Computational Geometry Aufgabe 1

Aufgabenstellung:

Ziel der Aufgabe ist es ein Programm zu schreiben, in welches gegebene Liniensegmente eingelesen werden können, welches dann die Anzahl an Schnittpunkten liefert.

Vorgehensweise:

Zunächst wird die Datei mit den Liniensegmenten Zeilenweise eingelesen, dabei enthält sie vier Koordinaten pro Zeile. Daraus ergeben sich die Koordinaten zweier 2D-Punkte, nämlich die des Starts- und die des Endpunkts.

Als Algorithmus wird naiv über alle Liniensegmente iteriert, wobei für jedes Liniensegment nochmals eine Iteration über alle verbleibenden Liniensegmente stattfindet. Auf diese Weise können alle Schnittpunkte erfasst werden, da jedes Segment mit jedem anderen Segment einmal überprüft wird. Zum ermitteln der Schnittpunkte wird die ccw-Formel eingesetzt. Dabei ergibt sich folgender Code:

```
inline bool are_intersecting(const Line& line1, const Line& line2)
{
    auto ccw1 = line1.ccw(line2.getStart());
    auto ccw2 = line1.ccw(line2.getEnd());
    auto ccw3 = line2.ccw(line1.getStart());
    auto ccw4 = line2.ccw(line1.getEnd());
    if (ccw1 == 0 && ccw2 == 0 && ccw3 == 0 && ccw4 == 0)
    {
        if (line1.is_between(line2.getStart()) || line1.is_between(line2.getEnd()))
        {
            return true;
        }
    }
    else if (ccw1 * ccw2 <= 0 && ccw3 * ccw4 <= 0)
    {
        return true;
    }
    return false;
}
```

Dabei tritt der zweite Fall dann ein, wenn ccw1 * ccw2 nicht positiv, sowie ccw3 * ccw4 nicht positiv ist. Dies ist dann der Fall, wenn sich ein Punkt des anderen Liniensegments links oder auf dem Liniensegment befindet, sowie ein Punkt sich rechts oder auf dem Liniensegment befindet. Der erste Fall tritt ein, wenn die Liniensegmente kolinear zueinander sind, dann gilt es nur zu prüfen, ob sich Start- oder Endpunkt des zweiten Segments innerhalb der Bounding-Box des ersten Segments aufhalten.

Auswertung:

Es ergeben sich folgende Messergebnisse (20 Durchläufe, gemitteltes Resultat):

```
Number of test runs: 20
Average test results for file 's_1000_10.dat':
        input time:                     2.65 ms
        intersection calculation time: 9.75 ms
        total time taken:               12.4 ms
        nunber of intersections found: 796
Average test results for file 's_1000_1.dat':
        input time:                     2.2 ms
        intersection calculation time: 7.75 ms
        total time taken:               9.95 ms
        nunber of intersections found: 11
Average test results for file 's_10000_1.dat':
        input time:                     18.85 ms
        intersection calculation time: 601 ms
        total time taken:               619.85 ms
        nunber of intersections found: 732
Average test results for file 's_100000_1.dat':
        input time:                     198.75 ms
        intersection calculation time: 63252.4 ms
        total time taken:               63451.2 ms
        nunber of intersections found: 77125
```

Während für 1000 Liniensegmente noch wenig Zeit benötigt wird, so wird im Vergleich zwischen 10000 und 100000 Punkten die Zeit-Komplexität von O (n^2) besonders deutlich.