# CSGI.GA.2270 – Computer Graphics

# 2022 Fall Assignments

# Due date: 10.05.2022 23:59

# Assignment 1

The goal of this first assignment is to get familiar with CMake and C++ by implementing two basic algorithms in geometry. In the first exercise, you will implement an algorithm for finding the convex hull of a point cloud in 2D. In the second exercise, you will implement a function to test whether a point is inside a polygon in 2D.

# Preparing the Environment

Follow instructions the general rules to setup what you need for the assignment. You can also install Meshlab to visualize the datasets (point cloud and polygons) that you will be using in this assignment (https://www.meshlab.net/).

In this assignment, we do not use any external library beside the standard C++ library. To represent points in 2D, we will use complex numbers, available in the standard library as `std::complex<double>`

File Format

We store point clouds as simple ASCII file in XYZ format. The file format is as follows:

```
3000
222.582 555.88 0
226.411 535.734 0
226.83 529.925 0
269.326 472.13 0
248.178 518.121 0
...
```

The first line indicates the number of points N. The following N lines contains a space-separated list of each point coordinates. Note that our case, we will ignore the 3rd coordinate when reading the data.

Polygons will be stored using a simple OBJ file format, which is a list of points, followed (in this case) by a list of edges forming the polygon:

```
v 115.976000 432.047000 0
v 116.430000 529.603000 0
v 123.848000 534.238000 0
...

l 1 2
l 2 3
l 3 4
```

Note that indexing starts with 1 in OBJ files, so remember to shift indices accordingly.

# Ex.1: Convex Hull

## Description

To compute the convex hull of a set of points in 2D, we will use Graham's scan algorithm. Imagine all the points are stakes hammered in the ground. Now start pulling a rope from one corner of the point cloud, going around the stakes before coming back to the starting point. If you pull tightly on the rope, it will take the shape of the convex hull of this point cloud. This is the basic principle of Graham's scan algorithm for computing the convex hull of a point cloud. More formally, the pseudo-code can be given as follows:

1. Find the lower-left point in the input point cloud P0.
2. Sort all input points counter-clockwise with respect to P0.
3. Greedily compute the convex-hull polygon by iterating over the sorted points. Whenever a "right-turn" is encountered, pop the middle-point from the hull.

https://en.wikipedia.org/wiki/File:GrahamScanDemo.gif

### Code

After compiling the code following the process described in the general instructions, you can launch the program from command-line as follows:

```
mkdir build; cd build; cmake ..; make
./convex_hull ../data/points.xyz output.obj
```

Once you complete the assignment, you should be able to open the resulting file with Meshlab:

```
meshlab output.obj
```

### Tasks

Each `TODO` in the provided code corresponds to part that you need to fill to complete the assignment.

1. Complete the function to compute if three points A, B, C form a salient angle (used to determine if the rope makes a *right turn* or not).
2. Complete the comparison structure to sort points in counter-clockwise order.
3. Implement Graham's scan algorithm using the information provided above.
4. Complete the function to read .xyz from a file. While it is perfectly fine to use C's `printf()` and `scanf()` functions in C++, the idiomatic way to read/write data is to use `std::iostream<>`, which provides additional type-safety compared to C's equivalent.

## Ex.2: Point In Polygon

### Description

In this second exercise, the goal is to test whether a point is inside a polygon or not. This is called the Point In Polygon test. The principle is as follows: given a point Q and a polygon P, draw a line from the query point Q to some other point far away in the plane (which should be outside P). Then, count the number of segments in P that this line intersects. This method to compute the inside/outside of a polygon is also called the even-odd rule.

## Code

Compile and run the code as follows:

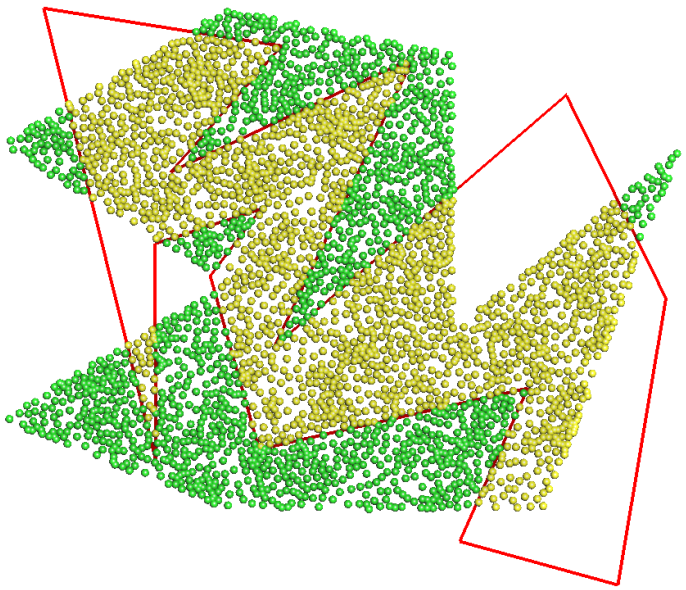
```
mkdir build; cd build; cmake ..; make
./point_in_polygon ../data/points.xyz ../data/polygon.obj result.xyz
```

## Tasks

1. Write the functions to load an OBJ polygon, and to save an XYZ point cloud.
2. Write the function to determine whether two segments intersect, and compute the location of the intersection.
3. Compute the bounding box of the input polygon, then find a point that is guaranteed to be outside the input polygon.
4. Write the *Point In Polygon* function.

## Result

Here is an image of the dataset provided in this assignment:



- The input `points.xyz` are shown in green.
- The polygon in `polygon.obj` is shown in red.
- The points from `points.xyz` which are inside `polygon.obj` are shown in yellow.

References: Daniele Panazzo