INFORMATIK

Universität des Saarlandes
Fachbereich 14 – Informatik
Dr. S. Schirra / Dr. E. Schömer

SOFTWAREPRAKTIKUM WS 99/00

5. Übungsblatt

(Abgabe: 22. - 26. November 1999)

1. **Aufgabe**: CGAL und konvexe Hülle (Punkte: 7)

   Schreiben Sie ein Programm, das mithilfe von Iteratoren ein Polygon als Folge von Punkten von einem File liest und in einem Container abspeichert. Die Punkte sollen nun so skaliert und verschoben werden, dass nach der Transformation alle im Bereich $[-1.0, 1.0] \times [-1.0, 1.0]$ liegen. Benutzen Sie dazu STL-Algorithmen und Iteratoren. Lesen Sie die Dokumentation zur CGAL Funktion `convex_hull_points_2()` in Teil 2 (Basic Library) des CGAL Referenz Manuals (siehe `http://www.mpi-sb.mpg.de/GALIA/Manual`). Das Verzeichnis `/home/stud/praxprog/Ueb5` enthält ein Beispielprogramm. Benuzten Sie die Funktion, um die Folge der Punkte auf der konvexen Hülle des (transformierten) Polygons zu berechnen. Zeichnen Sie nun das transformierte Polygon und dessen konvexe Hülle in ein OpenGL Fenster.

   Hinweis zur Transformation: Bestimmen Sie zunächst eine Bounding Box für die Punkte. Verschieben Sie die Punkte nun so, dass der Mittelpunkt der Bounding Box im Ursprung liegt, d.h., berechnen Sie für jeden Punkt den Differenzvektor zum Mittelpunkt der Bounding Box. Skalieren Sie nun geeignet.

2. **Aufgabe**: Primzahliterator (Punkte: 5)

   Schreiben Sie einen Primzahliterator. Der Iterator sollte die Anforderungen an einen Inputiterator erfüllen und über die Folge der Primzahlen iterieren.

3. **Aufgabe**: Vereinfachung von Polygonzügen (Punkte: 8)

   Gegeben sei ein Polygonzug durch eine Folge von Punkten $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}$, mit $\mathbf{v_i} = [v_{i1}, v_{i2}]^T \in \mathbf{R}^2$. Wir wollen eine Vereinfachung dieses Polygonzuges mit Hilfe des Douglas-Peucker-Algorithmus (siehe `http://www.mpi-sb.mpg.de/units/ag1/teaching/praxprog9900/Uebungen/ueb5/douglas-peucker.ps.gz`) berechnen. Dieser Algorithmus lässt sich rekursiv leicht beschreiben:

   1. Man finde den Punkt $\mathbf{v_k}$ mit $1 < k < n$, der den größten Abstand zur der Gerade durch die Anfangspunkte $\mathbf{v_1}$ und $\mathbf{v_n}$ hat. Diesen Abstand bezeichnen wir mit d.
   2. Falls $d < \varepsilon$ gib die Strecke $\mathbf{v_1}\ \mathbf{v_n}$ als Lösung aus,
      ansonsten führe den Algorithmus getrennt auf den Teillisten $\mathbf{v_1}, \ldots, \mathbf{v_k}$ und $\mathbf{v_k}, \ldots, \mathbf{v_n}$ durch.

   Dabei ist eine Fehlerschranke ε vorgegeben.

   Implementieren Sie den Algorithmus von Douglas-Peucker als Templatefunktion, so dass er eine Folge von Punkten über einen Iteratorrange erhält und über einen Outputiterator eine Folge von Punkten als Ausgabe produziert. Testen Sie Implementierung anhand der Gewässerdaten im Verzeichnis `/home/stud/praxprog/Geodata`, indem Sie die Original-Daten und die vereinfachten Daten mittels OpenGL visualisieren.

   Hinweis: Zur Berechnung des maximalen Abstandes einer Reihe von Punkten von ein und der selben Gerade kann man auf das Normieren (Wurzelziehen und Division) verzichten.