Algorithmics and C Programming
Project Summary
Fall 2014
LO27

2D Polygon Library

Nicolas Gaud

24 octobre 2014

1 Implementation and evaluation of the project

— Group of 1 or 2 persons at most.
— Implementation in C language.
— The project will be documented in a written report that outlines (15 to 20 pages) used data structures, the selected algorithms, optimizations and encountered difficulties. No copy-paste of entire pages of source code.
— The source codes will be commented and the names of various authors of the project have to be detailed at the beginning of each source file as well as a textual description specifying its purpose.
— The program will have a minimal GUI console. The GUI is not the core of the project, it should nevertheless be simple and user friendly.
— The report must be in **PDF** format and the source code have to be delivered no later than **January** 5^{th} **2015 at 6:00pm** by email addressed to `nicolas.gaud@utbm.fr`, (Subject : LO27 Project - Group : STUDENTNAME1UPPERCASE and STUDENTNAME2UPPERCASE) in a ZIP or TAR.GZ archive named in the following way :

LO27_STUDENTNAME1UPPERCASE_STUDENTNAME2UPPERCASE.zip

Any delay or failure to comply with these guidelines will be penalized.

A special attention should be paid on the following points when writing the program : the program runs smoothly without bugs (it's better not to provide a feature rather than provide it if it does not work, negative points), readability and clarity of the code (comments and indentation), complexity and efficiency of proposed algorithms, choice of data structures, modularity of the code (development of small functions, distributed in various files comprising functions consistently and appropriately named).

2 Project goal

The objective of this project is to provide a library of functions for manipulating 2D-polygons. You must provide a program that allows a user to interactively test every single functions provided by this library.

3 Work to achieve

3.1 Library

A library written in C containing the types and functions described below.

The library will be implemented using at least two C files (header and source). The final archive will at least contains in a dedicated directory the following files :

polygon.h the header file that contains the prototype of the functions, types, constants and variables provided by the library ;

polygon.c the associated source file that contains the body of the various proposed functions.

Makefile makefile to compile the sources, generate libraries and executable. At least, 3 targets in this Makefile :

- **all** compiles everything, generates the libraries and executable.
- **lib** generates the binary code of the library libPolygon.so.
- **clean** cleans the tmp files generated during the compilation process and the various binary files

polygonmain.c the main program.

A dynamic library may be compiled using the following command line :
$gcc -Wall -Werror -ansi -pedantic -shared -fpic <source files> -o libPolygon.so

Provided Types

— the type *Point* representing a 2D-Point (x,y) in a real two dimensional space ;

— the type *Polygon* representing a general polygon **as a circular doubly linked list of Points, this list should be designed as datatype maintaining an access to the first point as well as the number of points currently stored in the list. The considered polygons could be convex or concave but they do not contain any hole (see Figure 1).** The various points composing a polygon are indexed from 1 to N.

Provided Functions

1. $createPoint : Real \times Real \rightarrow Point$, creates a 2D-*Point* according to the specified abscissa and ordinate ;
2. $createPolygon :\rightarrow Polygon$, creates an empty polygon ;
3. $addPoint : Polygon \times Point \rightarrow Polygon$, adds the specified point to the specified polygon ;
4. $removePoint : Polygon \times Integer \rightarrow Polygon$, removes the i^{th} point from the specified polygon (points are indexed from 1 to N in a polygon) ;
5. $unionPolygons : Polygon \times Polygon \rightarrow Polygon$, computes the union between the two specified polygons (see figure 2) ;
6. $intersectionPolygons : Polygon \times Polygon \rightarrow Polygon$, computes the intersection between the two specified polygons (see figure 3) ;

