Institut für Informatik
Prof. Dr. Oliver Vornberger
Lukas Kalbertodt, B.Sc.

Universität Osnabrück, 19.04.2016
http://www-lehre.inf.uos.de/~cg
Testat bis 27.04.2016, 14:00 Uhr

Übungen zu Computergrafik

Sommersemester 2016

Blatt 3: Füllen, Transformation, Kurven

> **Achtung**: Die Quellen zur Aufgabenstellung für dieses Blatt werden erst am Mittwoch, den 20.04. um ca. 14:00 Uhr veröffentlicht, da sie die Musterlösung zu Blatt 2 enthalten.
>
> **Hinweis**: Wie vom letzten Blatt bekannt, sind auch dieses Mal wieder die wichtigen Stellen im Quelltext mit einem `// TODO` Kommentar versehen.

Aufgabe 3.1: Scanline-Fill (25 Punkte)

Vervollständigen Sie die Methode `scanlineFill()` der gegebenen Klasse `Polygon`, so dass das Polygon mit Hilfe des Scanline Algorithmus gefüllt wird. Die Methode wird automatisch von `paint` aufgerufen, wenn es sich um ein gefülltes Polygon handelt. Dieses kann in der Auswahlbox ausgewählt und somit erstellt werden.

Zur Vereinfachung implementieren die Klassen `Point` und `Line` bereits das Interface `Comparable`. Objekte vom Typ `Line` werden anhand des `y`-Wertes des Anfangspunktes verglichen, Objekte vom Typ `Point` anhand ihres `x`-Wertes. Dadurch ist es beispielsweise sehr einfach möglich, eine Liste von `Lines` mit dem Aufruf `Collections.sort(liste);` zu sortieren.

Aufgabe 3.2: Polygon-Symbol (10 Punkte)

Um in den folgenden Aufgaben schnell und einfach ein halbwegs komplexes Polygon zu erstellen, sollen Sie in dieser Aufgabe ein solch fertiges Polygon einbauen. Dazu müssen Sie lediglich in der Datei `OwnSymbolListener.java` an der markierten Stelle mehrere Punkte hinzufügen. Die Punkte *können* Sie hardcoden, also x- und y-Werte direkt in den Quelltext schreiben.

Das erzeugte Polygon kann ein Symbol Ihrer Wahl sein, wie z. B. ein Plus, eine Krone oder das Wappentier Ihres Hauses. Wichtig ist allerdings, dass das Symbol am Ort des Mausklicks und nicht immer an der selben Stelle entsteht.

Aufgabe 3.3: 2D-Transformationen (25 Punkte)

Machen Sie sich mit der Listener-Klasse `TransformListener` und der neuen Delegate-Methode `transformGraphicalObjectsNear(Point p)` vertraut. Implementieren Sie die Methoden `isNear(Point p)` und `transformBy(Matrix m)` der Klasse `Polygon`, so dass `isNear()` `true` zurück gibt, wenn der übergebene Punkt innerhalb des Polygons liegt, `false` andernfalls. `transformBy()` soll das Polygon anhand der übergebenen Transformationsmatrix dauerhaft verändern.

Die Funktionalität lässt sich durch Auswahl des Transformationswerkzeuges in der ComboBox testen. Rechts im Fenster können mehrere Matrizen zu dem aktuellen Matrix-Stack hinzugefügt werden.

Das Polygon-Transformationwerkzeug wendet alle Matrizen dieses Stacks nacheinander an. Falls eine Zelle in den Eingabefeldern der Matrix leer ist, wird diese als 0 interpretiert.

Aufgabe 3.4: Bezier-Kurven (20 Punkte)

Implementieren Sie den Algorithmus von de Casteljau in der Klasse `BezierCurve`, um die Kurve zu zeichnen. Zur besseren Veranschaulichung werden in dem gegebenen Code die Stützpunkte gezeichnet. Als Drawable zur Vorschau beim Zeichnen der Bezier-Kurve wird `DashedPolygon` verwendet; dies hat nichts mit Ihrer Implementation von `BezierCurve` zu tun.

Aufgabe 3.5: Fragen (20 Punkte)

Beantworten Sie Ihrem Tutor Fragen zur Vorlesung.