COT 4521-001: Introduction to Computational Geometry (Fall 2018)

Project 1: Line Segment Intersections

1 Objectives

In this assignment you will implement some of the basic algorithms of line segment intersections. The will also have the opportunity to work on optimizing the performance of your code.

2 Ground Rules

This assignment is intended to be done alone. You may ask others for help with figuring out strategies. However, the code must be yours.

3 Assignment Instructions

- Download the provided skeleton code and complete the empty functions in `Edge.pde` and `LineSegmentSet.pde`.
  - `boolean intersectionTest( Edge other )` — This function returns the result of an intersection test (true or false) for 2 line segments. You should use the cross product based method discussed in class.
  - `Point intersectionPoint( Edge other )` — This function returns the exact intersection point between 2 line segments. You should use the parametric intersection method, with the emphasis on CORRRECTNESS of the result it produces.
  - `Point optimizedIntersectionPoint( Edge other )` — This function returns the exact intersection point between 2 line segments. You can use whatever intersection method you like, with the emphasis on SPEED.
  - `public static void NaiveLineSegmentSetIntersection( ArrayList<Edge> input_edges, ArrayList<Point> output_intersections )` — Given a set of input segments, this function finds all intersections between those segments. This function should use the naive $O(n^2)$ approach. Pairwise intersections should tested using the `intersectionPoint()` function. Emphasis is on CORRECTNESS.
  - `public static void OptimizedLineSegmentSetIntersection( ArrayList<Edge> input_edges, ArrayList<Point> output_intersections )` — Given a set of input segments, this function finds all intersections between those segments. This function should be optimized in some way (you don't need to implement the full line sweep algorithm, but maybe take some inspiration). Pairwise intersections should tested using the `optimizedIntersectionPoint()` function. Results should be correct, but the emphasis is on SPEED.

- To help test your code, the skeleton code will provide random line segments and show the intersections points your code finds. You can change the number of line segments by using the '+'/'–' keys while running your sketch. We also provided 2 functionality for comparing naive and optimized versions of your algorithms:
  - `performanceTest()` (press 'p' when running your sketch) — This function will run a series of random line segments sets through both of your algorithms and compare the performance results.
  - `compareOutput()` (press 'c' when running your sketch) — This function will run a series tests that check if the output of the naive of random line segments sets through both of your algorithms and compare the performance results.

4 Submission

Compress your sketch into a single zip file and upload to canvas.