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

Universität Osnabrück, 12.04.2016
http://www-lehre.inf.uos.de/~cg
Testat bis 20.04.2016, 14:00 Uhr

Übungen zu Computergrafik

Sommersemester 2016

Blatt 2: 2D-Zeichnen

Aufgabe 2.1: Linien und Rechtecke (40 Punkte)

Importieren Sie das im Anhang befindliche JAR-Archiv `draw2d-uebung.jar` in Ihre Entwicklungsumgebung.

Die Applikation zeigt beim Start eine leere Zeichenfläche mit einer Auswahlbox und einem Button zum Löschen aller bereits gezeichneten Objekte. Die Applikation soll auf diesem Aufgabenblatt durch die Möglichkeit, die verschiedenen in der Auswahlbox vorgegebenen grafischen Objekte zeichnen zu können, ergänzt werden. Nehmen Sie das bereits implementierte Zeichnen von *Punkten* als Vorlage, um die übrige Funktionalität zu implementieren.

In dieser Aufgabe sollen Sie sich mit der Klassen- und Paketstruktur der Applikation vertraut machen. Erklären Sie Ihrem Tutor welcher Teil der Applikation für welche Aufgabe zuständig ist, und gehen Sie dabei erneut auf das Model-View-Controller-Entwurfsmuster ein.

In dieser Aufgabe geht es außerdem um das Zeichnen von Linien und Rechtecken. Implementieren Sie dazu den aus der Vorlesung bekannten Bresenham-Algorithmus in Ihrer Applikation. Bearbeiten Sie dazu die Klassen `Line` und `Rectangle` – wichtige Stellen sind mit `TODO: (A1)` markiert. Implementieren Sie die `paint` Methode der `Line` Klasse mit Hilfe des Bresenham-Algorithmus. Es ist für diese Aufgabe nicht nötig, andere Dateien, als die beiden genannten, zu bearbeiten.

Aufgabe 2.2: Anti-Aliasing (15 Punkte)

Implementieren Sie eine weitere Klasse `AntiAliasedLine` zum Zeichnen von Linien, bei denen die Aliasing-Effekte durch eine einfache Änderung des Bresenham-Algorithmus verringert werden. Wie in Aufgabe 1 sind wichtige Stellen in `AntiAliasedLine` mit `TODO: (A2)` gekennzeichnet. Auch hier reicht alleine die Bearbeitung der genannten Klasse aus.

Hinweis: Die Vorschau der Linie, die Sie beim Bewegen der Maus angezeigt bekommen, ist erst in Aufgabe 3 wichtig. Insbesondere nutzt diese Vorschau *nicht* die Klasse `AntiAliasedLine`. Um Ihre Implementation von Anti-Aliasing zu testen, müssen Sie die Linie durch einen zweiten Klick endgültig zeichnen lassen.

Wenn Sie beim Nutzen des Bresenham-Algorithmus einen Pixel mit Hilfe von `setPixel` setzen, kennen Sie auch den derzeitigen Fehler, also den Betrag, den der entsprechende Pixel in einer Koordinate vom optimalen Wert abweicht. Nutzen Sie diesen Fehler, um die Helligkeit des Pixels entsprechend zu verändern. Dazu können Sie `setPixel` einen Intensitätswert zwischen 1 und 0 übergeben. Setzen Sie außerdem neben den soeben gezeichneten Pixel einen weiteren, sodass die Summe der Helligkeiten wieder 1 ergibt, also 100%-iges schwarz.

Beispiel: Sie würden mit dem Bresenham-Algrithmus den Pixel mit den Koordinaten (10,10) einfärben. Der Fehler beträgt aber derzeit 0,2. Sie zeichnen die Linie flach nach oben rechts, so dass -

sobald der Fehler 0,5 überschreitet - der y-Wert um 1 verringert (Bedenken Sie die Umkehrung der y-Achse in Java!) werden wird. Um einen Anti-Aliasing-Effekt zu erzeugen, zeichnen Sie den Punkt bei den Koordinaten (10,10) also nicht in 100%-igem schwarz, sondern nur mit 80%-igem Farbwert. Außerdem setzen Sie an die Koordinaten (10,9) einen weiteren Pixel mit 20%-igem Farbwert.

Aufgabe 2.3: Gestrichelte Vorschau (15 Punkte)

Das Ihnen bereitgestellte Programmgerüst zeichnet bereits eine *Vorschau* der zu zeichnenden Objekte beim Bewegen der Maus. Dazu werden die Klassen `DashedLine` und `DashedRectangle` benutzt. Da diese beiden Klassen im vorgegebenen Code schon von `Line` respektive `Rectangle` erben, sieht die Vorschau genau so aus, wie das resultierende Objekt.

Erweitern Sie die beiden genannten Klassen so, dass die Vorschau beider Objekte mit gestrichelten statt durchgezogenen Linien gezeichnet wird. Als Abstand der Lücken in der gestrichelten Linie sollten Sie einen sinnvollen Wert nutzen. Zeichnen Sie die Vorschau außerdem im `XOR`-Mode, sodass jeder gesetzte Pixel der Vorschau-Linie den bisher gesetzen Pixel invertiert.

Achten Sie weiterhin darauf, nicht den kompletten Bresenham-Code aus Aufgabe 1 zu kopieren und anzupassen. Auf diverse Wege ist es möglich, hier doppelten Code zu vermeiden!

Aufgabe 2.4: Polygone (30 Punkte)

Erweitern Sie die Applikation um die Fähigkeit Polygone zu zeichnen. Beim Zeichnen eines Polygons soll der Benutzer wie in der vorherigen Aufgabe durch eine *gestrichelte* Linie unterstützt werden. Der Nutzer kann durch Klicken beliebig viele Polygonpunkte setzen. Das Polygon wird (ab-)geschlossen, indem der Benutzer wieder auf den Anfangspunkt klickt. Sorgen Sie jedoch dafür, dass der Nutzer den Anfangspunkt nicht pixel-genau treffen muss, sondern dass schon ein Klick in die *Nähe* des Startpunktes ausreicht.

Erstellen Sie hierzu die zusätzlichen Klassen `Polygon` und `DashedPolygon` (im model) und `PolygonListener` (im controller). Stützen Sie sich zur Implementation dieser Klassen auf die schon vorhandenen Implementationen (insbesondere z. B. `LineListener`). Achten Sie außerdem auf die mit `TODO: (A4)` markierten Stellen in den gegebenen Dateien.