Convex Hull Simulation with OpenGL

Cagin Agirdemir
Assignment Report in scope of
Computational Geometry Course
Hacettepe University, Ankara
caginagirdemir@hacettepe.edu.tr

Abstract

This assignment aims to consist of a convex hull by placing points randomly from users. Thereafter, a plastic rob comes to the convex hull when the user clicks to start the simulation. While plastic rob faced to intersection becomes convex hull shapes and stretch like a stretch elastic band.

Keywords

convex hull, simulation, shaping

I. Introduction

In initially, users must be entered points though mouse click on the empty scene. There is not have any limitation in terms of points count. When users click for the last point, do right click for the menu. In this menu, must choose convex hull options, in this way program draw the best convex hull shape accordingly to user points. I will report this simulation gradually from convex hull calculation to elastic band stretching calculation.

II. Convex Hull Definition

In geometry, the convex hull or convex envelope or convex closure of a shape is the smallest convex set that contains it. The convex hull may be defined either as the intersection of all convex sets containing a given subset of a Euclidean space, or equivalently as the set of all convex combinations of points in the subset. For a bounded subset of the plane, the convex hull may be visualized as the shape enclosed by a rubber band stretched around the subset.

Convex hulls of open sets are open, and convex hulls of compact sets are compact. Every compact convex set is the convex hull of its extreme points. The convex hull operator is an example of a closure operator, and every antimatroid can be represented by applying this closure operator to finite sets of points. The algorithmic problems of finding the convex hull of a finite set of points in the plane or other low-dimensional Euclidean spaces, and its dual problem of intersecting half-spaces, are fundamental problems of computational geometry. [1]

III. Convex Hull Calculation

We have discussed Jarvis's Algorithm for Convex Hull. The worst case time complexity of Jarvis's Algorithm is O(n^2).

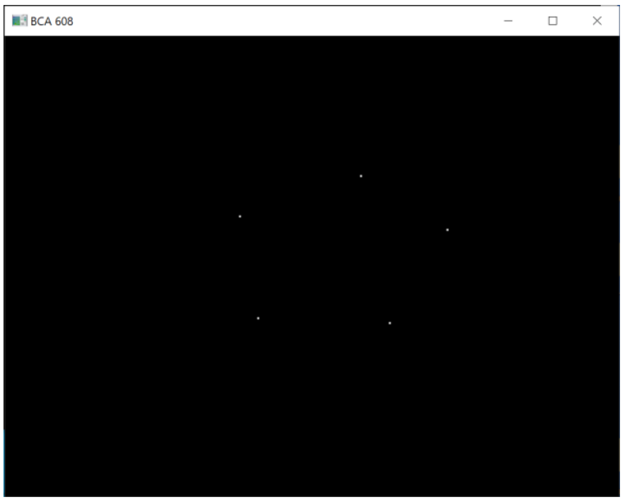

Figure 1. Select Convex Hull points from user

Using Graham's scan algorithm, we can find Convex Hull in O(nLogn) time. Following is Graham's algorithm

Let points[0..n-1] be the input array.

Find the bottom-most point by comparing y coordinate of all points. If there are two points with the same y value, then the point with smaller x coordinate value is considered. Let the bottom-most point be P0. Put P0 at first position in output hull.

Consider the remaining n-1 points and sort them by polar angle in counter clockwise order around points[0]. If the polar angle of two points is the same, then put the nearest point first.

After sorting, check if two or more points have the same angle. If two more points have the same angle, then remove all same angle points except the point farthest from P0. Let the size of the new array be m.

If m is less than 3, return (Convex Hull not possible)

Create an empty stack ‘S’ and push points[0], points[1] and points[2] to S.

Process remaining m-3 points one by one. Do following for every point ‘points[i]’

```
a. Keep removing points from stack while orientation of following 3 points is not counter clockwise (or they don’t make a left turn).
        a.1) Point next to top in stack
        a.2) Point at the top of stack
        a.3) points[i]
b. Push points[i] to S and Print contents of S
```

The above algorithm can be divided into two phases.

