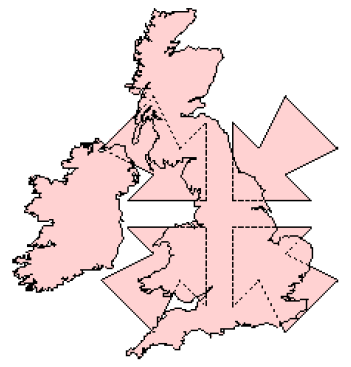
Algorithmic and C Programming - LO27

Project: 2D Polygon Library



###### Summary

[I. Main goal of the project: 3](#_Toc408164592)

[II. Explanations: 3](#_Toc408164593)

[1. Data structure: 3](#_Toc408164594)

[Type Point 3](#_Toc408164595)

[Type Polygon 3](#_Toc408164596)

[Type PolyList 4](#_Toc408164597)

[2. Mains functions : 4](#_Toc408164598)

[Polygon unionPolygons(Polygon poly1, Polygon poly2) 4](#_Toc408164599)

[Polygon intersectionPolygons(Polygon p1, Polygon p2) 4](#_Toc408164600)

[PolyList exclusiveORPolygons(Polygon p1, Polygon p2) 4](#_Toc408164601)

[PolyList differencePolygons(Polygon p1, Polygon p2) 4](#_Toc408164602)

[Polygon convexHullPolygon(Polygon poly) 5](#_Toc408164603)

[III. Difficulties and optimisations: 5](#_Toc408164604)

[1. Difficulties: 5](#_Toc408164605)

[2. Optimisations: 5](#_Toc408164606)

[IV. Conclusion: 5](#_Toc408164607)

# Main goal of the project:

The main goal of this project was to provide a library of functions for manipulating 2D-polygons.We had to provide a program that allows a user to interactively test every single functions provided by this library.

The library written in C contains the types and functions that we are going to describe all along this report.

# Explanations:

## Data structure:

### Type Point

The type Point represents a 2D-Point (x,y) in a real two dimensional space. x and y are doubles that represent the coordinates of the point.

### Type Polygon

The type Polygon represents a general polygon as a circular doubly linked list of Points. This list is designed as a datatype that maintains an access to the first point and to the number of points of the polygon. The considered polygons can be convex or concave but they can’t contain any hole.

### Type PolyList

The type PolyList represents a group of polygons as a linked list of datatype Polygon. This list is designed to always maintain an access to the next polygon. This kind of list is used in the function exclusiveORPolygons that return several polygons or in the main program.

## Mains functions :

In this part we are going to explain to you the running of the mains functions of our library. We only explain the complex parts of the algorithms of these functions, that is to say without the trivial cases.

### Polygon unionPolygons(Polygon poly1, Polygon poly2)

Trivial cases: poly1 empty, poly2 empty, one polygon containing another, poly1 equals to poly2,

General case:

* First we create a polygon with all the intersections between poly1 and poly2, called intersect.
* Next we find a point of intersection on a polygon (poly1) to start the process on it.
* We create the new polygon corresponding to the union of poly1 and poly2.
* We go into a while loop that runs until we finish to look over the entire polygon (unionpoly).
* In the while loop, if the point we are looking (in poly1) is not a point of intersect, we add this point to the union, unionpoly.
* Always in this case, if the next point is not the end point (the first point of union) and if this next point is already in unionpoly we move to the previous point of poly1. If the previous condition is not verified, we move to the next point (the same that we test).
* In the other case when the point we test is already a point of intersect, we add this point to unionpoly and we find the same point in the other polygon poly2.
* Then, we swap the two polygons to carry on the second polygon poly2.
* Finally we find the good side to continue the process, looking the outgoing side. To do that we shift the point value of 0.1 to see which side we have to go to.
* At the end we return the polygon unionpoly.

### Polygon intersectionPolygons(Polygon p1, Polygon p2)

The behavior of intersectionPolygons is really close to unionPolygon, only some small differences occur in the trivial cases or in the traversal of the two polygons.

### PolyList exclusiveORPolygons(Polygon p1, Polygon p2)

Even if this function seems hard to to, we realised that with two differents call of the function differencePolygons, the result was the same as excepted.

### PolyList differencePolygons(Polygon p1, Polygon p2)

Trivial cases:

General case:

### Polygon convexHullPolygon(Polygon poly)

Trivial cases:

General case:

# Difficulties and optimisations:

## Difficulties:

At the beginning we had difficulties to understand the main goal of the project but with the time and the experience that we acquired during all of your amazing lessons, we managed to do it successfully.

We had difficulties to find how to do algorithmically the union function. That took lot of time but we finally perform to compute it.

## Optimisations:

Maybe some optimizations could be bringing to our algorithms, but we prefer to develop every function required rather than trying to optimise some of them.

To give you an example, with a dedicated function for exclusiveORPolygon, we probably could save some computation and be more efficient in the whole algorithm.

# Conclusion:

To sum up this project, we can say that was a great learning experience which allowed us to develop our teamwork skills as well as our programming skills. For instance we think we are much more rigorous when we are developing our programs. Thanks to this project, we realize how important it is to comment the algorithm so that everyone can understand how it works. Moreover, we now know the modular programming that is really important to simplify the program with the re-use of the same functions in several subroutines. We encounter some difficulties but we overcame them successfully.