F. Hurtado, and H. Krasser, "Towards compatible triangulations," *Theoretical Computer Science*, vol. 296, no. 1, pp. 3 – 13, 2003, computing and Combinatorics. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397502004280

[5] P. S. Stanimirović, P. V. Krtolica, M. H. Saračević, and S. H. Mašović, "Decomposition of catalan numbers and convex polygon triangulations." *International Journal of Computer Mathematics*, vol. 91, no. 6, pp. 1315 – 1328, 2014. [Online]. Available: http://search.ebscohost.com.ezproxy.ulb.ac.be/login.aspx?direct=true&db=aph&AN=97586786&site=ehost-live

[6] B. Aronov, R. Seidel, and D. Souvaine, "On compatible triangulations of simple polygons," *Computational Geometry*, vol. 3, no. 1, pp. 27 – 35, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0925772193900285