Phase 1 (Sort points): We first find the bottom-most point. The idea is to pre-process points be sorting them with respect to the bottom-most point. Once the points are sorted, they form a simple closed path

What should be the sorting criteria? computation of actual angles would be inefficient since trigonometric functions are not simple to evaluate. The idea is to use the orientation to compare angles without actually computing them (See the compare() function below)

Phase 2 (Accept or Reject Points): Once we have the closed path, the next step is to traverse the path and remove concave points on this path. How to decide which point to remove and which to keep? Again, orientation helps here. The first two points in sorted array are always part of Convex Hull. For remaining points, we keep track of recent three points, and find the angle formed by them. Let the three points be prev(p), curr(c) and next(n). If orientation of these points (considering them in same order) is not counter clockwise, we discard c, otherwise we keep it. Following diagram shows step by step process of this phase. [2]

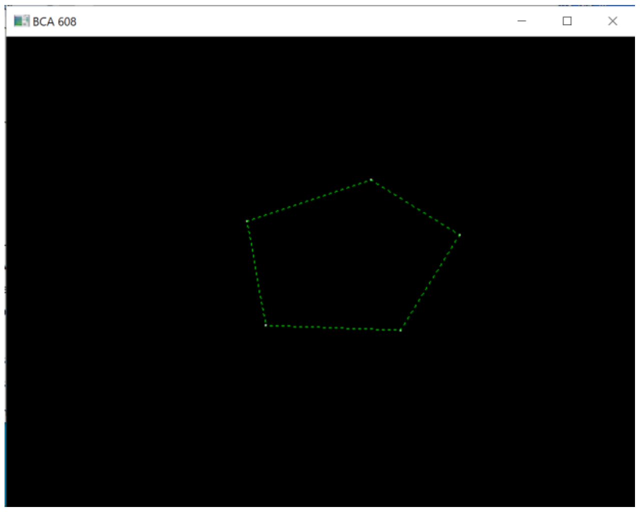

Figure 2. Calculated Convex Hull

IV. Elastic Band Stretching Working

After calculation of Convex Hull user would selected "draw elastic band" though left click menu.

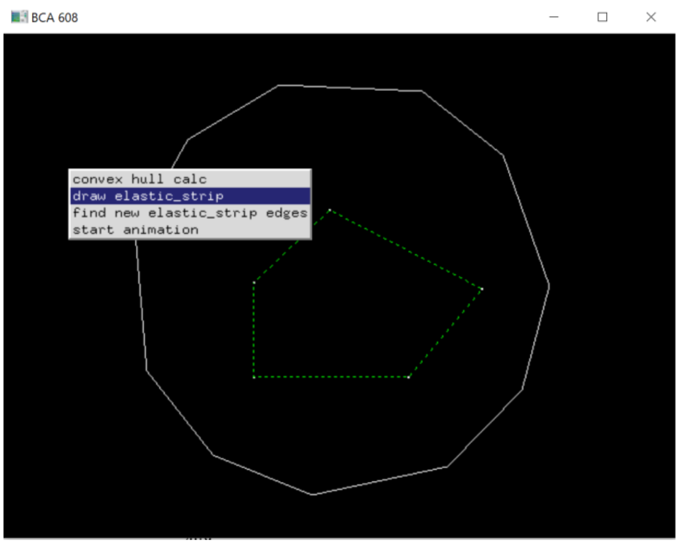

Figure 3. Drawing elastic band

The elastic band consists of ten points which are defined before from me. But, these points not enough for stretching. In the program, should determine a new point according to Convex Hull edges. Therefore, elastic band points which will be overleaping with Convex Hull, determine from every Convex Hull point to the closest elastic band border.

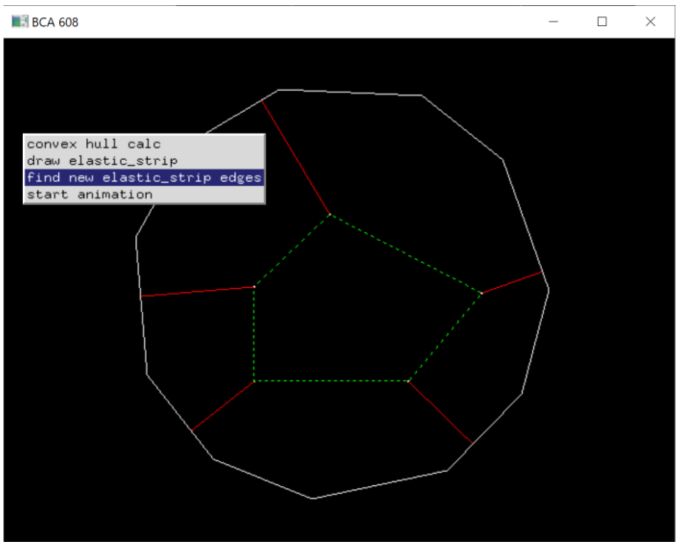

Figure 4. Determine new elastic band edges

In the last part, when the user clicks start animation, all elastic band points move to Convex Hull along by all slope comes from between the elasctic band point and the midpoint of Convex Hull. direction. There is a delay command for animation reality and make recognizable that animation.

