Advances in Engineering Software 64 (2013) 11–19

Contents lists available at SciVerse ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

A simple algorithm for Boolean operations on polygons

Francisco Martínez *, Carlos Ogayar, Juan R. Jiménez, Antonio J. Rueda

Departamento de Informática, Universidad de Jaén, Campus Las Lagunillas, s/n, 23071 Jaén, Spain

ARTICLE INFO

Article history:
Received 8 June 2012
Received in revised form 21 January 2013
Accepted 14 April 2013
Available online 7 June 2013

Keywords:
Boolean operations polygons
Polygon clipping
Polygon overlay
Computational geometry
Computer graphics
Geometric operations

ABSTRACT

In this paper a simple and efficient algorithm for computing Boolean operations on polygons is presented. The algorithm works with almost any kind of input polygons: concave polygons, polygons with holes, several contours and self-intersecting edges. Important topological information, as the holes of the result polygon, is computed.

1. Introduction

Boolean operations on polygons play an important role in different applied fields such as Computer Graphics, GIS or CAD.

Many algorithms have been developed for polygon clipping, in which several polygons are clipped against a clipping polygon. However, these algorithms often impose strong restrictions on the clipping polygon. For example, some algorithms only work with convex [1,2] or rectangular [3] clipping polygons.

For the general case of Boolean operations on polygons, i.e., concave polygons with holes, several contours and self-intersections less solutions are available. Greiner and Hormann propose a simple and elegant algorithm [4], but it does not properly deal with degenerate cases. However, the algorithm has been extended to deal with degenerate cases [5] and even to cope with holes and polygons with several contours [6]. Unfortunately, this last algorithm is not so elegant as the original one.

Rivero and Feito [7] and Peng et al. [8] present simple methods, from a mathematical point of view, for computing Boolean operations on polygons that are based on the simplex theory proposed by Feito [9]. Unfortunately, these methods are difficult to implement, inefficient and produce a result polygon with almost null topological information—just a list of unconnected edges.

From the field of Computational Geometry some algorithms have been proposed for the more general problem of the overlay of two subdivisions of the plane [10,11]. These algorithms extend the one by Bentley and Ottmann [12] for computing the intersection points in a set of line segments using the plane sweep technique. They compute the overlay in $O((n+k)\log n)$ time, where n denotes the total number of edges of the input overlays and k is the number of intersections between their edges. The algorithm described in [10] does not deal with some degenerate cases, as it is the case of input polygons with overlapping edges. Vatti [13] has also presented a, less efficient algorithm, based on the plane sweep paradigm. In [14] the sweep-line technique is used to compute a data structure based on trapezoidal maps that allows to clip polygons, a drawback of this approach is the $O(n^2)$ size of the data structure.

In [15] we have also extended the algorithm by Bentley and Ottmann [12], but only for computing Boolean operations on polygons in $O((n+k)\log n)$ time. Because we solve a simpler problem our algorithm is also quite simpler and easier to understand than [10,11]. However, the algorithm does not compute which of the result polygon contours are holes, this information is neither computed by the other algorithms for computing Boolean operations [4,7,8,13]. Owing to this information is important in some applications—for instance, it is needed for computing a polygon area—, in this paper we present a modified algorithm that computes the holes of the result polygon. Furthermore, the new algorithm uses a simpler criterion for selecting and joining the edges belonging to the result polygon.

The paper is structured as follows. In Section 2 the algorithm and the format of the result polygon are sketched. Sections 3–5 give a detailed description of the algorithm. Section 6 analyzes its running time and Section 7 explains how the degenerate and special cases are dealt with. Section 8 explains some optimizations,

* Corresponding author. Tel.: +34 953212887.
E-mail address: fmartin@ujaen.es (F. Martínez).