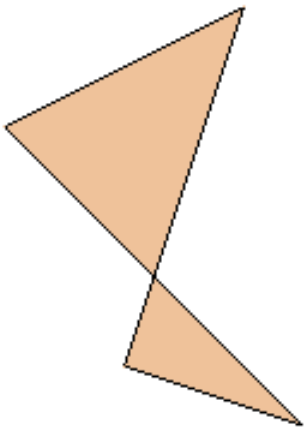

FIGURE 1 – Example of a polygon

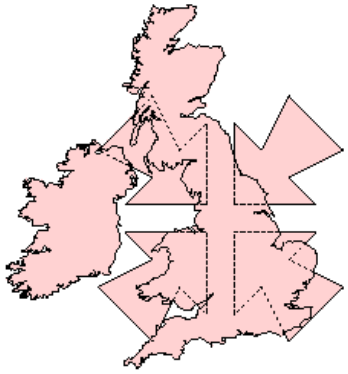

FIGURE 2 – Examples of the union between two polygons

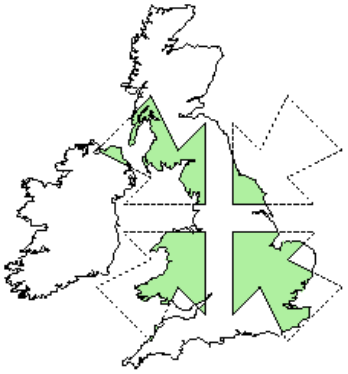

FIGURE 3 – Examples of the intersection between two polygons

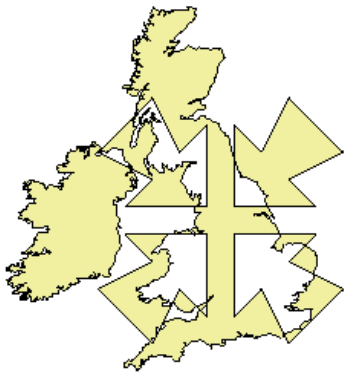

FIGURE 4 – Examples of the exclusive OR between two polygons

7. $exclusiveORPolygons : Polygon \times Polygon \rightarrow Polygon$, computes the exclusive OR between the two specified polygons (see figure 4) ;
8. $differencePolygons : Polygon \times Polygon \rightarrow Polygon$, computes the difference between the two specified polygons (see figure 5) ;

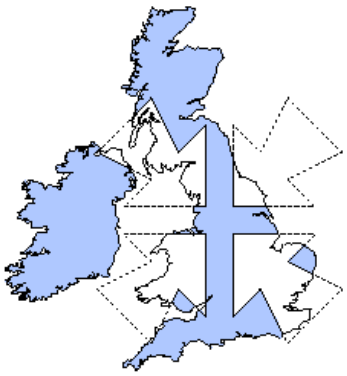

FIGURE 5 – Examples of the difference between two polygons

9. $containsPoint : Polygon \times Point \rightarrow Boolean$, returns true if the specified polygon contains the specified point, false otherwise ;
10. $containsPolygon : Polygon \times Polygon \rightarrow Status$, returns a `Status` enumeration type that could take the following values :

    INSIDE if the second polygon is fully inside the first one ;

    OUTSIDE if the second polygon is fully outside the first one ;

    INTERSECT if the second polygon is partially inside/outside the first one, in other words intersecting the second one ;

    ENCLOSING if the first polygon is fully inside the second one ;

    EQUAL both polygons are exactly equal.
11. $centralSymmetry : Polygon \times Point \rightarrow Polygon$, computes the central symmetry of the specified polygon according to the specified point ;

12. $rotatePolygon : Polygon \times Point \times Float \rightarrow Polygon$, computes the rotation of the specified polygon according to the specified point and the specified angle in radians (counterclockwise) ;
13. $scalePolygon : Polygon \times Float \rightarrow Polygon$, scales the specified polygon according to the specified factor ;
14. $translatePolygon : Polygon \times Point \times Point \rightarrow Polygon$, computes the translation of the specified polygon according to the vector defined by the two specified points (vector going from the first specified point to the second one) ;
15. $convexhullPolygon : Polygon \rightarrow Polygon$, computes the convex hull [1] of the specified polygon using the Graham's method [2] ;
16. $printPoint : Point \rightarrow \varnothing$, print a 2D-Point on the console terminal ;
17. $printPolygon : Polygon \rightarrow \varnothing$, print a Polygon on the console terminal ;
18. $toString : Polygon \rightarrow char*$, outputs the list of points of a Polygon into a string **strictly** respecting the following template (no space at all) :

$$[[x_1, y_1], [x_2, y_2], [x_n, y_n]]$$

19. any other useful functions you need.

3.2 Main program and graphical user interface

To provide a main program using the previous library (libPolygon.so) and enabling a user to test all of its provided functionalities in a simple and friendly way.

This main program, named *polygonmain.c*, will be compiled using the following command line :
$ gcc -Wall -Werror -ansi -pedantic -L<librairy directory> polygonmain.c -o polygon.exe -lPolygon

In running the program, the variable LD_LIBRARY_PATH have to specify the directory containing the library *libPolygon.so*.

4 The project deliverables

A ZIP archive or TAR.GZ
— the report in pdf format describing every algorithms and the corresponding results.
— Makefile to build the library and get the executable associated to the file polygonmain.c
— source files of the libraries : polygon.c and polygon.h and all other files containing the definition of the other required types (e.g. doubly linked list of something).
— the binary of the library libPolygon.so
— the main program polygonmain.c

1. `http://en.wikipedia.org/wiki/Convex_hull`
2. `http://en.wikipedia.org/wiki/Graham_scan`