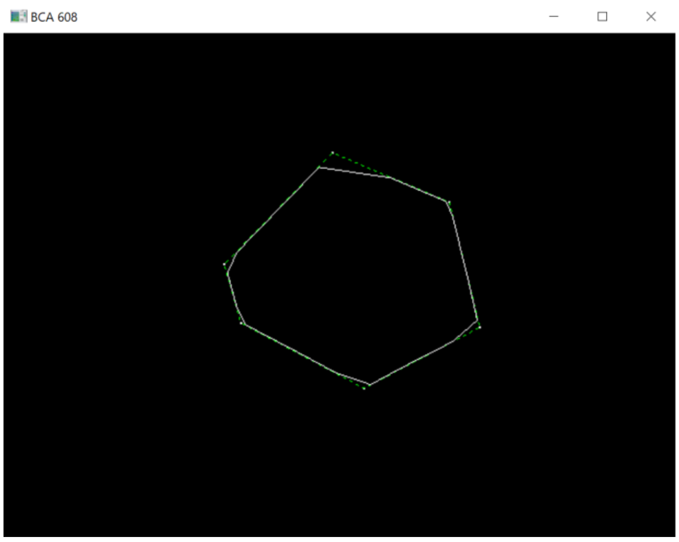

Figure 5. Finished Animation

There is some miss between the elastic band and Convex Hull overleap because of the slope direction arrange by the midpoint. Therefore occurs a difference between where it should intersect and where it intersects.

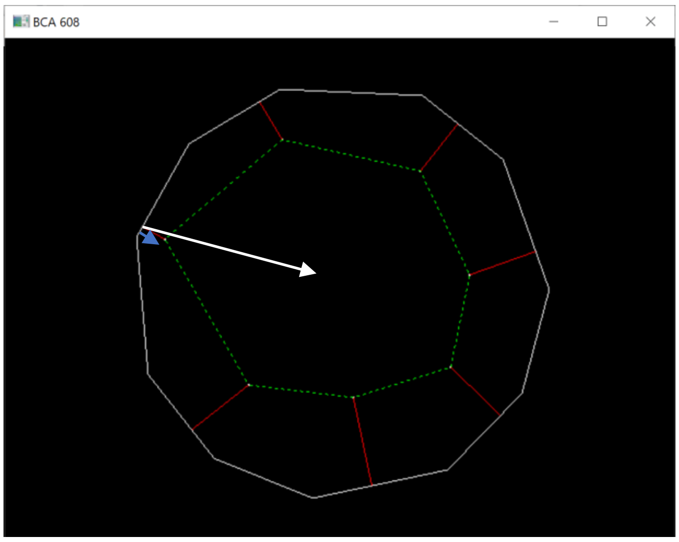

Figure 5. Point to midpoint direction (white) and must be direction (blue)

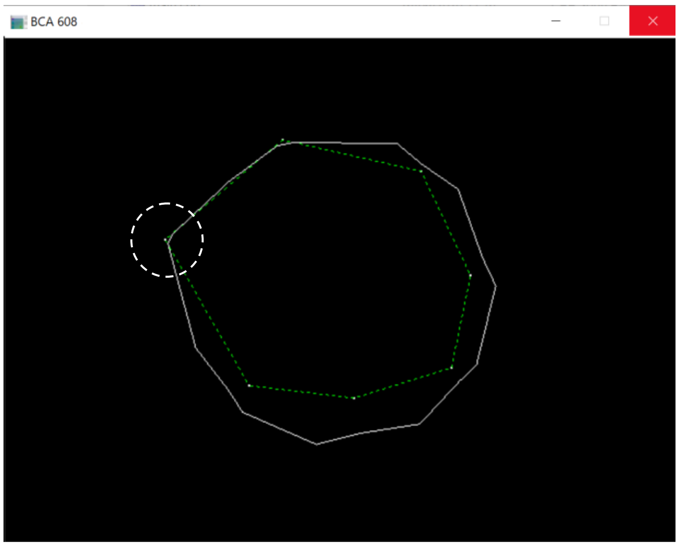

Figure 6. The Difference

References

[1] https://en.wikipedia.org/wiki/Convex_hull#Equivalence_of_definitions, 2019.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, 3rd edition, PHI Learning Pvt. Ltd., 2009.