0965-9978/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.advengsoft.2013.04.004

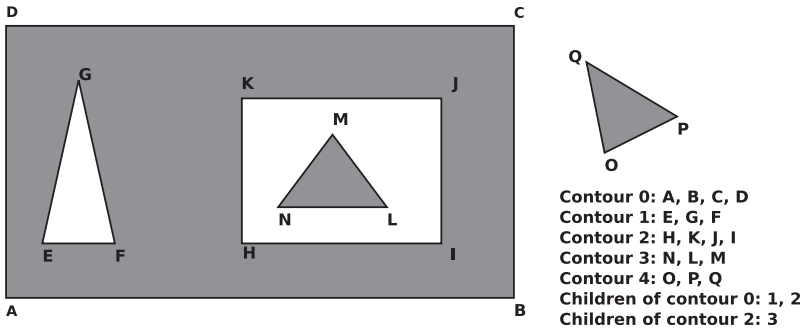

Fig. 1. Polygon specification.

in Section 9 an experimental comparison with Vatti's algorithm is made. Finally, Section 10 draws some conclusions.

2. Foundations

In this section the algorithm for computing Boolean operations on polygons is outlined. First, the format of the result polygon is explained.

2.1. Polygon specification

Our algorithm works with polygons that consist of several contours. Each contour is a simple polygon and the edges of the contours are interior disjoint. Next, we make the following definitions:

- *External contour*: it is a contour not included in any of the other polygon contours.
- *Internal contour*: it is a contour included in at least one of the other polygon contours.
- *Parent contour*. Given an internal contour C, let P be the contour equals to the intersection of all the polygon contours that contain C—i.e., the smaller contour that contains C. Then, we say that P is the *parent contour of* C and that C is a *child contour of* P

A polygon consisting of several contours can be represented as follows:

- The vertices of the contours included in an even number of contours are listed in counter-clockwise order.
- The vertices of the contours included in an odd number of contours are listed in clockwise order.
- For every parent contour its children contours are listed.

For example, Fig. 1 shows a polygon represented this way. Given this representation the area of a polygon can be computed as the sum of the signed areas of its contours. Note that the amount of storage required by this representation is linear in the number of vertices of the polygon.

The result polygon computed by our algorithm follows this representation. However, our algorithm does not impose constraints about the orientation of the vertices of the input polygons. The contours of the input polygons can be listed clockwise or counter-clockwise, regardless they are included in an odd or even number of contours. It is neither necessary to specify child contours. The only restriction imposed by our algorithm on an input polygon is that two edges of the same polygon cannot overlap.

2.2. Outline of the algorithm

The boundary of the result of a Boolean operation on two polygons consists of those portions of the boundary of each polygon that lie inside—or outside, depending on the kind of operation—the other polygon, see Fig. 2. For example, the intersection consists of those portions lying inside, whereas the union consists of those portions lying outside.

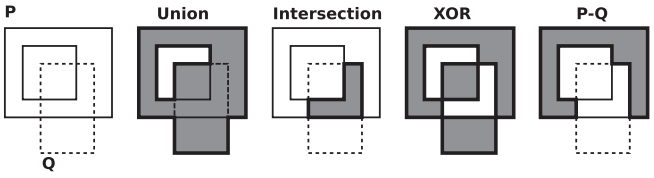

Fig. 2. Boolean operations on polygons.

We propose the following scheme to compute a Boolean operation between two polygons:

1. Subdivide the edges of the polygons at their intersection points.
2. Select those subdivided edges that lie inside—or outside—the other polygon.
3. Join the selected edges to form the contours of the result polygon and compute the child contours.

The algorithm is based on the following idea: after subdividing polygons edges at their intersection points—see Fig. 3—, a subdivided edge lies inside or outside the other polygon and therefore it belongs or not to the result polygon.

In the next sections the algorithm, which has two stages, is described. In the first stage the polygon edges are subdivided and the edges in the result polygon are selected. In the second stage the selected edges are joined to form the contours of the result polygon and the child contours are computed.

2.3. Computing the child contours

In this subsection we describe the approach used to compute the child contours. Given a contour C and its bottom left vertex v we shoot a vertical ray from v that goes downward. Let $e \in C_2$ be the first edge of the polygon crossed by the ray. e represents an

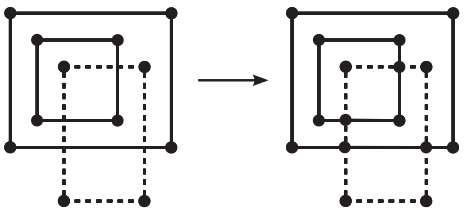

Fig. 3. Subdividing polygon edges at their intersection points.