COT 4521: Intro. to Computational Geometry (Fall 2020)

Project 1: Line Segment Intersections

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 (MOSS will be used).

Submission

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

Assignment Instructions

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.

- Download the provided skeleton code and complete the empty functions in `AABB.pde`, `Point.pde`, `Edge.pde`, and `LineSegmentSet.pde`. Of the highest importance:
  - `boolean AABB::intersectionTest( AABB other )` — This function returns the result of an intersection test (true or false) for 2 axis aligned bounding boxes.
  - `Point Edge::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 Edge::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 OptimizedLineSegmentSetIntersectionAABB( ArrayList<Edge> input_edges, ArrayList<Point> output_intersections )` — Given a set of input segments, this function finds all intersections between those segments. This method should use the axis aligned bounding box method discussed in class. However, you do NOT need to use an interval tree.
- 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 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 algorithms are *consistent* (not necessarily correct).